

2012

Towards Confident Body Sensor Networking

Zhen Ren

College of William & Mary - Arts & Sciences

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Ren, Zhen, "Towards Confident Body Sensor Networking" (2012). *Dissertations, Theses, and Masters Projects*. Paper 1539623606.

<https://dx.doi.org/doi:10.21220/s2-haez-0y29>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

Towards Confident Body Sensor Networking

Zhen Ren

Zhenjiang, Jiangsu, China

Bachelor of Science, Nanjing University, 2003

**A Dissertation presented to the Graduate Faculty
of the College of William and Mary in Candidacy for the Degree of
Doctor of Philosophy**


Department of Computer Science

**The College of William and Mary
August 2012**

APPROVAL PAGE

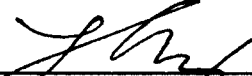
This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy



Zhen Ren

Approved by the Committee, June 2012



Committee Co-Chair

Assistant Professor Gang Zhou, Computer Science
College of William and Mary



Committee Co-Chair

Associate Professor Haining Wang, Computer Science
College of William and Mary



Associate Professor Qun Li, Computer Science
College of William and Mary



Associate Professor Weizhen Mao, Computer Science
College of William and Mary



Associate Professor Xue Liu, Computer Science
McGill University

COMPLIANCE PAGE

Research approved by

Protection of Human Subjects Committee

Protocol number(s): PHSC-2010-11-28-7026-pdkieffaber

Date(s) of approval: 2011-01-05

ABSTRACT PAGE

With the recent technology advances of wireless communication and lightweight low-power sensors, Body Sensor Network (BSN) is made possible. More and more researchers are interested in developing numerous novel BSN applications, such as remote health/fitness monitoring, military and sport training, interactive gaming, personal information sharing, and secure authentication. Despite the unstable wireless communication, various confidence requirements are placed on the BSN networking service. This thesis aims to provide Quality of Service (QoS) solutions for BSN communication, in order to achieve the required confidence goals.

We develop communication quality solutions to satisfy confidence requirements from both the communication and application levels, in single and multiple BSNs. First, we build communication QoS, targeting at providing service quality guarantees in terms of throughput and time delay on the communication level. More specifically, considering the heterogeneous BSN platform in a real deployment, we develop a radio-agnostic solution for wireless resource scheduling in the BSN. Second, we provide a QoS solution for both inter- and intra-BSN communications when more than one BSNs are involved. Third, we define application fidelity for two neurometric applications as examples, and bridge a connection between the communication QoS and application QoS.

©Copyright August 2012

Zhen Ren

All rights reserved

TABLE OF CONTENTS

| | |
|---|------------|
| Dedication | v |
| Acknowledgments | vi |
| List of Tables | vii |
| List of Figures | vii |
| 1 Introduction | 2 |
| 1.1 Building Communication QoS | 4 |
| 1.2 Connecting Communication QoS with Application QoS | 6 |
| 1.3 Exposing Communication QoS Availability to Other BSNs | 9 |
| 1.4 Contributions | 10 |
| 1.5 Dissertation Organization | 12 |
| 2 Related Works | 14 |

| | | |
|----------|--|-----------|
| 2.1 | Building Communication QoS | 14 |
| 2.2 | Connecting Communication QoS with Application QoS | 16 |
| 2.3 | Exposing Communication QoS Availability to Other BSNs | 18 |
| 3 | Building Communication QoS | 20 |
| 3.1 | Problem Definition and Analysis | 21 |
| 3.1.1 | Group-Polling v.s. Individual-Polling | 21 |
| 3.1.2 | Throughput Assurance | 23 |
| 3.1.3 | Joint Assurance of Throughput and Time Delay | 24 |
| 3.2 | BodyT2 Design | 30 |
| 3.2.1 | Admission Control | 30 |
| 3.2.2 | Time Resource Scheduling | 35 |
| 3.2.3 | Enforcing Time Schedule on VMAC | 38 |
| 3.3 | Performance Evaluation | 39 |
| 3.3.1 | Performance Results of TelosB Mote Lab Tests | 42 |
| 3.3.2 | Performance Results of Real Body Experiments in an Android Phone- centric BSN | 45 |
| 3.4 | Conclusions | 47 |
| 4 | Connecting Communication QoS with Application QoS | 49 |
| 4.1 | Motivation | 50 |
| 4.1.1 | Communication Pattern | 51 |
| 4.1.2 | EEG-based Neurometric Application and Fidelity Definition | 55 |

| | | |
|----------|--|-----------|
| 4.1.2.1 | Mild Cognitive Impairment (MCI) Detection using Tsallis Entropy (qEEG) Ratio | 55 |
| 4.1.2.2 | Depression Detection using Cerebral Asymmetry Score | 58 |
| 4.1.3 | Mismatch between Communication Pattern and Application Fidelity | 60 |
| 4.2 | Problem Definition and Analysis | 62 |
| 4.2.1 | Energy Minimization | 63 |
| 4.2.2 | Application Fidelity Maximization | 64 |
| 4.3 | Exploiting Data Sensitivity to Minimize Energy Consumption | 67 |
| 4.3.1 | Data Decimation for a Single Application | 67 |
| 4.3.2 | Data Decimation for Multiple Applications | 68 |
| 4.4 | Exploiting Data Sensitivity to Maximize Application Fidelity | 70 |
| 4.5 | Evaluation | 70 |
| 4.5.1 | Evaluation of the Energy Minimization Algorithm | 72 |
| 4.5.2 | Evaluation of the Fidelity Maximization Algorithm | 74 |
| 4.6 | Conclusions | 79 |
| 5 | Exposing communication QoS availability to other BSNs | 81 |
| 5.1 | BuddyQoS Overview | 82 |
| 5.2 | BuddyQoS Hybrid MAC Design | 85 |
| 5.3 | Inter-BSN Admission Controller and Resource Scheduler | 86 |
| 5.3.1 | Communication Paradigm for Shared BSNs | 87 |
| 5.3.2 | Resource Estimation for Shared BSNs | 90 |
| 5.3.2.1 | Resource Estimation for Management Information Exchange | 90 |

| | | |
|----------|---|------------|
| 5.3.2.2 | Resource Estimation for Data Delivery | 91 |
| 5.3.3 | Admission Decisions | 96 |
| 5.3.4 | Resource Scheduler | 97 |
| 5.4 | Buddy Management Design | 98 |
| 5.4.1 | Neighbor Discovery | 98 |
| 5.4.2 | Sensor Sharing Coordination | 100 |
| 5.5 | Performance Evaluation | 101 |
| 5.5.1 | Performance Comparison with Default CSMA Release in TinyOS . . | 103 |
| 5.5.2 | Performance Results with an Increasing Number of Neighboring BSNs | 106 |
| 5.6 | Conclusion | 107 |
| 6 | Conclusion | 109 |
| | Bibliography | 113 |
| | Vita | 119 |

*To my father and mother,
for their great love and support.*

ACKNOWLEDGMENTS

The work in this dissertation would not have been done without my advisors, Dr. Gang Zhou and Dr. Haining Wang. I would like to express my deepest gratitude to them for their patient guidance and support.

I am grateful to my honorable committee members, Qun Li, Weizhen Mao, and Xue Liu, for their valuable comments.

Also, I have been working with a wonderful group of outstanding colleagues, Matt Keally, Andy Pyles, Xin Qi, Yantao Li, and George Simmons. I appreciate all their help and feedback on work.

Finally, I would like to thank my families and friends. The unconditional love from my father and my mother has given me power in both my work and my life. Thanks to my dear friends, Jie Xu, Qian Si, Zheng Zhang, Yunlian Jiang, Chenying Zhang, and Yuan Li, for their company and help through the years.

LIST OF TABLES

| | | |
|-----|---|-----|
| 3.1 | Interference Settings | 41 |
| 4.1 | Notations for Problem Definitions | 63 |
| 5.1 | System Parameter Configuration | 102 |

LIST OF FIGURES

| | | |
|-----|--|----|
| 3.1 | The Constructed BodyT2 Scheduling | 27 |
| 3.2 | Scheduling the Next Packet Train | 34 |
| 3.3 | A Phone-centric BSN | 41 |
| 3.4 | The Aggregator | 41 |
| 3.5 | Performance Comparison through TelosB Mote Lab Tests: Delivered Through- put% | 42 |
| 3.6 | Performance Comparison through TelosB Mote Lab Tests: Deadline Miss Ratio | 44 |
| 3.7 | Performance Comparison through TelosB Mote Lab Tests: Energy Consump- tion Per Delivered Byte | 44 |
| 3.8 | BodyT2 Performance Through Real Body Experiments: Delivered Band- width% | 45 |
| 3.9 | BodyT2 Performance Through Real Body Experiments: Deadline Miss Ratio | 46 |

| | | |
|------|--|----|
| 3.10 | BodyT2 Performance Through Real Body Experiments: Energy Consumption Per Byte | 47 |
| 4.1 | EPOC EMotiv Neuroheadset Electrodes | 51 |
| 4.2 | Working with EMotiv Neuroheadset | 51 |
| 4.3 | Home Environment for EEG Data Collection | 52 |
| 4.4 | CDF of Connection Length | 53 |
| 4.5 | Trace of Preparing Food in Kitchen | 53 |
| 4.6 | CDF of Packet Loss Ratio | 54 |
| 4.7 | qEEG Ratio Application Fidelity | 57 |
| 4.8 | Cerebral Asymmetry Score Fidelity | 59 |
| 4.9 | Total Weighted Fidelity | 60 |
| 4.10 | Decimation Assignment Computed with Energy Minimization Algorithm . . | 73 |
| 4.11 | Fidelity Achieved with Energy Minimization Algorithm | 74 |
| 4.12 | Decimation Assignment Computed with Fidelity Maximization Algorithm . | 75 |
| 4.13 | Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 =$ $\{.2, .8\}$ | 75 |
| 4.14 | Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 =$ $\{.5, .5\}$ | 76 |
| 4.15 | Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 =$ $\{.8, .2\}$ | 77 |
| 5.1 | BuddyQoS Architecture | 83 |
| 5.2 | Interval T Divided into Two Periods | 87 |

| | | |
|-----|---|-----|
| 5.3 | Communication Paradigm for Shared BSNs | 88 |
| 5.4 | SNACK Suppression Example | 89 |
| 5.5 | Delivered Throughput | 103 |
| 5.6 | Control Overhead | 104 |
| 5.7 | Data Packet Transmission Time | 105 |
| 5.8 | Percentage of Delivered Throughput with Increasing Number of BSNs . . . | 106 |

Towards Confident Body Sensor Networking

CHAPTER 1

INTRODUCTION

A Body sensor network (BSN) consists of a group of wireless sensors, which are either wearable on or implanted into a human body to monitor vital physiological parameters and body movements. The data collected by body sensors are usually transmitted to an aggregator (e.g., a cell phone) and then is reliably delivered to a data center (e.g., a hospital) in real-time for analysis. BSNs have attracted significant interest from a wide range of applications, including smart health-care [31] [25] [21] [44], assisted living [37], emergency response [22], athletic performance evaluation [7] , and interactive controls [12] [59] [51] [33] [20]. For example, a wireless electroencephalography (EEG) headset can be included in a BSN to sample neuro signals generated from the human brain activities. Using EEG biomarkers, neurometric applications can be developed to monitor and diagnose various neurological and psychological diseases, such as coma, depression and age-related cognitive changes.

Applications in BSN place confidence requirements on the networking. Take the neurometric applications for example, almost all such applications demand high application fidelity and very low mis-diagnosis. Also, considering the EEG data transmitted from the headset to the aggregator, each neurometric application requires certain amount of data for the purpose of monitoring or diagnosis. In addition, the data should be transmitted with limited delay in many cases, such as real-time diagnosis and interactive control.

However, satisfying these confidence requirements in the BSN context is difficult. The low power sensor devices place limitation on available communication resource (250Kbps bandwidth in popular sensor nodes like TelosB), and environment noise further caused unpredictable wireless link quality. In addition, new research challenges are created due to the distinct characteristics of BSN, comparing with the conventional wireless sensor network (WSN)[13]. Firstly, the mobility of the BSN wearer adds further instability for the wireless communication in BSN. Secondly, the nodes placed on human body usually would not be deployed with high density as in the traditional WSN. Thirdly, the BSN network architecture requires less hop and may use specific node (such as a cellphone) to relay sensor data. Lastly, BSN sensor data also has different transmission rate and latency bound. These differences result in requirement for new BSN communication QoS solutions. In this work, we plan to address it through three aspects: (1)Building communication QoS, (2)Connecting communication QoS with application QoS and (3)Exposing communication QoS availability to other BSNs.

Building Communication QoS. At the communication level, the service quality is harmed by the irregular wireless communications within a BSN [49] [72], as well as the mobility of BSNs. So, we propose to build the communication QoS in the BSN. More

specifically, we provide radio-agnostic solution for joint throughput and time delay performance assurance within a BSN.

Connecting Communication QoS with Application QoS. At the application level, application fidelity requirements are different depending on the individual applications, and cannot be mapped directly to communication QoS. Taking two neurometric applications as examples, our proposed solution aims to match BSN communication with the neurometric application fidelity requirements.

Exposing Communication QoS Availability to Other BSNs. When multiple BSNs are involved, it is even harder to provide communication quality assurance with the complicated network structure where inter- and intra-BSN connections coexist. When an application requires the cooperation of multiple BSNs, we aim to provide communication QoS for inter- and intra-BSN connections.

1.1 Building Communication QoS

Many BSN applications are performance-critical, requiring stringent throughput and time delay performance assurance. For instance, in a BSN, an off-the-shelf wireless EEG device, the Emotiv EPOC Neuroheadset, can generate 4kbps EEG data streams from each of its 14 electrodes. A depression detection application requires EEG data from 2 electrodes and a Mild Cognitive Impairment (MCI) analysis requires 4 electrodes. This means, the BSN networking should provide assurance for 8kbps and 16kbps throughput requirements for the two applications, respectively. In addition, for time-critical applications, such as interactive control, responses should be made in real-time, which puts a time-delay requirement on the

data transmission.

To provide joint throughput and time delay performance assurance within BSNs, two research challenges need to be addressed: the irregular BSN link quality and the heterogeneous BSN radio platforms. In [70], the general low power wireless sensor communication is reported to be notoriously irregular. In [49] [72], the link quality in a BSN is reported to be highly dynamic and even harder to predict than in a general wireless sensor network due to interference from environment [73], body activities [37], and body fading [60]. In order to ensure the requested performance in the presence of such irregular BSN link quality, available resources must be adaptively rescheduled according to efficiency and cost. Also, existing body sensor devices, especially medical sensor devices, often use heterogeneous radio platforms, such as CC1000, ZigBee/CC2420, and Bluetooth. It is indispensable to achieve the performance assurance in a radio-agnostic manner to support platform portability.

In literature, many existing works propose specific MAC protocols or extensions to specific MAC protocols and radio platforms for providing statistical throughput and/or time delay performance assurance. Representative works are [58], [18], [5], [69], [30], [14], and [47]. Some other works, even though radio-agnostic is discussed, do not provide any performance assurance but instead provide best effort solutions for enhancing throughput and/or reducing time delay. Representative works are [39], [48], [66], [23], and [50]. Another group of works provide either throughput or time delay performance assurance, but not both. Representative works are [53], [1], [40], [26], and [72]. In [29], a solution is presented for multiple BSN data streams that can guarantee different throughput requirements but with only a single time delay bound. However, this work does not meet our goal of allowing different data streams to request both different throughputs and time delays. Moreover,

[29] is based on an individual-polling scheme, in which each data packet transmission from a sensor mote is preceded by a polling message from the central aggregator, rather than the more effective group-polling scheme, in which multiple data packet transmissions are allowed after a single polling message. Consequently, [29] is not appropriate for radio agnostic performance assurance and also introduces a minimum of 50% communication overhead.

In our proposed work, a novel and efficient radio agnostic solution is developed for heterogeneous BSNs. This solution allows different data streams to request different throughput and time delay performance assurances. For the purpose of radio-agnostic design, a group-polling scheme is used, which also reduces communication overhead.

1.2 Connecting Communication QoS with Application QoS

At the application level, the definition of QoS varies across different applications, and the relationship between communication QoS and application QoS is also unknown. In this work, the relationship between communication and application QoS is studied through two neurometric applications as examples. We use the commercial Emotiv EPOC Neuroheadset in BSN to collect EEG data for the neurometric applications. The headset is designed to communicate with a laptop as base station instead of the aggregator. As the local station does not move with the BSN, the communication pattern between EEG sensors on the headset and the base station is quite lossy due to human mobility. The current communication pattern of EEG headset is examined to exploit connection between communication and application fidelity.

Another issue to be considered here is the energy consumption. The recently developed low-cost wireless EEG sensor devices make it possible to add EEG neurometrics to in-situ physiological monitoring applications. Comparing to the conventional scalp EEG, the small wireless EEG headset or headband removes the need for cleaning the scalp area and hence requires minimum preparation for EEG collection. It also removes the electrodes wires and hence frees patients from the clinical environment, enabling ubiquitous and less invasive in-situ physiological monitoring. However, using battery instead of in-wall power supply, such EEG devices should optimize its energy usage of EEG data sampling and networking.

There are some existing works that design neurometric applications with a certain degree of fidelity. At the application level, algorithms for emotion detection and epileptic seizure detection are developed, respectively, and the application fidelity is evaluated in terms of the detection accuracy [43] [63]. Some other works [2] [61] [62] [4] trade the application accuracy with energy efficiency through algorithm improvements. The trade-off between energy consumption and data quality has also been studied in body sensor networks [6] [24] [3], but they only consider the distortion of sampled sensor readings rather than the application fidelity requirements. At a lower level, channel-aware QoS solutions have been developed for wireless sensor networks, e.g., frequency adaptation and encoding adaptation have been exploited in [71] and [68], respectively, in order to enhance the energy efficiency of communication. But they do not associate the low level QoS metrics with the application fidelity requirements. In addition to communication solutions, new hardware design, such as the chip design improvement in [67], is also proposed to improve the energy efficiency.

This work aims to improve the communication design of EEG sensing applications while meeting applications' fidelity requirements. To this end, we investigate the sampling and

networking of an off-the-shelf Emotiv EPOC Neuroheadset [17] for neurometric applications¹. In real scenarios, we measure the communication pattern between EEG sensors on the EPOC neuroheadset and a nearby base station (a laptop) to which the sensors directly communicate. A mismatch is uncovered between the current lossy communication pattern and the neurometric application fidelity requirements. Three pitfalls that contribute to this mismatch are identified in the design. First, the current EEG wireless headset design does not consider the lossy communication patterns in reality, thus the application fidelity sharply drops when the wireless link quality degrades. Second, all headset’s electrodes sample at the same speeds without considering applications’ fidelity requirements. But like an I-frame is more significant than a B- or P-frame in an MPEG-4 video stream, different EEG sensing channels may also have different impacts on the fidelity of different neurometric applications. Unlike the existence of standardized I-/B-/P-frames in the MPEG-4 standard, there are no standardized data frames and it is also difficult to standardize them for different neurometric applications. Therefore, a generic approach is needed to automatically learn the sensitivity of different neurometric applications’ fidelities to the EEG sensory data. Third, applications’ priorities are neglected. For instance, to a 29 years old human subject with tentative diagnosis of depression, the electrode readings used for depression detection is more important than those used for the detection of aging-related cognitive changes.

To address these problems, we propose a series of generic techniques, with which we are able to quantify the sensitivity of neurometric application fidelity to the EEG data. These techniques are also used to improve the energy efficiency and neurometric fidelity of EEG

¹Note that the headset uses private hardware and software design, and it does not provide direct communication control to developers. However, its manufacture encourages [17] researchers like us to give communication improvement suggestions which they will incorporate in future EEG sensing applications.

sensor sampling and networking.

1.3 Exposing Communication QoS Availability to Other BSNs

The BSN performs in-situ monitoring in people's daily life, of which group activities take a large part when people spend time with families and friends, or work with colleagues. For example, a group of athletes training and living together may all wear BSNs to monitor their athletic performance, as well as their daily schedules; or a group of senior people in a senior community may wear BSNs to monitor their health status and daily activities. In these scenarios, multiple BSNs often coexist within the communication range of each other, and their applications can utilize the shared sensor data from other buddy BSNs. Here a **buddy** BSN is the BSN worn by a family member, a friend, or a colleague, who can be trusted. Concerning the privacy issue, we assume that only non-private sensor data, such as environment temperature readings and background noise samples, can be shared with another trusted buddy. Sharing sensor data among multiple buddy BSNs has the following benefits. First, when a group of people are together, their BSNs may have multiple sensors collecting the same data, such as environment temperature or background noise, then the extra sensors can be turned off to save energy, and only one sensor is needed to broadcast its sensor data to be shared by all coexisting BSNs. Second, the applications in a BSNs can borrow the sensors from other neighboring BSNs to enhance their performance. Take the activity recognition application as an example, some activities are hard to classify with data from a single BSN, such as watching television and chatting with friends. In these cases, the sensor data from other BSNs help increase the classification accuracy.

Applications in BSNs require certain data throughput to satisfy application fidelity requirements. To provide QoS for both inter- and intra-BSN communications in multiple neighboring buddy BSNs, the major challenge is how to properly allocate the wireless resources to both inter- and intra-BSN communications in multiple BSNs. For the intra-BSN communication in a BSN, possible interference from other BSNs traffic should be taken into consideration. For the inter-BSN communication, the QoS assurance should be provided for all the BSNs sharing the same sensor. Here a BSN also needs to discover the other buddies and establish the sharing relationship.

There are existing studies providing the communication QoS in a single BSN, such as [72] [55], and between BSNs and static networks, such as [46], but the communication quality among multiple BSNs is not considered. Some other works, such as BikeNet [16], Bubble-sensing [41], and CaliBree [45], have developed new applications using the data delivered between different devices carried by different people and utilizing people rendezvous [28], but the communication qualities have not yet been studied. Researchers have also investigated the human mobility models and analyzed the inter-contact time of different individuals [56] [34] [35] [64] [15]. However, these works only aim to improve network connectivity and performance by predicting when the devices on different people can communicate with each other, but assume perfect communication, rather than address the inter- and intra-network interference by providing the communication QoS.

1.4 Contributions

The main contribution of this work can be summarized as follows:

- *Building Communication QoS.* We jointly consider throughput and time delay performance assurance for heterogeneous body networks that demand radio-agnostic solutions.

We theoretically prove that joint throughput and time delay assurance with a group-polling scheme is NP-hard, while throughput assurance only is P.

We propose BodyT2, a practical solution for joint throughput and time delay performance assurance in heterogeneous body networks. We also develop a phone-centric body network to demonstrate BodyT2's effectiveness.

Through both TelosB mote lab tests as well as on body deployments in an Android phone-centric personal wearable system, our performance evaluation demonstrates that BodyT2 greatly outperforms existing solutions.

- *Connecting Communication QoS with Application QoS.* We uncover and analyze a mismatch between the lossy EEG sensor communication pattern and the high neurometric application fidelity requirements.

We propose generic techniques that can automatically learn the sensitivity of application fidelity to sampled sensor readings, especially for neurometric applications.

With the learned sensitivity, we propose an energy minimization algorithm that allows us to minimize the energy usage in EEG sampling and networking with given application fidelity requirements.

With the learned sensitivity, we also propose a fidelity maximization algorithm that allows us to maximize the sum of all applications' fidelities with a given data buffer on a wireless EEG headset.

- *Exposing communication QoS availability to other BSNs.* We propose a framework called BuddyQoS, which enables the neighboring buddy BSNs to discover each other and share sensors between them.

We enables the upper layer applications to request the same or different throughput assurance for data streams from either the local BSN or the neighboring buddy BSNs. With these requests, BuddyQoS provides statistical throughput assurance for the communications in each BSN.

In particular, we expose the communication QoS availability across different neighboring buddy BSNs. Here exposing the communication QoS availability means, when a BSN shares its local nodes with its neighboring buddies, they collaborate to provide the required throughput assurance for the shared data streams. For the shared nodes, the resources needed to deliver the data to all listening BSNs are estimated in their local BSNs. Based on the estimation, the data streams are allocated with sufficient resource, and the resource allocation is broadcast to all listening BSNs.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows. First, Chapter 2 presents the related works. Chapter 3 aims to build communication QoS, and proposes a novel approach to provide joint throughput and time delay assurance in a radio-agnostic. Chapter 4 farther connects communication QoS with application QoS. Taking two neurometric application as example, we propose to automatically learn the data sensitivity to application fidelities, based on which EEG sampling and networking in the BSN can be improved. Chapter 5

studies the connections among multiple BSNs, and expose communication QoS to each other. Finally, Chapter 6 concludes the dissertation.

CHAPTER 2

RELATED WORKS

In this chapter, we present the related works to our study. Section 2.1 includes designs with communication quality considerations in wireless networks. Section 2.2 includes works concerning physiological and psychological application fidelity. Section 2.3 includes studies on rendezvous applications, when sensors on two people should be connected for data transmission.

2.1 Building Communication QoS

On the communication level, a number of works have been proposed to provide throughput and/or time delay performance assurance for wireless ad hoc networks and sensor networks. However, they are usually designed as specific MAC protocols or extensions, rather than

being radio-agnostic. Based on a hardware-assisted time synchronization radio platform (called FireFly), both a MAC [58] and a real time operation system [18] are developed to provide high throughput and bounded time delay. A real-time voice stream-capability [42] is also built with the Firefly nodes. [5] provides differentiated service to guarantee communication delay and loss based on the Distributed Coordination Function (DCF) specific to IEEE 802.11. [69] maintains average delay assurance to real-time multimedia applications in wireless ad hoc networks that use IEEE 802.11. [30] devises, in compliance with EDCA-incorporated UWB MAC, a framework that provides deterministic throughput for real-time traffic. [14] explores IEEE 802.15.4 radio's capability of supporting discrete event control applications with short time delay requirement. [47] proposes a MAC for delivering low-latency event driven alarm information in wireless sensor networks with minimum overall delay.

Several radio-agnostic solutions also exist in literature. But rather than providing throughput and/or time delay performance assurance, they usually schedule available resources to optimize or improve specific performance metrics, like improving throughput and reducing time delay. [39] and [48] propose resource management approaches to enhance network throughput. [66] schedules resources to minimize the average delay in bandwidth-sharing networks with a linear topology. [23] develops a sensor network operating system that incorporates preemptive priority scheduling to achieve short time delay. [50] derives the average delay bound and maximized throughput for one-hop wireless networks.

Several other radio-agnostic designs exist for assuring throughput or time delay performance, but not both. [53] considers a multicast network where only time delay bounds are assured. [1] analyzes the real-time capacity of multihop wireless networks and pro-

poses a fixed-priority scheduling to guarantee time delay. [40] presents a velocity-monotonic scheduling that accounts for time delay and distance in large-scale networks. [26] guarantees end-to-end delay for three types of communication services in wireless sensor networks with geographic location information. [72] guarantees different throughput requirements from applications, but time delay requirements are ignored.

We are also aware that [29] presents a radio-agnostic solution for both throughput and time delay performance assurance. However, all data streams in the same network are subject to a single time delay bound, unable to request different time delay assurances as what we propose in this work. Moreover, this design introduces 50% communication overhead, since it is based on the individual-polling rather than group-polling scheme.

2.2 Connecting Communication QoS with Application QoS

Several existing works propose neurometric applications with fidelity consideration. The emotion detection application developed in [43] aims to find the features that are robust to EEG signal noise and have strong discriminative capacity. In [63], an onset epileptic seizure detection algorithm is designed. It uses machine learning to extract spectral, spatial, and temporal features from sampled EEG signals to achieve high accuracy and short delay. However, to meet the application fidelity requirements, these works only consider improving the detection and classification algorithms, but do not take lossy wireless communication into account. Moreover, at the presence of data loss, how to maximize application fidelity is not addressed.

At the application level, efforts have also been paid to trade accuracy with energy

efficiency. In [2], a synchronization likelihood channel selection method is developed to reduce the number of EEG data streams used in emotion assessment, with only a slight loss of classification performance. In [61], a screening detector is developed to help multi-feature detection algorithms reduce energy consumption by processing much fewer features. In [62], a machine learning technique is used to construct an epilepsy detector with fewer channels. In [4], lossless sensor data compression is proposed to reduce the communication energy usage at the expense of the increased computation complexity. However, different from our work, none of these solutions propose to automatically learn the sensitivity of application fidelity to sensory data for reducing energy consumption.

The trade-off between energy and data quality has been studied in body sensor networks. Barth et. al. [6] evaluate the energy usage with respect to data distortion. Hanson et. al. [24] explore energy-fidelity scalability to adjust compression ratios while still maintaining data quality. Au et. al. [3] provide real-time energy profiling and management for achieving desired sensing resolutions. However, in these works, fidelity is quantified with the metrics like “mean square error” and “data resolution”. These metrics only measure how much the raw sensor data is distorted but cannot quantify the distortion of the neurometric application fidelity, in terms of diagnosing accuracy or false alarm that we focus on.

At a lower level, channel-aware QoS solutions have been developed for enhancing the system performance. MMSN [71] provides frequency adaption to maximize parallel transmission among neighboring nodes for energy efficiency. ACR [68] proposes adaptive encoding scheme to enhancing collision recovery and transmission efficiency. However, these solutions do not associate the lower level QoS with application fidelity.

At the hardware level, some solutions like [67] aim to improve the energy efficiency in

EEG acquisition, digitization, and feature extraction on the chip. However, it is not a communication solution like ours.

2.3 Exposing Communication QoS Availability to Other BSNs

In the BSN area, there are works studying the communication QoS in a single BSN. BodyQoS [72] provides statistical throughput assurance, and BodyT2 [55] provides joint throughput and time delay assurance for communications in a single BSN with heterogeneous radios in sensors. MCMAC [46] develops a TDMA-based MAC protocol to support communication between BSNs and static networks. However, these works do not consider the communication qualities among multiple BSNs.

In other areas, there are works enhancing mobile applications based on the data exchanged between different mobile devices on different individuals, utilizing people rendezvous [28]. BikeNet [16] is built for the community of cyclists, where cycling-related data are collected to quantify cyclist performance and the cyclist environment. Bubble-sensing [41] proposes to distribute sensing tasks among multiple sensors on different people, and relies on the rendezvous to deliver the required data back to the task initiator. CaliBree [45] proposes to increase system accuracy with distributed self-calibration, depending on the opportunistic rendezvous to determine their relative miscalibration. However, these works do not take communication quality issues into consideration. CaliBree [45] proposes a light weighted solution considering the limited sensing contact time, but there is no communication quality assurance during the rendezvous.

Also, in the context of opportunistic routing, human mobility is well studied for the

purpose of enhancing connectivity and performance of the opportunistic routing. Rhee et. al. propose truncated Levy walk (TLW) [56] and Lee et. al. propose Self-similar Least Action Walk (SLAW) model [34] to produce synthetic walk traces with all human mobility features for simulation and theoretical analysis. In [35], the authors analytically derive the delay-capacity tradeoffs for Levy mobility model. Srinivasa et. al. propose CREST [64] to estimate the time remaining for the next rendezvous using conditional residual time. Eisenman et. al. [15] perform numerical analysis on sensor sharing and substitution strategies for enhancing data fidelity and delay. These works assume negligible time for data delivery, and connections between sensors on two human body only involve two nodes. Their main focus is usually on inter-contact time to predict when the data can be delivered to the next way-point. These works assume that once the sensor nodes are close to each other, the data are forwarded successfully without addressing the communication quality issue that we focus on in this work.

CHAPTER 3

BUILDING COMMUNICATION QOS

Many BSN applications require stringent performance assurance in terms of communication throughput and bounded time delay, we propose BodyT2, which provide this joint assurance in a novel radio-agnostic manner. In our approach, the underlying MAC and PHY layers can be heterogeneous and their details do not need to be known to upper layers like the resource management. Such a radio-agnostic performance assurance is critical because a range of radio platforms are adopted for practical body sensor usage. Our approach is based on a group-polling scheme that is essential for radio-agnostic BSN design. Through theoretical analysis, we prove that with the group-polling scheme, achieving joint throughput and time delay assurance is an NP-hard problem. For practical system deployment, we propose the BodyT2 framework that assures throughput and time delay performance in a heterogeneous BSN. Through both TelosB mote lab tests and real body experiments in an Android phone-

centric BSN, we demonstrate that BodyT2 achieves superior performance over existing solutions.

3.1 Problem Definition and Analysis

In this section, we theoretically analyze BSN resource scheduling in order to meet requested performance assurance. We first explain the asymmetric BSN architecture and compare two BSN scheduling schemes: group-polling and individual-polling. Then, based on the more effective group-polling scheme, we prove that scheduling for the throughput performance assurance is a P problem, while the joint throughput and time delay assurance is NP-hard.

3.1.1 Group-Polling v.s. Individual-Polling

An asymmetric architecture is desired for BSNs in which a comparatively more powerful aggregator polls less powerful sensor nodes for data communication [72]. Two scheduling schemes have been proposed based on this asymmetric BSN architecture. In the *individual-polling* [29] scheme, each data packet transmission from a node is preceded by a polling packet from the aggregator that specifies which node is polled. Since this scheme adds in a minimum of 50% communication overhead, it is not appropriate for practical radio-agnostic system deployment. A more effective and energy efficient *group-polling* scheme is introduced in [72], in which multiple data packet transmissions are allowed from a node following a single polling packet from the aggregator. The series of packets sent after a polling packet, which can be more than one packet, is called a *packet train*. Group-polling is strongly preferred over individual-polling mainly for the following two reasons:

- *Efficiency.* Compared with individual-polling, group-polling requires much fewer polling packets to deliver the same amount of data packets, greatly saving communication bandwidth ($\leq 250\text{Kbps}$ in popular sensor motes like TelosB) and energy (sensor motes are usually powered by AA batteries). The saved communication bandwidth can be used to serve more data streams in a BSN, enhancing the BSN capacity. By listening to more sparsely transmitted polling messages, sensor motes have more sleeping time and hence the system lifetime is extended.

- *Catering to Radio-Agnostic BSN Designs.* Since heterogeneous radio platforms are widely adopted in the commercial market, radio-agnostic performance assurance is needed in BSNs. Group-polling better caters to this demand than individual-polling since it operates on a virtual MAC (VMAC) abstraction [72]. For throughput performance assurance, VMAC abstracts common MAC behaviors with time-domain parameters: T_{minPkt} and T_{maxPkt} . These are respectively the lower and upper bound of the time that the underlying MAC uses for handling a packet transmission request. When the channel is clear, the radio control is returned to VMAC within T_{minPkt} ; when suffering interference, the underlying MAC may return the radio control within T_{maxPkt} and report giving up after exceeding the maximum number of backoffs and/or retransmissions. During runtime, VMAC also measures the average MAC response time T_k for each mote k in a BSN, which reflects the average communication cost of a specific mote for a single data packet communication. So, $T_k \in [T_{minPkt}, T_{maxPkt}]$.

Without knowledge of the underlying MAC implementation, the aggregator using individual-polling has to reserve the maximum time T_{maxPkt} for a single data packet transmission. In most cases, the data packet can be successfully transmitted with time much less than

T_{maxPkt} , so the rest of the reserved time is wasted. However, with group-polling the aggregator can efficiently estimate the time needed to transmit a packet train as $T_k \times NumofPkt$. Even though the underlying MAC is only allowed to send a data packet when the remaining reserved time is no less than T_{maxPkt} (otherwise, we risk losing control of the underlying radio), this packet's real transmission time T_k^* is usually much less than T_{maxpkt} . The difference ($T_{maxPkt} - T_k^*$) can be salvaged and merged to the time reserved for sending the next packet. In this way, fluctuation of the transmission time is absorbed and tolerated.

3.1.2 Throughput Assurance

In BodyQoS [72], throughput performance assurance is provided with the group-polling scheme. Each data stream i specifies its throughput requirement b_i and the scheduling algorithm determines the resource, specifically the time resource, for the data stream.

Definition 3.1 (BodyQoS Scheduling problem) *Suppose group-polling is used in a BSN. Given a fixed-length time interval $T_{interval}$ and N data streams in the BSN with throughput requirements $\{b_i\}$, the problem is to decide the time schedule for each data stream, such that in $T_{interval}$ the delivered throughput is no less than the requested throughput.*

In order to solve this problem, BodyQoS first computes the required bandwidth for each data stream when the channel is clear, which is called the ideal bandwidth. Also, the time to send one packet is T_{minPkt} when there is no interference, and the number of data packets to be delivered within $T_{interval}$ is $\lceil \frac{b_i \times T_{interval}}{8 \times S_{pkt}} \rceil$, where S_{pkt} is the effective payload size of a single data packet in bytes. Then at run time, the effective bandwidth is measured. With the ratio of the ideal bandwidth to the moving average result of the effective bandwidth,

BodyQoS dynamically recomputes the average packet sending time and the number of data packets, the product of them is the time needed for delivering stream i 's data packets. The time for sending one polling message is estimated as T_{maxPkt} , and BodyQoS adopts a constant number (1 is default) of polling messages within $T_{interval}$ for each data stream, which is configured as a system-wide parameter.

Admission decisions are made based on the total required throughput of all QoS streams, and the scheduling algorithm computes the time schedule for each stream. Since it needs constant time complexity to compute the time of both data communication and polling for individual data streams, computing the required time schedule for all motes in the network is a P problem. In summary, with only the throughput requirement in the group-polling scheme, the BodyQoS scheduling problem is solvable in polynomial time.

3.1.3 Joint Assurance of Throughput and Time Delay

For time delay performance assurance, $d_{k,i}$ is introduced to denote the requested time delay bound for data stream i on sensor mote k . The complete performance assurance requirement is denoted as $(b_{k,i}, d_{k,i}, p_{k,i})$ where $b_{k,i}$ specifies the throughput requirement and $p_{k,i}$ denotes the priority. Instead of scheduling polling messages for individual data stream as in [72], here the aggregator aggregates polling messages for all data streams on the same mote. To put it another way, the aggregator does not specify how much time each stream on a mote uses but only allocates enough time to satisfy the total throughput requirement of all streams on the same mote. Thus, a packet train sent from a mote can contain data packets from different data streams.

Now, the scheduling problem is more complicated with the added time delay requirement

since it needs to ensure that individual data packets are delivered within $d_{k,i}$. This is equivalent to ensuring that the gap between any two consecutively scheduled packet trains for mote k is bounded by $d_{k,i}$ minus the time of transmitting one polling packet and one data packet. So, if data arrive just after the end of a packet train, the data can be timely transmitted in the next packet train. When multiple data streams are on the same mote k , the aggregator considers the minimum delay requirement $\min_i \{d_{k,i}\}$. For convenience of presentation, we introduce two intermediate symbols:

- $B_k = \frac{\sum_i b_{k,i}}{S_{pkt}}$ is the number of packets required to be sent for all streams on mote k in a unit time.

- $G_k = \min_i \{d_{k,i}\} - T_{maxPkt} - T_k$ is the maximum gap allowed between consecutive packet trains of mote k .

The packet train schedule can be represented as $\{(st_{k,j}, et_{k,j})\}$, where $st_{k,j}$ is the start time for the aggregator to send the polling message of the packet train j of mote k and $et_{k,j}$ is the latest time a data packet from this packet train is allowed to be received at the aggregator. The BodyQoS Scheduling Problem in Def. 3.1 can be extended to the following BodyT2 Scheduling problem.

Definition 3.2 (BodyT2 Scheduling problem Π) *Suppose group-polling is used in a BSN. Given N motes in the BSN with performance requirements (B_k, G_k) , the problem is to decide the time schedule $\{(st_{k,j}, et_{k,j})\}$ such that for all $k \in [1, N]$, $j \in \mathbb{N}$, the following constraints are satisfied:*

- *Length Constraint.* $\forall k, j, et_{k,j} - st_{k,j} = T_{maxPkt} + T_k \times \lceil (et_{k,j} - et_{k,j-1}) \times B_k \rceil$. It ensures that the allocated time is enough to transmit both the data and polling packets for

all streams on mote k based on the throughput requirements.

- *Gap Constraint.* $\forall k, j, st_{k,j} - et_{k,j-1} \leq G_k$. It ensures that the gap between any two consecutively allocated packet trains of mote k is bounded by the minimum time delay requirement of all streams on mote k .

- *Disjoint Constraint:* $\forall k_1, k_2, j_1, j_2$, if $k_1 \neq k_2$ or $j_1 \neq j_2$, then $st_{k_1, j_1} \neq st_{k_2, j_2}$; if $st_{k_1, j_1} < st_{k_2, j_2}$, then $et_{k_1, j_1} \leq st_{k_2, j_2}$. It ensures that time periods allocated to different packet trains do not overlap, i.e., no internal interference.

Lemma 3.1 *The BodyT2 Scheduling problem Π is NP-hard.*

With the following three steps, we demonstrate that a known NP-complete problem, the Partition problem (Π'), is polynomially reducible to our BodyT2 Scheduling problem Π . Let π' and π refer to any instances of problems Π' and Π , respectively. We construct a polynomial reduction f that converts any instance π' of the Partition problem to some instance $\pi = f(\pi')$ of our BodyT2 Scheduling problem such that π' has a solution if and only if $\pi = f(\pi')$ has a solution.

Step 1: Construct the polynomial reduction f from π' to π .

Definition 3.3 (Partition problem Π') *Given a finite set A of numbers, is there $A' \subseteq$*

$$A, \text{ such that } \sum_{a_k \in A'} a_k = \sum_{a_{k'} \in A - A'} a_{k'} ?$$

For any partition problem instance π' with set $A = \{a_1, \dots, a_n\}$ of n integers, we choose a constant c such that $c \times a_k \geq 2$ for all $k \in [1, n]$. We construct the following instance $\pi = f(\pi')$ of the BodyT2 Scheduling problem with $n+1$ motes. We let $T_k = T_{maxPkt} = 1, \forall k$,

and let $T = c \times (\sum_{a_k \in A} a_k + 2 \times a_{n+1})$, where $a_{n+1} \geq 2/c$ (so $c \times a_{n+1} \geq 2$). We define (B_k, G_k)

as:

$$(B_k, G_k) = \begin{cases} (\frac{c \times a_k - 1}{T}, T - c \times a_k), & k \in [1, n] \\ (\frac{c \times a_{n+1} - 1}{T/2}, T/2 - c \times a_{n+1}), & k = n + 1 \end{cases}$$

This reduction can clearly be done in polynomial time.

Step 2: Prove that if π' has a solution, then $f(\pi')$ has a solution.

For any partition problem π' , assume there is a solution such that $A' = \{a_{k'_1}, \dots, a_{k'_{n_2}}\}$, $A - A' = \{a_{k_1}, \dots, a_{k_{n_1}}\}$, $n_1 + n_2 = n$, and $\sum_{a \in A'} a = \sum_{a \in A - A'} a = T/2c - a_{n+1}$. We have the following scheduling $f(\pi')$ which repeats with a cycle of length T .

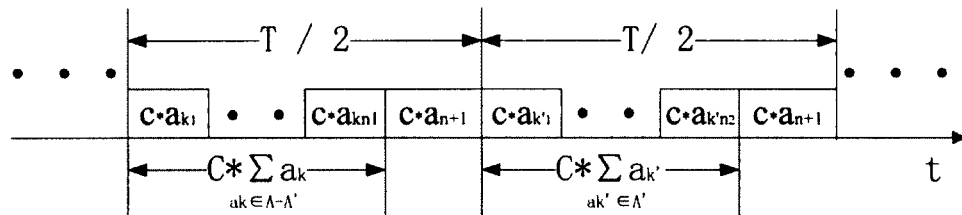


Figure 3.1: The Constructed BodyT2 Scheduling

As shown in Figure 3.1, packet trains of mote k_1, \dots, k_{n_1} are scheduled in the first half of T and packet trains of mote k'_1, \dots, k'_{n_2} are scheduled in the second half. We then have:

for mote $k = k_1, \dots, k_{n_1}$,

$$st_{k_1,1} = 0, \quad st_{k_2,1} = et_{k_1,1}, \quad \dots$$

$$st_{k,j} = st_{k,j-1} + T, j > 1$$

$$et_{k,j} = st_{k,j} + c \times a_k,$$

for mote $k = k'_1, \dots, k'_{n_2}$,

$$\begin{aligned}
st_{k'_1,1} &= T/2, & st_{k'_2,1} &= et_{k'_1,1}, \quad \dots \\
st_{k,j} &= st_{k,j-1} + T, \quad j > 1 \\
et_{k,j} &= st_{k,j} + c \times a_{k'},
\end{aligned}$$

for mote $n + 1$,

$$\begin{aligned}
st_{n+1,j} &= \begin{cases} T/2 - c \times a_{n+1}, & j = 1 \\ st_{n+1,j-1} + T/2, & j > 1 \end{cases} \\
et_{n+1,j} &= st_{n+1,j} + c \times a_{n+1}.
\end{aligned}$$

Now, we check whether the three constraints in Def. 3.2 are satisfied. First, we check for mote $k \in [1, n]$. Since the right side of the Length Constraint equals $1 + T \times B_k = 1 + T \times \frac{c \times a_k - 1}{T} = c \times a_k$ and the left side of it equals $et_{k,j} - st_{k,j} = c \times a_k$, the Length Constraint is satisfied. The Gap Constraint also stands as $st_{k,j} - et_{k,j-1} = T - c \times a_k = G_k$. In any interval T , $st_{k_1,j} < \dots < st_{k_n,j}$, $et_{k_1,j} = st_{k_1,j} + c \times a_{k_1} = st_{k_1,1} + (j-1) \times T + c \times a_{k_1} = st_{k_2,1} + (j-1) \times T = st_{k_2,j}$, so the packet trains of mote $k \in [k_1, k_n]$ do not overlap, i.e., the Disjoint Constraint stands. In a similar way, we can also prove that the Disjoint Constraint stands for mote $k \in [k'_1, k'_{n_2}]$.

Second, we check for mote $n + 1$. The Length Constraint is satisfied as its right side equals to $1 + T \times B_{n+1} = 1 + \frac{T}{2} \times \frac{c \times a_{n+1} - 1}{T/2} = c \times a_{n+1} = et_{n+1,j} - st_{n+1,j}$, which equals to its left side. Since $st_{n+1,j} - et_{n+1,j-1} = T/2 - c \times a_{n+1} = G_{n+1}$, the Gap Constraint also holds. In the same period, $et_{k_{n_1},j} = st_{k_1,j} + \sum_{k=k_1, \dots, k_{n_1}} (et_{k,j} - st_{k,j}) = st_{k_1,1} + (j-1) \times T + c \times \sum_{a_k \in A-A} a_k = (j-1) \times T + T/2 - c \times a_{n+1} = st_{n+1,2j-1}$. In a similar way, $et_{k'_{n_2},j} = st_{n+1,2j}$. So, the packet trains of mote $n + 1$ do not overlap with those of other motes and the Disjoint Constraint stands. Therefore, the schedule in Figure 3.1 is feasible.

Step 3: Prove that if $f(\pi')$ has a solution, then the corresponding π' has a solution.

Assume that $f(\pi')$ has a schedule $\{(st_{k,j}, et_{k,j})\}$ that satisfies the three constraints in Def. 3.2. We need to construct a solution for the corresponding π' .

First, in the schedule $\{(st_{k,j}, et_{k,j})\}$, we can prove that there must exist a period T that satisfies:

- $\forall k \in [1, n], \exists$ exactly one j , such that $(st_{k,j}, et_{k,j}) \subseteq T$ (abusing the denotation T a little bit) and

$$\begin{cases} et_{k,j} - st_{k,j} = c \times a_k; \\ st_{k,j} - et_{k,j-1} = G_k; \end{cases}$$

- For mote $n+1, \exists$ exactly one j , such that $(st_{n+1,j}, et_{n+1,j}) \subseteq T, (st_{n+1,j+1}, et_{n+1,j+1}) \subseteq T$ and

$$\begin{cases} et_{n+1,j} - st_{n+1,j} = et_{n+1,j+1} - st_{n+1,j+1} = c \times a_{n+1}; \\ st_{n+1,j} - et_{n+1,j-1} = st_{n+1,j+1} - et_{n+1,j} = G_{n+1}; \end{cases}$$

- $T = (et_{n+1,j-1}, et_{n+1,j+1})$. This can be proven by contradiction.

Second, we construct a subset of motes $\{k'_1, \dots, k'_{n_2}\}$ such that during time period T ,

$$(st_{k,j}, et_{k,j}) \subseteq \begin{cases} (et_{n+1,j}, st_{n+1,j+1}), & k \in \{k'_1, \dots, k'_{n_2}\} \\ (et_{n+1,j-1}, st_{n+1,j}), & k \in \{k_1, \dots, k_{n_1}\} \end{cases}$$

$$\text{where } \{k_1, \dots, k_{n_1}\} = \{1, \dots, n\} - \{k'_1, \dots, k'_{n_2}\}$$

With the Disjoint Constraint, we can derive

$$\begin{aligned} \sum_{k \in \{k'_1, \dots, k'_{n_2}\}} (et_{k,j} - st_{k,j}) &= c \times \sum_{k \in \{k'_1, \dots, k'_{n_2}\}} a_k \leq G_{n+1} \\ \sum_{k \in \{k_1, \dots, k_{n_1}\}} (et_{k,j} - st_{k,j}) &= c \times \sum_{k \in \{k_1, \dots, k_{n_1}\}} a_k \leq G_{n+1} \end{aligned}$$

Since $\sum_{k \in [1, n]} (et_{k,j} - st_{k,j}) = 2 \times G_{n+1}$, we have $c \times \sum_{k \in \{k_1, \dots, k_{n_1}\}} a_k = c \times \sum_{k \in \{k'_1, \dots, k'_{n_2}\}} a_{k'}$. So, the partition problem π' has a solution $A' = \{a_{k'_1}, \dots, a_{k'_{n_2}}\}$.

Therefore, with steps 1~3, we prove Lemma. 3.1, i.e., our BodyT2 Scheduling problem is NP-hard.

3.2 BodyT2 Design

Since the BodyT2 Scheduling problem for joint throughput and time delay assurance is NP-hard, it is nontrivial to obtain the optimal solution. In this section, we propose an empirical solution for practical system deployment. We present the necessary/sufficient conditions for admission control and also the algorithms for admission control and time resource scheduling. We also extend the existing VMAC [72] for enforcing the time resource scheduling result to meet the time delay performance requirements in addition to the throughput performance requirements.

3.2.1 Admission Control

The admission controller examines the performance assurance requests $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $k \in [1, n]$ and makes ACCEPT/REJECT decisions. In time period T , the admission controller computes the total required time for satisfying all streams' requests when interference is captured and reflected by T_k . This includes both data and polling packets. The total number of data packets mote k needs to transmit is $D(k, T) = \lceil B_k \times T \rceil$ (B_k as defined in Section 3.1.3). The total number of polling packets for mote k , defined as $P(k, T)$, equals the number of packet trains scheduled for that mote. In BodyQoS [72] which only provides throughput assurance, $P(k, T)$ is simply fixed as 1 for each T , but when the time delay

assurance is jointly considered it is more difficult to determine. The total required time for both data and polling packets can be computed as $D(k, T) \times T_k + P(k, T) \times T_{maxPkt}$ which needs to be no more than the total available time T .

The Necessary and Sufficient Admission Conditions If mote k is scheduled to send $P(k, T)$ packet trains during T , the sum of gaps between its packet trains plus the time for sending the $P(k, T)$ polling packets is $T - D(k, T) \times T_k$. Also, the gap between any two consecutive packet trains of mote k should be bounded by G_k (defined in Section 3.1.3). So, $T - D(k, T) \times T_k \leq P(k, T) \times (G_k + T_{maxPkt})$. When the gap decreases, the number of packet trains increases. Since G_k is the maximum gap allowed, the minimum number of packet trains is:

$$P_{min}(k, T) = \frac{T - D(k, T) \times T_k}{G_k + T_{maxPkt}}. \quad (3.1)$$

So, the minimum required time for sending data and polling packets for mote k is:

$$S_{min}(k, T) = D(k, T) \times T_k + P_{min}(k, T) \times T_{maxPkt}. \quad (3.2)$$

Therefore, the necessary condition of admission control is:

$$\sum_k S_{min}(k, 1) \leq 1. \quad (3.3)$$

To derive a sufficient admission condition, assume a round-robin schedule in which all motes within T receive the same number of polling messages from the aggregator. The number of polling messages is estimated as the maximum value of $P_{min}(k, T)$ for all k . In this way, a sufficient condition for admission control can be derived as:

$$\sum_k (D(k, 1) \times T_k + \max_k \{P_{min}(k, 1)\} \times T_{maxPkt}) \leq 1. \quad (3.4)$$

Algorithm 1 Admission Control

Input: performance requests $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$ for data stream $i \in N$ on mote $k \in [1..n]$, the average packet transmission time $\{T_k\}$ for mote k

Output: ACCEPT or REJECT decision

repeat

if the necessary condition in Inequ. (3.3) is broken **then**

 REJECT and remove the request with the lowest $p_{k,i}$ from $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$; continue;

end if

if the sufficient condition in Inequ. (3.4) stands **then**

 return ACCEPT;

end if

$t_c = 0$; \forall remaining k , let $et_{k,j-1} = 0$ and $R_k = 0$;

loop

 call Algorithm 2 with input $(\{(b_{k,i}, d_{k,i}, p_{k,i})\}, t_c, \{et_{k,j-1}\}, \{R_k\})$ and get output $(\{st_{k,j}, et_{k,j}\}$ or FAILURE);

if Algorithm 2 returns FAILURE **then**

 REJECT and remove the request with the lowest $p_{k,i}$ from $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$; break;

else

$t_c = et_{k,j}$; $et_{k,j-1} = et_{k,j}$;

end if

if at least one packet train is allocated to each mote **then**

 return ACCEPT;

end if

end loop

until $\{(b_{k,i}, d_{k,i}, p_{k,i})\} = \emptyset$

 return ACCEPT;

The Admission Control Algorithm With the necessary and sufficient conditions, the admission controller can make preliminary decisions: if the necessary condition fails, a REJECT decision is made; if the sufficient condition holds, an ACCEPT decision is made; otherwise, if the sufficient condition fails but the necessary condition holds, it is hard to tell whether an appropriate schedule can be obtained for the requested data streams. As we have proven in Section 3.1.3, this is actually an NP-hard problem. Therefore, we integrate an

empirical solution into our admission control Algorithm 1. With the help of Algorithm 2 (to be explained later), Algorithm 1 tries to make an appropriate schedule, i.e., determining the start and end time of packet trains for all motes to meet the joint throughput and time delay constraints. If a schedule is found, an ACCEPT decision is made; otherwise, a REJECT decision is made. When a REJECT decision is made, the data stream with the lowest priority is removed and the admission controller tries to make ACCEPT/REJECT decisions again with the remaining data streams. This process repeats until either an ACCEPT decision is made or all data streams are finally rejected and removed. The later case happens when interference is so strong that no packets can be timely delivered.

Algorithm for Scheduling the Next Packet Train Algorithm 2 presents details of scheduling the next packet train. It is used in both the admission control Algorithm 1 and the time resource scheduling Algorithm 3 that we will discuss later. In Algorithm 2, we introduce R_k to denote the number of expected but unsent packets from mote k . So, by the end of a packet train $et_{k,j}$, even though the aggregator expects to receive $D(k, et_{k,j} - et_{k,j-1})$ ($D(k, t)$ as defined in Section 3.2.1) packets from mote k based on the throughput requirement, it may actually receive $D(k, et_{k,j} - et_{k,j-1}) - R_k$ packets. A negative R_k value means that the aggregator receives more packets than expected from mote k , so it allocates less time for mote k 's next packet train. When sensor data sampling and packet arrival are uniformly distributed, R_k provides flexibility to time resource scheduling. Since R_k is measured and can only have a nonzero value at runtime, R_k is set to zero in admission control. Jointly considering R_k and Def. 3.1. Length Constraint, we have:

$$et_{k,j} - st_{k,j} = (D(k, et_{k,j} - et_{k,j-1}) + R_k) \times T_k + T_{maxPkt}. \quad (3.5)$$

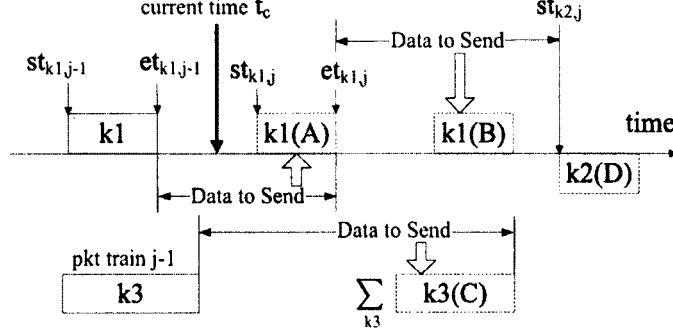


Figure 3.2: Scheduling the Next Packet Train

Suppose the most recently scheduled packet train, say packet train $j - 1$ for mote k , has the schedule of $(st_{k,j-1}, et_{k,j-1})$, then the latest start time of mote k 's next packet train j should be $et_{k,j-1} + G_k$. In this algorithm, we try to schedule the next packet train j for the mote that has the minimum $et_{k,j-1} + G_k$ value, say mote k_1 , which is similar to the earliest deadline first policy. An empirical rule we use here is: we give mote k_1 's packet train j a schedule if and only if we can foresee that any other mote, say k_2 as in Algorithm 2 and Figure 3.2, can also have its packet train j scheduled.

As shown in Figure 3.2, k_1 is the mote that has the earliest start time $st_{k,j} = et_{k,j-1} + G_k$. k_2 is another arbitrary mote that has a later start time $st_{k_2,j}$. k_3 is another arbitrary mote with its start time $st_{k_3,j}$ in between those of k_1 and k_2 . Suppose k_3 's most recent packet train schedule is $(st_{k_3,j-1}, et_{k_3,j-1})$. Then, during $(t_c, st_{k_2,j}]$, k_3 desires to send at least one packet train (C in Figure 3.2). The total time that all such k_3 motes require is $\sum_{k_3} S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$ which can be computed according to Equation (3.2). Also, during $(et_{k_1,j}, st_{k_2,j}]$, mote k_1 requires time $S_{min}(k_1, st_{k_2,j} - et_{k_1,j})$ to send packet train B. The time between packet trains A and D should be long enough to schedule packet trains B and C, that is,

$$\sum_{k_3} (S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1}) + R_{k_3} \times T_{k_3}) + S_{min}(k_1, st_{k_2,j} - et_{k_1,j}) + R_{k_1} \times T_{k_1} \leq st_{k_2,j-1} - et_{k_1,j}. \quad (3.6)$$

Here, $st_{k_2,j} = et_{k_2,j-1} + G_k$ which is the latest possible start time of mote k_2 's next packet train j .

With Inequ. (3.7), we make sure that there is enough room to schedule packet train A. Also, with Inequ. (3.8), we make sure that the distance between packet train A and mote k_1 's previous packet train $j-1$ is bounded by G_{k_1} .

$$(D(k_1, et_{k_1,j} - et_{k_1,j-1}) + R_{k_1}) \times T_{k_1} + T_{maxPkt} \leq et_{k_1,j} - t_c \quad (3.7)$$

$$et_{k_1,j} - et_{k_1,j-1} - (D(k_1, et_{k_1,j} - et_{k_1,j-1}) + R_{k_1}) \times T_{k_1} - T_{maxPkt} \leq G_{k_1} \quad (3.8)$$

Finally, $et_{k_1,j}$ is computed as the largest value that satisfies Inequ. (3.6)~(3.8) and $st_{k_1,j}$ is computed with Equation (3.5).

3.2.2 Time Resource Scheduling

In time resource scheduling, the aggregator sequentially computes the time allocated to each packet train. More specifically, the time resource scheduling Algorithm 3 calls Algorithm 2 to compute a schedule $(st_{k,j}, et_{k,j})$ for the next packet train as well as a schedule $(st_{k',j}, et_{k',j})$ for the packet train after the next. BodyT2 communication supports two kinds of data: the QoS data that requires throughput and time delay assurance, and the best effort data that does not. If enough time ($\geq 2 \times T_{maxPkt}$) is available before starting the next packet train,

Algorithm 2 Scheduling the Next Packet Train

Input: performance requirements $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, the current time t_c , the end time of the most recently scheduled packet trains for all motes $\{et_{k,j-1}\}$, $\{R_k\}$

Output: the next packet train schedule $(st_{k,j}, et_{k,j})$ or (FAILURE)

$\forall k$, compute the G_k value based on its definition in Section 3.1.3

get $\min_k \{et_{k,j-1} + G_k\}$ and assume it is $et_{k_1,j-1} + G_{k_1}$

for any k_2 ($k_2 \neq k_1$) **do**

 /*Check if the period $(st_{k_1,j}, st_{k_2,j}]$ is long enough for packet trains of all other motes (say k_3 as an arbitrary one)*/

for any k_3 ($st_{k_1,j} \leq st_{k_3,j} \leq st_{k_2,j}$, $k_3 \neq k_1$, $k_3 \neq k_2$) **do**

 With Equation (3.2), estimate $S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$ which is the time that mote k_3 needs in $(t_c, st_{k_2,j}]$

end for

 compute $\sum_{k_3} S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$

 estimate the largest $et_{k_1,j}$ that satisfies Inequ. (3.6), (3.7), and (3.8)

if \exists such $et_{k_1,j}$ **then**

 return (FAILURE)

end if

end for

$et_{k_1,j} =$ the minimum $et_{k_1,j}$ value computed above for all k_3

compute $st_{k_1,j}$ with Equation (3.5)

return $(st_{k_1,j}, et_{k_1,j})$

VMAC is called to poll for best effort data. Then, when time proceeds to $st_{k,j}$, VMAC is called to poll mote k to enforce schedule $(st_{k,j}, et_{k,j})$. The time resource scheduling waits while mote k transmits QoS data packets. The execution of current schedule ends when either an early termination of this packet train is received from mote k due to lack of data or the time proceeds to $st_{k',j}$. After that, parameters T_k and R_k are updated to assist scheduling the next packet train while the process repeats.

Algorithm 3 Time Resource Scheduling

Input: performance requirements $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $\{R_k\}$

Output: function calls to VMAC

$\forall k, et_{k,j-1} = 0; R_k = 0$

loop

call Algorithm 2 with input $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $t_c =$ the current time, $\{et_{k,j-1}\}$, $\{R_k\}$ and get output $((st_{k,j}, et_{k,j})$ or FAILURE)

if Algorithm 2 returns FAILURE **then**

/* this only happens when the interference level largely increases after the admission control*/

execute the admission control Algorithm 1 again to remove low priority streams;continue;

end if

$et_{k,j-1} = et_{k,j}; R_k = 0;$

if $st_{k,j} \geq$ the current time + $2 \times T_{maxPkt}$ **then**

call VMAC to poll for best effort data

end if

wait until the time proceeds to $st_{k,j}$;

call Algorithm 2 with input $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $t_c = et_{k,j}$, $\{et_{k,j-1}\}$, $\{R_k\}$ and get output $((st_{k',j}, et_{k',j})$ or FAILURE);

if Algorithm 2 returns FAILURE **then**

for the same reason above, execute the admission control Algorithm 1 again to remove low priority streams;continue;

end if

call VMAC to poll mote k for QoS data;

wait until the time proceeds to $st_{k',j}$ or mote k terminates the packet train early; then, update the values of $et_{k,j}$, T_k , and R_k with runtime measurements and let $et_{k,j-1} =$

$et_{k,j}$;

end loop

3.2.3 Enforcing Time Schedule on VMAC

VMAC is located on both the aggregator and motes for enforcing the time resource scheduling result computed by Algorithm 3. We extend the existing VMAC [72] to enforce the newly added time delay requirement in addition to the throughput requirement. The extended VMAC not only checks the remaining allocated time but also the specified time delay constraint for each packet transmission. It also notifies the aggregator to terminate the packet train if there is no packet to send.

On the aggregator, VMAC receives calls from the above scheduler and calls the underlying real MAC functions. For a packet train schedule $(st_{k,j}, et_{k,j})$, VMAC sends a polling message to mote k with the allocated time length $PL_{k,j} = et_{k,j} - st_{k,j} - T_{maxPkt} + (st_{k',j} - et_{k,j})$. Here, $st_{k',j} - et_{k,j}$ is the gap between mote k 's packet train j and mote k' 's packet train j . Since this gap immediately follows the scheduled time period $et_{k,j} - st_{k,j} - T_{maxPkt}$ and is also not scheduled to any other packet train, it is allocated to extend the length of packet train j for mote k . When VMAC is called to poll for best effort data before a packet train schedule $(st_{k,j}, et_{k,j})$, it broadcasts a message, indicating that the following time period $(st_{k,j} - \text{current time} - T_{maxPkt})$ is open for all motes' best effort communication. During this period, potential collision resolution among different motes' transmissions is handled by the underlying specific MAC protocols.

When a mote, say mote k , receives a polling message, VMAC enforces the time resource scheduling result by feeding QoS or best effort data to the aggregator within the allocated time periods. When polled for QoS packets with length $PL_{k,j}$ (computed in the previous paragraph), VMAC on mote k computes the amount of data that each stream i on mote k

requests to send since the end of mote k 's previous packet train. Then, VMAC organizes the data into a packet train in which the packets with earlier deadlines, including those for retransmissions, are put ahead of those with later deadlines. Before sending each data packet, VMAC conducts the following checks:

- If the remaining allocated time is less than T_{maxPkt} , VMAC does not send the data packet and the packet train terminates. This ensures that the control of the underlying radio is returned to the upper layers before the allocated time expires. Again, it is worthy to repeat that in most cases it takes less time than T_{maxPkt} to deliver this data packet. However, VMAC is able to salvage the unused time of this data packet to send the next data packet.

- If the deadline of the data packet is earlier than the current time plus T_k , it is immediately dropped since we may otherwise waste time on a packet that finally misses its deadline.

- If the current data packet is the only QoS data packet remaining in the mote, VMAC sets the *NoMoreData* bit in the replied packet's header which informs the aggregator of the early termination of the packet train.

3.3 Performance Evaluation

BodyT2 is implemented in TinyOS 2.x with NesC, and evaluated through both TelosB mote lab tests and real body experiments in an Android phone-centric BSN. BodyT2 is compared with the state-of-the-art BodyQoS [72] as well as the default best effort solution in the standard TinyOS 2.x release. Three performance metrics are used: (i) the percentage

of delivered throughput, i.e., the timely delivered data throughput over the requested data throughput; (ii) the data packet deadline miss ratio, which is computed as the number of data packets that miss their deadlines divided by the number of data packets requested to be sent from motes; and (iii) the average energy consumed to timely deliver one application data byte to the aggregator. Detailed evaluation settings are given below:

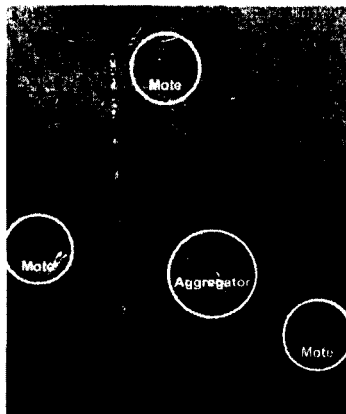
TelosB mote lab tests. A data stream with performance requirement (5kbps throughput, 200ms time delay) is admitted into BodyT2 to report data from source to the aggregator in the lab experiments. Besides the existing interference from the lab environment like WiFi and Zigbee [73], a TelosB node is also introduced to generate explicit interference (see Tab. 3.1).

Real body experiments in an Android phone-centric BSN. We also develop an Android phone-centric BSN to demonstrate the effectiveness and efficiency of BodyT2 and present the prototype BSN in Figure 3.3. The aggregator of the BSN is zoomed to Figure 3.4 in which one TelosB is plugged in the USB hub to directly communicate with the Android phone. Multiple sensor motes can also be plugged in the USB hub and operate on different frequencies for improving the aggregator throughput. Additional sensor motes can be attached on the human body and wirelessly communicates to the aggregator. Our main technical contributions for developing such a BSN lies in four aspects: Android OS kernel support, hardware support, TinyoS support, and application support. Due to space limitations, more technical details are not presented here but available in our technical report [52].

In our real body experiments, TelosB devices are attached to a human body as shown in Figure 3.3: a TelosB is attached to the left chest that generates a data stream with the per-

Table 3.1: Interference Settings

| Interference Level | Interference Strength | Interference Period |
|--------------------|--|---------------------|
| Level 0 | Lab background noise | 0s~120s |
| Level 1 | Lab background noise + 1 noise packet every 30ms | 120s~180s |
| Level 2 | Lab background noise + 1 noise packet every 25ms | 180s~240s |
| Level 3 | Lab background noise + 1 noise packet every 20ms | 240s~300s |

**Figure 3.3:** A Phone-centric BSN**Figure 3.4:** The Aggregator

formance requirement (4kbps throughput, 500ms time delay) and requests BodyT2 service; a TelosB is attached to the left wrist that generates a data stream with the performance requirement (2kbps throughput, 1000ms time delay bound) and also requests BodyT2 service; a TelosB mote is attached slightly above the right hip that generates a data stream with the performance requirement (4kbps bandwidth, 500ms time delay) but requests best effort service; the same aggregator as shown in Figure 3.4 is put inside the bottom left pocket of the jacket for data collection and analysis.

All experiments described above are repeated multiple times and similar results are observed. In the following subsections, we present two groups of representative results which demonstrate that BodyT2 largely outperforms the existing BodyQoS and best effort

solutions.

3.3.1 Performance Results of TelosB Mote Lab Tests

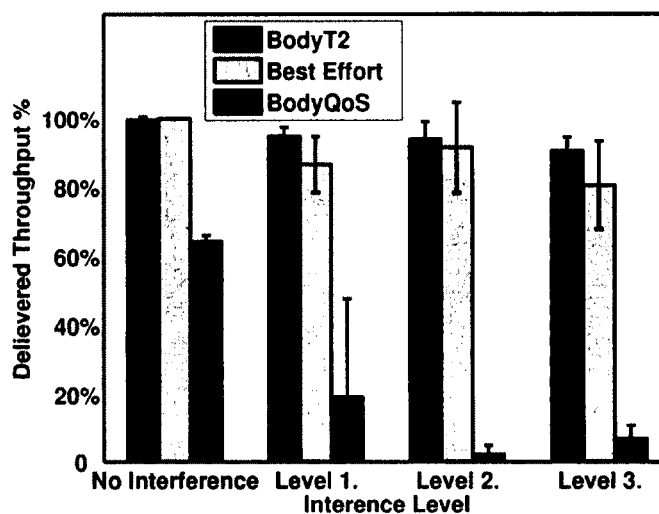


Figure 3.5: Performance Comparison through TelosB Mote Lab Tests: Delivered Throughput%

Figure 3.5 plots the mean and standard deviation of the percentage of timely delivered throughput when different interference levels are present in the lab experiment. We first observe that BodyT2 achieves a higher timely delivered throughput ratio than those of best effort and BodyQoS. In fact, BodyT2 achieves up to 10% higher throughput ratio than best effort and 91% higher throughput ratio than BodyQoS. Second, we observe that BodyT2 achieves a more stable throughput delivery ratio than those of best effort and BodyQoS. As shown in the figure, the largest standard deviation for BodyT2 is 5.2% under interference level 2, while best effort has the largest standard deviation of 13.2% under interference level 2 and BodyQoS has the largest standard deviation 28.7% under interference level 1. Third, we observe that the performance gain of BodyT2 over best effort and BodyQoS

increases when interference increases. For instance, throughout the 4 interference periods, BodyT2 has a less obvious decrease of the throughput delivery ratio than those of best effort and BodyQoS. BodyT2 achieves superior performance than existing approaches because its design addresses the joint throughput and time delay requirements, while the existing approaches do not. We are also aware that BodyQoS performs much better than best effort in [72] when only the throughput requirement is considered, but it performs worse than best effort when the time delay requirement is jointly considered here. This is because BodyQoS is not designed to address the time delay requirement and hence data packets can be held too long to be timely delivered. Due to uncertainty of the lab background noise, the interference intensity may fluctuate with time. So, packets that were scheduled to be sent out but actually unsent in the previous time period, when the interference is comparatively strong, may be able to be sent out in the current time period, when the interference is comparatively weak, to fulfill the throughput requirement. This is why sometimes the percentage of delivered throughput exceeds 100%.

Figure 3.6 presents the data packet deadline miss ratio. First, we see that BodyT2 achieves an extremely low deadline miss ratio ($< 5\%$) under all 4 interference levels, while best effort has 17.9% packets missing deadlines under interference level 3 and BodyQoS misses all deadlines under interference level 2. Second, we see that the deadline miss ratios for best effort and BodyQoS largely increase when interference increases. For example, best effort's deadline miss ratio raises 11% from interference level 2 to 3. Meanwhile, BodyT2's deadline miss ratio remains almost constantly low. For similar reasons, BodyQoS performs the worst among the three. BodyQoS misses all deadlines under interference level 2 but has nonzero throughput delivery ratio under interference level 2, because data packets not

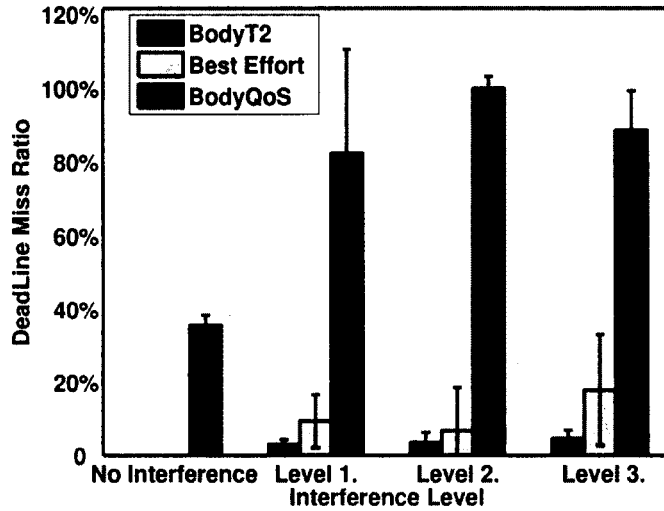


Figure 3.6: Performance Comparison through TelosB Mote Lab Tests: Deadline Miss Ratio

delivered in the previous time period, i.e., under interference level 1, are sent out here.

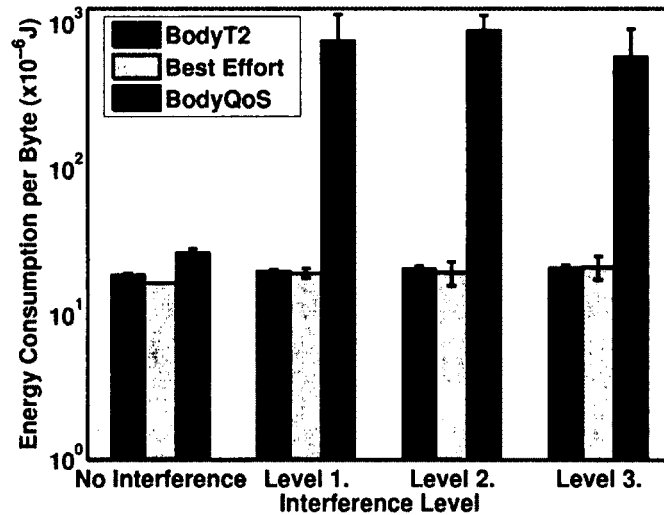


Figure 3.7: Performance Comparison through TelosB Mote Lab Tests: Energy Consumption Per Delivered Byte

Figure 3.7 shows the energy consumption per timely delivered application data byte, measured in Joules (J). As the number of timely delivered data byte for BodyQoS drops

to zero, we may have division by zero. So, we assign a very large energy consumption value $1 \times 10^{-3} J$ in such cases. Since the y-axis value for BodyQoS is much larger than that of BodyT2 and best effort, we plot the y-axis with a log scale. From Figure 3.7, we observe that BodyT2 uses similar energy as that of best effort. We also observe that when interference increases BodyT2's energy consumption per timely delivered data byte remains stable, but best effort's energy consumption per timely delivered data byte fluctuates and becomes less stable. This is because fewer data bytes are timely delivered in best effort than BodyT2 when interference increases even though best effort dose not waste more energy retransmitting packets that finally miss deadlines.

3.3.2 Performance Results of Real Body Experiments in an Android Phone-centric BSN

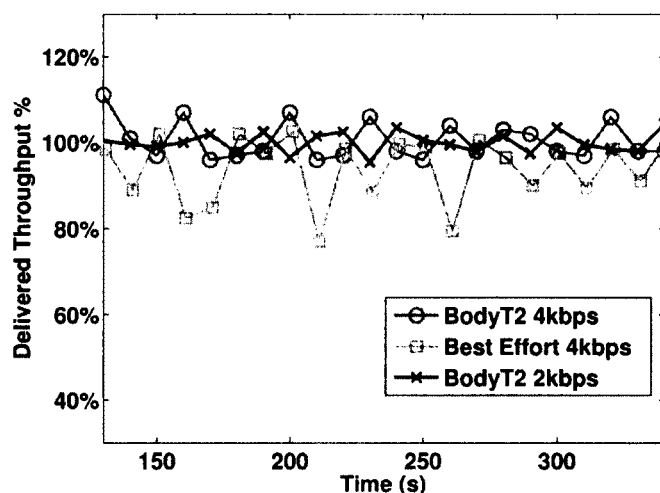


Figure 3.8: BodyT2 Performance Through Real Body Experiments: Delivered Bandwidth%

Figure 3.8 plots the the timely throughput delivery ratio. We observe that both BodyT2

data streams on average maintain $\sim 100\%$ timely throughput delivery ratio. However, the best effort data stream on average has $< 100\%$ timely throughput delivery ratio which also fluctuates significantly. For example, the best effort stream achieves only 77% ratio at 210s and 79.5% ratio at 260s, but BodyT2 data streams' ratios never go below 95.5%. This demonstrates BodyT2's effectiveness and best effort's ineffectiveness in supporting multiple data streams' throughput and time delay performance requirements. Here, for the same reason as we have presented when explaining Figure 3.5, we also observe that the percentage of delivered throughput fluctuates above and below the 100% line.

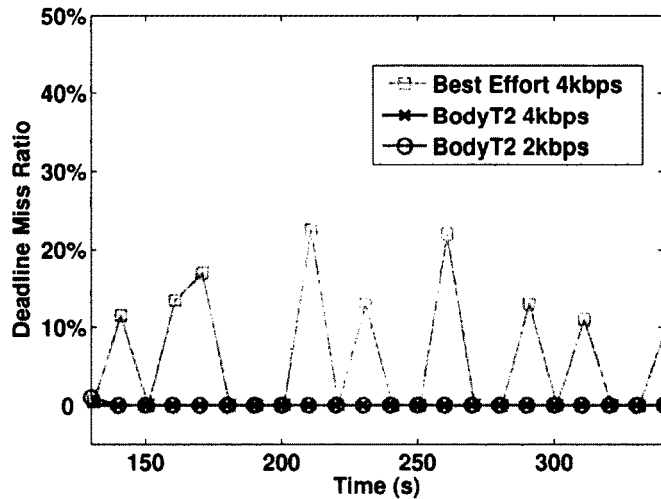


Figure 3.9: BodyT2 Performance Through Real Body Experiments: Deadline Miss Ratio

Figure 3.9 depicts the data packet deadline miss ratio. We observe a near zero deadline miss ratio for both BodyT2 data streams but up to 22% deadline miss ratio for the best effort data stream. This is because on the one hand, best effort uses the resources remaining after QoS resource scheduling, and on the other hand, the best effort approach does not consider deadline when scheduling resources.

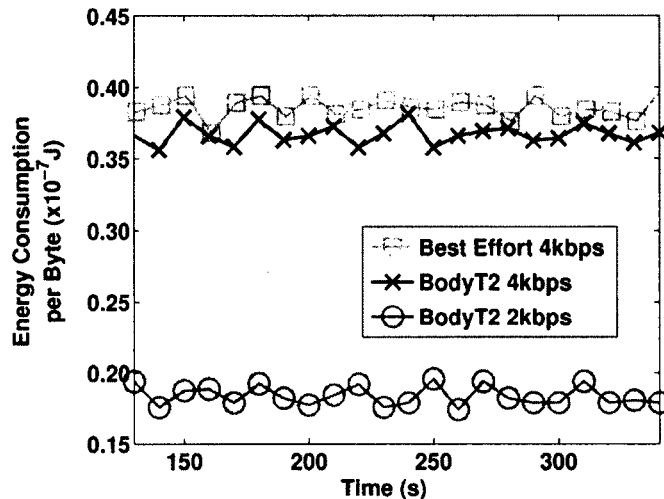


Figure 3.10: BodyT2 Performance Through Real Body Experiments: Energy Consumption Per Byte

Figure 3.10 shows the energy consumption per timely delivered application data byte. We observe that while both the BodyT2 data streams and the best effort data stream have similar energy efficiency on average, the energy efficiency fluctuation of the best effort data stream is much higher than that of the BodyT2 data streams. The maximum energy consumption per timely delivered data byte on the two BodyT2 data streams are $1.88 \times 10^{-5} J$ and $1.92 \times 10^{-5} J$, respectively. But the maximum value of the best effort data stream is $2.24 \times 10^{-5} J$, which is 17% ~ 20% higher than that of BodyT2. This is because the group-polling scheme and also adaptive resource scheduling in BodyT2 can absorb and tolerant fluctuations of link qualities but best effort can not.

3.4 Conclusions

Joint throughput and time delay performance assurance is critical for many BSN applications. We proposes a novel approach to provide this joint assurance in a radio-agnostic

manner. Our approach is based on a group-polling scheme that is essential for radio-agnostic BSN design. We rigorously prove that with the group-polling scheme resource scheduling for the throughput performance assurance is P, while the joint throughput and time delay assurance is NP-hard. For practical system deployment, we propose the BodyT2 framework that assures throughput and time delay performance in a heterogeneous BSN. Through both TelosB mote lab tests and real body experiments in an Android phone-centric BSN, we demonstrate that BodyT2 achieves superior performance over existing solutions.

CHAPTER 4

CONNECTING COMMUNICATION QOS WITH APPLICATION QOS

As the application QoS varies depending on different applications, we especially focus on one type of applications in the BSN: the neurometric applications using electroencephalography (EEG) sensing. There has been great interest in exploring the applications of EEG to monitor and diagnose human mental health problems. In comparison with other BSN applications, a high volume of data traffic is usually collected through wireless in current EEG sensing applications. With a newly developed wireless neuroheadset that requires minimum preparation for data collection, we can incorporate EEG neurometrics into in-situ and ubiquitous physiological monitoring. As EEG headset is a resource constraint

system while providing critical health services, the design of EEG sensor sampling and networking must take both high application fidelity and energy efficiency into account. However, through our empirical study in realistic environments with an off-the-shelf Emotive EPOC Neuroheadset, we uncover a mismatch between lossy EEG sensor communication and high neurometric application fidelity requirements.

To tackle this problem, we first propose a learning technique that automatically learns the sensitivity of neurometric application fidelity to EEG data. The learned sensitivity is also used to develop an energy minimization algorithm and a fidelity maximization algorithm. The energy minimization algorithm minimizes the energy usage in EEG sampling and networking while meeting applications' fidelity requirements. The fidelity maximization algorithm maximizes the sum of all applications' fidelities through the incorporation and optimal utilization of a limited data buffer. The effectiveness of our proposed solutions is validated through trace-driven experiments.

4.1 Motivation

We use a newly developed commercial wireless EEG headset, the Emotiv EPOC Neuroheadset, to collect EEG signals. Whereas the device is not specially designed for the usage of health care, it is the state-of-the-art high resolution wireless neuro-signal acquisition device we can get from the market. As shown in Figure 4.1, the neural signals are sampled with 14 scalp sensors, and transmitted using a custom wireless chipset operating in the 2.4GHz band. A proprietary wireless connection is used to deliver the sampled data to a local base station plugged with an USB dongle, as shown in Figure 4.2. The headset's communication

module does not open its control to developers. From our observation, no packet storage or retransmission scheme exists in its default design.

In our experiments, we first study the real communication pattern of the headset. We observe that although the packet loss rate is low when the headset is connected to the base station, it can be easily disconnected. Then, we define the fidelities of two applications using two common neurometric indices, the Tsallis Entropy (qEEG) Ratio [8] and the Cerebral Asymmetry Score [27]. Finally, based on our experiment results, we uncover a mismatch between the current wireless communication pattern and the applications' fidelity requirements.

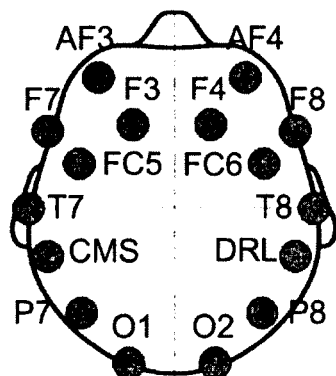


Figure 4.1: EPOC EMotiv Neuroheadset Electrodes



Figure 4.2: Working with EMotiv Neuroheadset

4.1.1 Communication Pattern

We collect 27 communication traces between the EEG headset and the local station in both working and home environment as shown in Figure 4.3. To study the influence of different environments on the wireless communication, we collect traces in different locations around offices and hallways in an academic building, and between the living room and the kitchen

in an residential apartment. The communication traces are collected as the subject wearing the headset performs various normal daily activities, like walking, working and cooking.

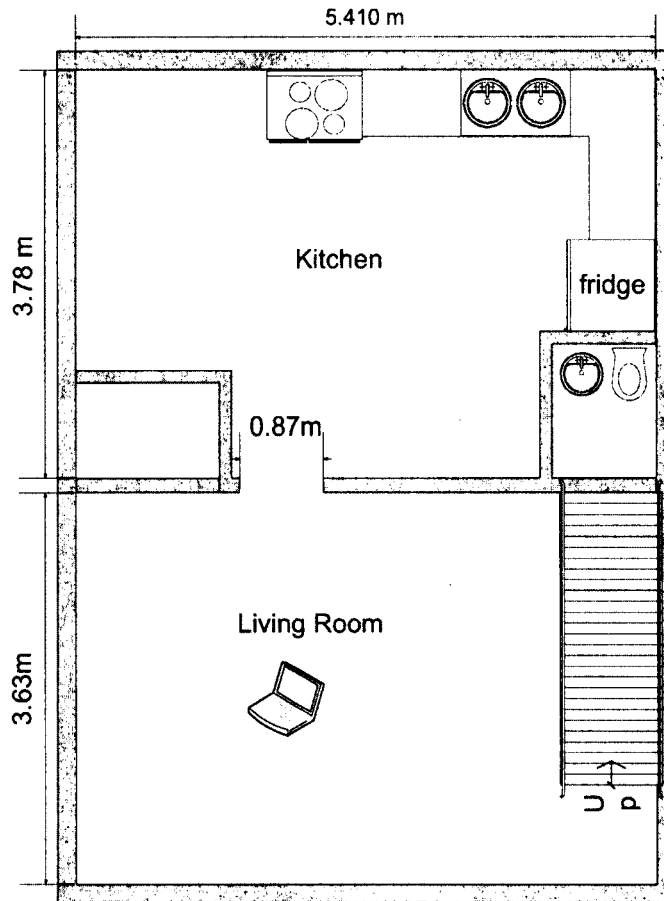


Figure 4.3: Home Environment for EEG Data Collection

The communication between the EEG headset and the local station can be impacted by distance, human mobility and environmental factors. The wireless connection is generally reliable within the same room, but could be easily disconnected when the headset is more than 4 ~ 5 meters away from the local station. No packet can be delivered during a disconnection period, until the headset is moved back to the local station and reconnects with it again. Even with connection, some packets could be dropped if the headset is not

very close to the local station, especially when there are environmental obstacles in between, or people walking around causing interference.

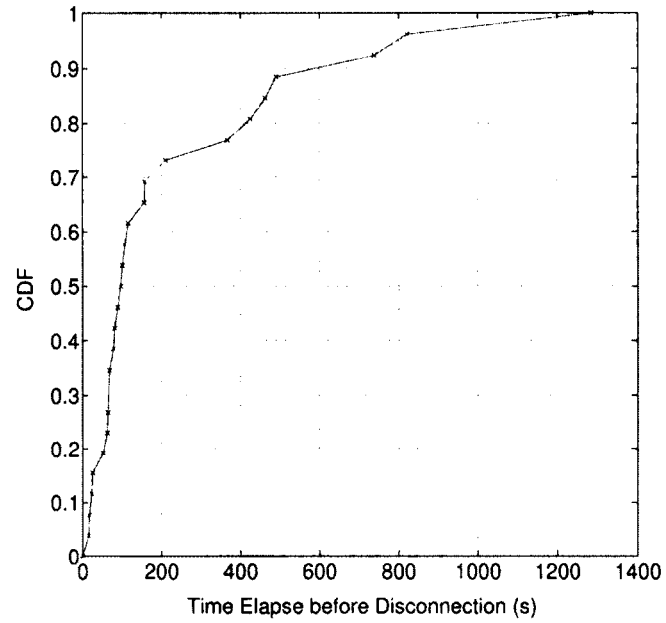


Figure 4.4: CDF of Connection Length

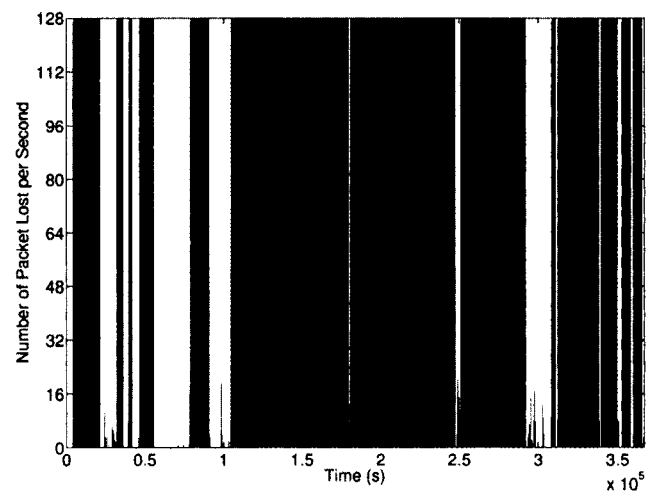


Figure 4.5: Trace of Preparing Food in Kitchen

Easily disconnected. We observe that the communication between the EEG headset and the local base station is impacted by distance, human mobility and environmental factors. The EEG headset's wireless communication range is very limited, and the headset is easily disconnected beyond 4 ~ 5 meters. The trace plotted in Figure 4.5 is collected in home environment (see Figure 4.3 for the apartment floor print) when the headset wearer was preparing food in the kitchen and moving around the house from time to time, with the local base station set in the middle of the living room. The Y axis is the number of lost packets in each second. When the headset is disconnected, all 128 packets (the full sampling rate is 128 packets per second) are lost in each second during disconnection. In this trace, the headset is disconnected for about 2/3 of the time. The time elapsed before disconnection is short, ranging from 16 seconds to 21 minutes, with its cumulative distribution function (CDF) plotted in Figure 4.4.

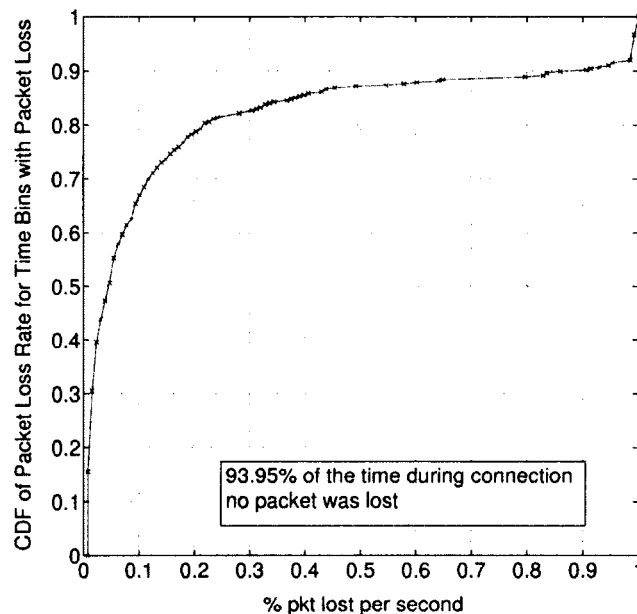


Figure 4.6: CDF of Packet Loss Ratio

Low packet loss rate with connection. When the wireless communication is not disconnected, we divide the collected EEG data into bins of one second. Figure 4.6 plots the CDF of the packet loss rate for the bins that contain packet losses, which only account for 6.05% of all used bins. From Figure 4.6, we can see that about 80% of these bins have less than 20% packet loss. Considering that 93.95% of the bins do not even have any packet loss, only 1.18% of the total transmitted packets are lost. In other words, once the headset is connected, the packet loss rate is very low.

4.1.2 EEG-based Neurometric Application and Fidelity Definition

With two widely used neurometric indices, we define the fidelity of specific neurometric applications: the Tsallis Entropy ratio used for distinguishing mild dementia subjects from normal aging subjects [8], and the cerebral asymmetry score used for identifying depressed subjects [27].

4.1.2.1 Mild Cognitive Impairment (MCI) Detection using Tsallis Entropy (qEEG) Ratio

The Tsallis entropy (qEEG) ratio, R_{po} , is the ratio of measured qEEG values between prefrontal cortex and occipital lobe. We compute the prefrontal qEEG with EEG readings from electrodes AF3 and AF4, and the occipital qEEG with EEG readings from O1 and O2. The electrodes positions are shown in Figure 4.1. The readings are averaged for all the electrodes from the same regions, and then divided into an epoch of 30 seconds. The critical points, which are the local maxima and minima of the averaged readings, further divide the epoch into multiple intervals. Considering the variance of the EEG signal in both slow and

rapid manners, the qEEG equals to one minus the ratio of the sum of all the rapid variances divided by the slow variance. The slow variance is the measure of signal variance over each epoch, but the rapid variance is measured for each interval, as show in Equation 4.1.

$$var_{Slow} \equiv \sum_{i=1}^N (x_i - \bar{x})^2 \quad (4.1)$$

$$var_{Rapid} \equiv \sum_{x_i \in Interval_j} (x_i - \bar{x}_j)^2 \quad (4.2)$$

$$qEEG = 1 - \frac{\sum_{Interval_j} var_{Rapid}}{var_{Slow}} \quad (4.3)$$

$$R_{po} = \frac{\text{prefrontal } qEEG}{\text{occipital } qEEG} \quad (4.4)$$

A subject can be classified as a MCI patient or normal people based on its R_{po} values. According to [8], the R_{po} metric demonstrates statistically significant differences between MCI patients and normal people. A mean R_{po} of 1.65 is measured for MCI patients, but 1.21 for normal people, and $\mu_0 = 1.2$ is proposed as the classification threshold. Based on statistical hypothesis test technique, we use two tailed T-test to decide if the mean of the subject's R_{po} is below or above the threshold μ_0 . The statistic to be tested is $t = \frac{\bar{x} - \mu_0}{s/\sqrt{n-1}}$, where \bar{x} is the mean of measured R_{po} values, s is the standard derivation, and n is the number of measured R_{po} values. With the t value, a p -value can be found using a table of values from Student's t-distribution. If the calculated p -value is above the critical value $t_{\alpha/2}(n-1)$, which can be found in the statistical table with given statistical significance α , then the subject is classified as MCI patient; otherwise, the subject is classified as normal.

Now we define and quantify the application fidelity of MCI detection. We define the application fidelity as the MCI detection accuracy. To quantify the application fidelity, we use the result of classification with full information, i.e., EEG signals sampled with

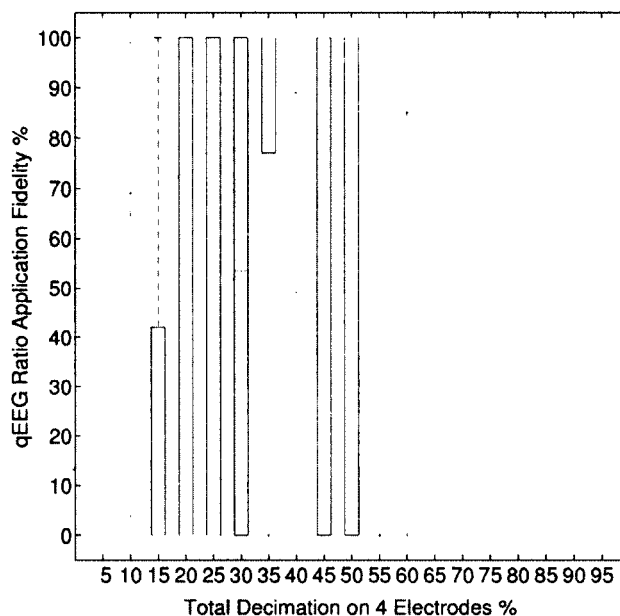


Figure 4.7: qEEG Ratio Application Fidelity

the full rate, as the ground truth. Then, using partial information, i.e., EEG signals with reduced sampling rate, we perform the MCI classification again. Here we use the term **Data Decimation** to refer to retaining a lower rate than the full rate for data sampling and transmission. The data decimation rate is computed as the ratio of the new sampling rate after decimation compared with the full rate. For example, 10% data decimation means that the data sampling and transmission rate is reduced to 10% of the full rate. With the full-sampling-rate traces we collected, the data decimation is performed by randomly keeping a portion of the EEG readings and dropping the rest. Finally, we calculate the percentage of the classification results that are identical to the ground truth. This percentage is defined to be the application fidelity of MCI detection.

Figure 4.7 presents the fidelity of MCI detection. Since for a given total data decimation

rate, different combinations of data decimation rates (which we call data decimation assignments) can be applied to the same four electrodes, we randomly choose 30 of the possible data decimation assignments and plot their fidelity results with box plot in Figure 4.7. This box plot shows the median, 25th and 75th percentiles, upper and lower adjacent values, and outliers.

4.1.2.2 Depression Detection using Cerebral Asymmetry Score

The cerebral asymmetry score quantifies the difference of the α band powers between left and right cerebral hemisphere. Readings from two symmetrical electrodes, F3 and F4, are used to compute the asymmetry score. The series of sampled EEG data is divided into chunks of 2.05s, with 75% overlapping [27]. A Fast Fourier Transform [32] is applied to each chunk, and the power density (measured in $\mu V^2/Hz$) is computed as the sum of the activities across all bins within the α band divided by the number of bins. Here the α band is one of the five EEG bands: δ , 1 – 4Hz; θ , 4 – 8Hz; α , 8 – 13Hz; β , 13 – 20Hz; and electromyogram (EMG), 70 – 80Hz. All power density values are log-transformed to normalize their distribution. The asymmetry score equals the log of the right hemisphere α band power minus the log of the left hemisphere α band power (Log right α band power - Log left α band power in $\mu V^2/Hz$).

The asymmetry score can be used to decide if a subject is in depression, as its value is very different between a depressed patient and normal people. According to [27], the mean values of the asymmetry scores are about -0.2 and 0.2 for depressed and normal people, respectively. The asymmetry scores of normal people are hardly below -0.2 , but the asymmetry scores of depressed patients are hardly above 0.2 . Thus, based on the

measured asymmetry scores, we use T-test for classification. If the test shows that the subject's mean asymmetry score is above 0.2 with given significant level α , the subject is classified as normal; if the mean is below -0.2 , the subject is classified as depressed; if the mean is between -0.2 and 0.2 , the subject is classified as undetermined.

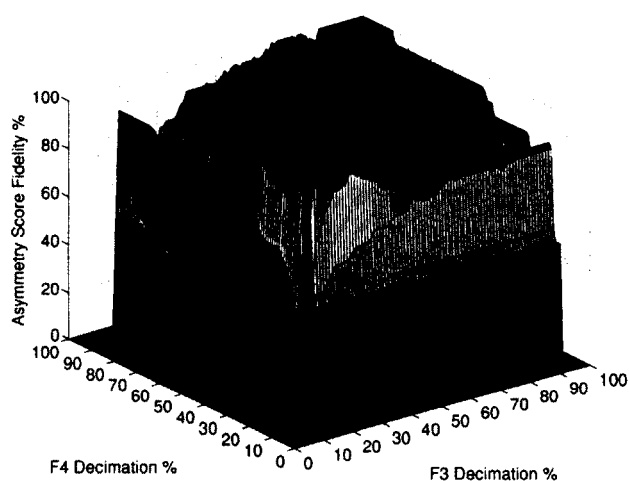


Figure 4.8: Cerebral Asymmetry Score Fidelity

Similar to the MCI detection, the fidelity of the depression detection application is defined as the percentage of identical event detection results between when the full-sampling-rate data is used and when decimated data is used. Figure 4.8 plots the fidelity of asymmetry score when different rates of decimation are applied.

Assuming both applications are equally important, each with 0.5 priority weight, Figure 4.9 plots the total weighted fidelity with data decimation rates on all 6 electrodes, using box plot.

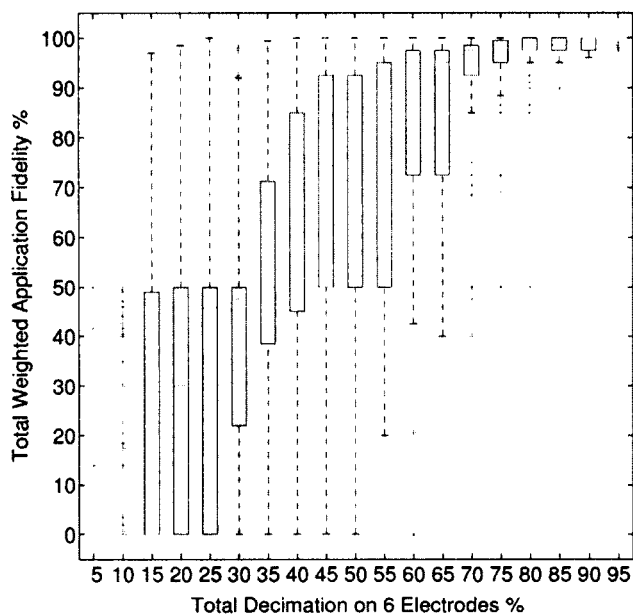


Figure 4.9: Total Weighted Fidelity

4.1.3 Mismatch between Communication Pattern and Application Fidelity

From the communication pattern analysis in Section 4.1.1, we can see that the wireless EEG headset is either well connected or disconnected with the local base station most of the time. So, we discuss the application fidelity under these two scenarios, respectively.

Well connected. We find that the current design uses a higher sampling rate than necessary. As we apply data decimation to the sampled data, the result shows that it is still possible to maintain high fidelity for the two example applications. For example, in Figure 4.8, the fidelity of depression detection can still be as high as 94% when both F3 and F4 electrodes' readings are applied with 15% data decimation rate. This implies that the application does not need the full sampling rate of 128Hz for all the electrodes on the neuro-

headset. From Figures 4.7, 4.8 and 4.9, we can see that the application fidelity is generally monotonous with the data decimation rate, but can be degraded with inappropriate data decimation assignments. This means that different data streams have different impacts on specific application's fidelity. For instance, when F4 electrode maintains the full sampling rate and F3 only applies 20% data decimation rate, the depression detection fidelity drops sharply to 39%. The fidelity is much worse than what is given in our previous example, although the total data decimation rate of the two electrodes is much higher (a total data decimation rate of 40% comparing with 15%). From the variation of the MCI detection application fidelity shown in Figure 4.7, we can also see that even when the total data decimation rate is fixed, the fidelity may still vary with different data decimation assignments, e.g., with the total data decimation rate fixed as 45%, the application fidelity can vary from zero to 100%.

Disconnected. We find that the current design overlooks application fidelity requirements when the headset is disconnected. The Emotiv neuroheadset does not have any data buffer. When the headset is disconnected, the sampled data is completely lost, and the application fidelity is not provided during the disconnection period. As a result, the use of EEG headset will be interrupted when the subject cannot remain close enough to the local base station. For example, in the kitchen cooking trace we collected, the subject needs to wear the headset for more than 45 minutes to perform a 15 minutes effective EEG monitoring.

Summary. Based on the measurement results and analysis, we can see that the existing commercial neuroheadset does not take the realistic communication patterns and application fidelity requirements into its design consideration. The current neural signal sampling

design is inadequate for meeting the application fidelity requirements and handling real-time communication scenarios. Thus, we propose a new approach, which can automatically learn the sensitivity of application fidelity to EEG data, and utilize the learned sensitivity to cope with the mismatch between the neuroheadset’s wireless communication pattern and the application fidelity requirements. Note that our approach is not limited to the Emotiv EPOC Neuroheadset, as it does not depend on any specific design of the wireless EEG device.

4.2 Problem Definition and Analysis

Base on the data sensitivity to application fidelity, we apply different data decimation for different data sources to achieve different application optimization goals. Depending on whether the wireless link is well connected or disconnected, we formally define two application optimization problems. (1) In many scenarios as we have illustrated in Section 4.1, full data sampling rate is usually not needed to produce the requested application fidelity. So, we propose to optimize the total data decimation of all data streams while meeting the user requested application fidelity. Obviously, minimizing the total data decimation lowers the total energy usage, which is essential for the battery powered wireless EEG headset. (2) In the cases when the wireless communication is disconnected, a local buffer is used to temporarily cache the data for later delivery once the wireless link is connected again. Since the buffer space is limited, here we propose to optimize the usage of the limited space by optimizing the data decimation among different data streams, so that the weighted sum of all application fidelities is maximized.

In Table 4.1, we list the notations that we will use for the formal definition of these two problems.

Table 4.1: Notations for Problem Definitions

| Symbol | Definition |
|------------|---|
| a_i | The i^{th} application in the system, $i \in \{1..N\}$ |
| w_i | The weight associated with application a_i . Larger weights are given to more important applications |
| s_j | It is the j^{th} data stream in the system, $j \in \{1..M\}$. All M data streams are assumed to have the same full data sampling rate |
| S_i | It denotes the subset of data streams requested by application a_i . Different applications may require the same set of data streams. |
| t_j | The data decimation assigned to data stream s_j , $j \in \{1..M\}$ |
| T_i | The data decimation assignment, $T_i = \{t_{i1}, t_{i2}, \dots\}$, to the data streams set, $S_i = \{s_{i1}, s_{i2}, \dots\}$, that is requested by application a_i |
| $f_i(T_i)$ | It denotes the fidelity function for application a_i . The function input is the data decimation assignment T_i to the data streams set S_i that is requested by application a_i . The function output is the application a_i 's fidelity, which is in the range of $[0,1]$ |

4.2.1 Energy Minimization

When the wireless link is connected, we formally define the energy minimization problem as follows.

Definition 4.1 (Energy Minimization Problem) *For the purpose of minimizing energy, how to optimize the data decimation assignment T_i to the data streams used by application a_i , so that each a_i 's desired fidelity threshold F_i is satisfied and the total decimation*

$$\frac{1}{M} \sum_{j \in \{1..M\}} t_j \text{ is minimized.}$$

Here, we implicitly use the fact that more decimation (or more data) means more communication energy cost. Moreover, this problem can be formalized as follows:

$$\begin{aligned}
\min \quad & \frac{1}{M} \sum_{j \in \{1..M\}} t_j \\
\text{s.t.} \quad & f_i(T_i) \geq F_i
\end{aligned} \tag{4.5}$$

In this case, the constraint functions are described by *oracle models* (which are also called black box). In an oracle model, we don't know the f_i s explicitly, but we can evaluate f_i s manually or by computer program. This is referred to as querying the oracle [9]. In [9], it is also mentioned that some prior information like oracle model's convexity is sometimes given or could be assumed depending on the application context. Intuitively, in the energy minimization problem, more data decimation (more communication energy) always means more application fidelity. In short, it is reasonable for us to assume the fidelity function is an increasing function of the assigned decimation.

4.2.2 Application Fidelity Maximization

When the wireless connection is disconnected, we formally define the application fidelity maximization problem as follows.

Definition 4.2 (Application Fidelity Maximization Problem) *Given a data buffer size, that is a given upper threshold for $\frac{1}{M} \sum_{j \in \{1..M\}} t_j$, how to optimize the data decimation assignment $T_i = \{t_1..t_M\}$ for data streams, so that the sum of all applications' weighted fidelity $\sum_{i=1}^N w_i \cdot f_i(T_i)$ is maximized.*

Based on the definition, we find that it is a variant of Resource Allocation Problem (RAP). RAP aims to assign the available resources to all agents in an economic way. One

example of RAP is that a *gambler*, frustrated by persistent losses and envious of his *friends'* winnings, decides to allow a group of his fellow gamblers to make bets on his behalf. He decides he will wager a *fixed sum of money*, but that he will apportion his money among his friends based on how well they are doing. Certainly, if he knew psychically ahead of time which of his friends would win the most, he would naturally have that friend handle all his wagers. Lacking such clairvoyance, however, he attempts to allocate each race's wager in such a way that his *total winnings* for the season will be reasonably close to what he would have won had he bet everything with the luckiest friends [38].

Comparing to the above RAP example, in problem 4.2, the '*gambler* should be some coalition like users or programs and the *limited money* should be resources like buffer size. Moreover, the *friends* should be the data streams which contribute to the final winning while the *winnings* should be the applications' fidelities. Up to now, we have seen these two problems are similar to each other to some extent, but they still have some differences. The difference between the example and problem 4.2 is that each *friend* only affects his own winning output while each data stream affects several outputs (application fidelities). Therefore, we can see that the problem in Definition 4.2 is a variant of basic resource allocation problem.

Many algorithms exist for solving RAP. One class of resource allocation algorithms among them is the weighted majority algorithm class [38] whose basic framework is described as follows:

In the above framework, the weight vector represents how many resources should be allocated to each agent (could be gambler or application) and all resources are initially evenly allocated. After that, the algorithm goes into the allocating loop. In each round

Algorithm 4 Weighted Majority Algorithm Framework

Input: $\beta \in [0, 1]$, initial weight vector $\mathbf{u}^1 \in [0, 1]^N$ with $\sum_{i=1}^N u_i^1 = 1$, number of trials T

Output: The allocation vector \mathbf{p}^t

for $t = 1$ to T **do**

Choose allocation $\mathbf{p}^t = \frac{\mathbf{u}^t}{\sum_{i=1}^N u_i^t}$

Receive loss vector $\mathbf{l}^t \in [0, 1]^N$ from environment

Set the new weights vector to be $u_i^{t+1} = u_i^t \beta^{l_i^t}$

end for

of the loop, each agent's action is simulated and the loss of the agent is computed. Then all agents' losses are formed as a loss vector. With the loss vector, the algorithm adjusts each agent's weight based on the following rule: increasing more resources to the agent who makes profits in this round and reducing resources to those agents who lose in this round. The parameter β controls each adjustment's scale and each adjustment step determines how fast the current allocation strategy approaches the optimal solution. Thus, β can be used to control how fast the algorithm converges. When the next round begins, the resources are reallocated according to the new weight vector. The algorithm terminates after certain number of rounds and the final weight vector represents the final decision on how to allocate resources to the agents.

Since the weighted majority algorithm class has promising and solid convergence analysis result, we design an algorithm (in Section 4.4) based on it to solve the fidelity maximization problem.

4.3 Exploiting Data Sensitivity to Minimize Energy Consumption

Since data communication and processing dominate the energy cost of a sensing system, we assume that the total energy consumption is approximately proportional to the amount of data the sensing system needs to communicate and process. Thus, for the problem defined in Def. 4.1, when the wireless communication is reliable, we propose an energy minimization algorithm to minimize the data decimation. The proposed algorithm consists of two parts. First, for an individual application, a solution is given to determine the optimal data decimation assignment among the data streams required by this application. Then, considering all applications in the system, we propose to adjust the data decimation across different applications to minimize the total data decimation while still meeting the desired application fidelity thresholds.

4.3.1 Data Decimation for a Single Application

First, we formally define the problem of data decimation for a single application as follows:

Definition 4.3 (Single Application Data Decimation) *For application a_i , given a specified data decimation requirement $l_i = \frac{1}{|S_i|} \sum_{t_j \in T_i} t_j$, the problem is to find the optimal data decimation assignment $T_i = \{t_{i1}, t_{i2}, \dots\}$, so that the given application fidelity f_i is maximized.*

We notice that this problem is similar to the traditional sensitivity analysis [10] [11], which tries to identify the relative importance of each input (corresponding to each data stream in the problem stated in Def. 4.3). So, a basic sensitivity analysis technique called

Local Method is used here to assign the data decimation to different data streams of a single application. To decide the importance of each data stream s_j , the Local Method checks the simple derivative of the fidelity function f_i with respect to s_j 's data decimation change. While all other data streams use the same full sampling rate, s_j is applied with decimation t_j which is changed in K enumeration steps. We denote the change between the k^{th} and $(k+1)^{th}$ data decimation as Δt_{jk} , and the corresponding application fidelity change as Δf_{ik} , $k \in \{1..K-1\}$. Then, the average derivative for data stream s_j is $\frac{1}{K-1} \sum_{k=1}^{K-1} |\frac{\Delta f_{ik}}{\Delta t_{jk}}|$. Intuitively, with a larger value of the average derivative, the j^{th} data stream is more important to the application. Thus the decimation of data streams j is computed as:

$$t_j = \frac{l_i \times |S_i| \times \sum_{k=1}^{K-1} |\frac{\Delta f_{ik}}{\Delta t_{jk}}|}{\sum_{\forall t_j \in T_i} \sum_{k=1}^{K-1} |\frac{\Delta f_{ik}}{\Delta t_{jk}}|}. \quad (4.6)$$

4.3.2 Data Decimation for Multiple Applications

From Figures 4.7 and 4.8, we find that the application fidelity generally increases with the increase of data decimation. Based on this monotonous property, we design the main algorithm for multi-application data decimation, which is inspired by the bisection method for quasiconvex optimization [9]. This algorithm utilizes an approach similar to the binary search, which repeatedly adjusts the data decimation l_i to all data streams requested by each application a_i . This loop stops when all the resulting applications' fidelities $\{f_i\}$ s are above and close enough to the required thresholds $\{F_i\}$ s within a specified tolerance interval ϵ ($\epsilon > 0$). With each possible l_i , the data decimation is assigned to the data streams of a_i according to Equation 4.6.

Algorithm 5 Algorithm: Energy Minimization

Input: Fidelity functions $\{f_i(T_i)\}$, fidelity requirements $\{F_i\}$, the tolerance ϵ
Output: Decimation to all data streams $\{t_1..t_M\}$
 $\forall j = 1..M, t_j = 0$
for $i = 1$ to N **do**
 Initiate the data decimation upper bound $u_i = 1$ and lower bound $b_i = 0$
 Compute the initial data decimation for application a_i as $l_i = \frac{1}{2} \cdot (b_i + u_i)$;
 repeat
 /* Distribute the total data decimation l_i of application a_i to all data streams that
 application a_i requests, according to Equation 4.6*/
 for $t_j \in T_i$ **do**
 if $t_j <$ the result of Equation 4.6 **then**
 $t_j =$ the result of Equation 4.6
 end if
 end for
 Calculate $f_i(T_i)$
 if $f_i - F_i > 0$ **then**
 if $f_i - F_i < \epsilon$ **then**
 Break
 end if
 $u_i = l_i$
 else if $f_i - F_i < 0$ **then**
 $b_i = l_i$
 else
 Break
 end if
 $l_i = \frac{1}{2} \cdot (b_i + u_i)$
 until false
end for
 Return(the total data decimation $T = \frac{\sum_{i=1}^M t_j}{M}$)

4.4 Exploiting Data Sensitivity to Maximize Application Fidelity

When the data with the full sampling rate cannot be all immediately delivered due to network disconnection, the local buffer is utilized to cache data for later deliver when the network is connected again. In this section, we propose a novel algorithm to optimize the usage of the limited buffer, so that the total weighted application fidelity is maximized.

Given the total possible data decimation T , which is decided by the available buffer size, our algorithm iteratively adjust the data decimation among applications. This algorithm is inspired by the Weighted Majority Algorithm [38], which tackles the resource allocation problem in game theory. From each iteration, the trends of changes are learnt for both the total weighted fidelity and individual applications' fidelities. Then, the algorithm increases decimation for those applications that impact more on the total weighted fidelity. It also follows Equation 4.6 to compute the data decimation assignment to data streams requested by each application.

As the algorithm iteratively computes data decimation $\{t_1, \dots, t_M\}$, we denote the current iteration step index as q . The results of the q^{th} iteration are denoted with superscript q . The algorithm stops iterating after a specified number of rounds, which is denoted by Q . So $q \in \{1..Q\}$.

4.5 Evaluation

We evaluate the above two proposed solutions with trace-driven experiments. We use 20 EEG data traces collected from a human subject in an office environment. Each trace

Algorithm 6 Algorithm: Fidelity Maximization

Input: Fidelity functions $\{f_i(T_i)\}$, the total data decimation T , the iteration bound Q , $\beta \in [0, 1]$ (β controls the algorithm's converging speed)

Output: Decimation to all data streams $\{t_1..t_M\}$

*/** $\forall i$ initiate $\{f_i^0\}$, $\{l_i\}$ and fidelity change Δf_i^* */**

$\forall i, f_i^0 = 1, l_i = T, \Delta f_i = 0$

*/**initially assume $\forall j, t_j = T^*$ */**

$\forall j, t_j^1 = T$

Initiate the iteration step index $q = 1$

repeat

$\forall i$, compute f_i^q with $\{t_j^q\}$

*/** If the total weighted fidelity decreases, recompute Δf_i^* */**

if $\sum_{i=1}^N f_i^{q-1} \cdot w_i - \sum_{i=1}^N f_i^q \cdot w_i > 0$ **then**

$\forall i$, compute $\Delta f_i = (f_i^{q-1} - f_i^q) \cdot w_i$ for application a_i

end if

*/** Adjust l_i for each application a_i , according to Δf_i^* */**

for $i = 1$ to N **do**

if $\Delta f_i > 0$ **then**

$l_i = l_i * \beta^{-\Delta f_i}$

end if

*/** Compute the q^{th} data decimation for application a_i with the adjusted l_i , according to Equation 4.6**/**

for $j = i1..im_i$ **do**

if t_j^q is NULL OR $t_j^q <$ the result of Equation 4.6 **then**

$t_j^q =$ the result of Equation 4.6

end if

end for

end for

*/** Normalize $\{t_j^q\}$ */**

$\forall j, t_j^q = \frac{T \cdot M \cdot t_j^q}{\sum_{j=1}^M t_j^q}$

$q = q + 1$

until $q > Q$

Return (the data decimation assignment $\{t_1, \dots, t_M\}$)

contains data generated by six EEG electrodes (AF3/AF4, O1/O2, and F3/F4 as shown in Figure 4.1) over a 30-minute period with a 128Hz sampling rate. Two applications operate at the local base station: the first application computes the asymmetry score with the EEG data from electrodes F3/F4 for depression detection; the second application computes the qEEG ratio with the EEG data from electrodes AF3/AF4 and O1/O2 for MCI detection.

Algorithms 5 and 6 are evaluated in the following subsections 4.5.1 and 4.5.2, respectively. The 20 EEG data traces are separated into a training set and an evaluation set. Each of these two algorithms first runs on the training set to compute the data decimation assignment. Then, the computed decimation assignment is applied to the evaluation set. On both sets, the fidelity is computed in the way described in 4.1.2. The evaluation is repeated following a 5-fold cross-validation style, i.e., the 20 EEG data traces are divided into five groups with each group having four traces; each of the five groups is used in turn as the evaluation set, and the other four groups are used as the training set. In total, each experiment runs 5 rounds, and the fidelities achieved in these five rounds are averaged.

4.5.1 Evaluation of the Energy Minimization Algorithm

Figures 4.10 and 4.11 show the evaluation results of the Algorithm 5. The application fidelity requirements increase from 0.6 to 1, as shown by the x-axis of the two figures. Figure 4.10 shows the total decimation for all electrodes (2 electrodes for depression detection, and 4 electrodes for MCI detection) used by each application to achieve the required fidelity threshold. Figure 4.11 shows the achieved application fidelity when the decimation assignment computed by Algorithm 5 is applied to the evaluation set. We observe that both applications approximately achieve the required neurometric fidelities with the computed

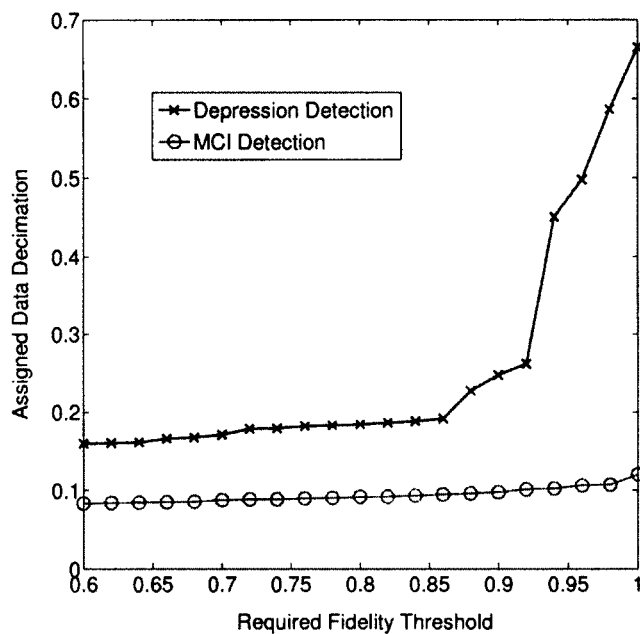


Figure 4.10: Decimation Assignment Computed with Energy Minimization Algorithm

decimation assignment, which is lower than the full sampling rate. So, energy can be saved by sampling and sending the decimated data only. We observe that when the required threshold is around 0.9, only less than 0.25 and 0.1 decimations are needed for the two applications, respectively. That is to say, 75% and 90% of the sampling and communication energy can be saved by our solution for each application, respectively.

Figure 4.10 also reflects different sensitivities of different applications' fidelities to the EEG data. The computed data decimation increases as the required fidelity threshold increases, but the decimation of data for depression detection needs to be largely increased to achieve a fidelity higher than 85%. This implies that when a higher application fidelity is required, a higher decimation should be assigned to the data that is requested by depression detection.

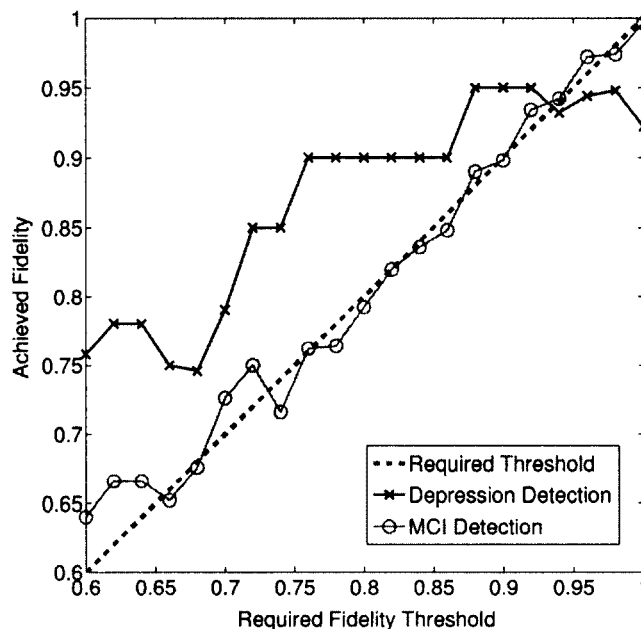


Figure 4.11: Fidelity Achieved with Energy Minimization Algorithm

Figure 4.11 shows the fidelity achieved by the decimation assignment computed by Algorithm 5. Most of the achieved fidelity is higher than or close to the required threshold for both applications. For MCI detection, the achieved fidelity is always slightly higher or roughly equal to the required fidelity. For depression detection, the achieved fidelity is higher than the required fidelity when the required fidelity is less than 92%, after which the requested high fidelity threshold is not met. One possible reason is that the fidelity is only an approximate, but not a strict, increasing function of data decimation (See Figure 4.8).

4.5.2 Evaluation of the Fidelity Maximization Algorithm

In this evaluation, we compare our fidelity maximization algorithm with the default even decimation assignment algorithm, which distributes the total decimation evenly to the six needed electrodes.

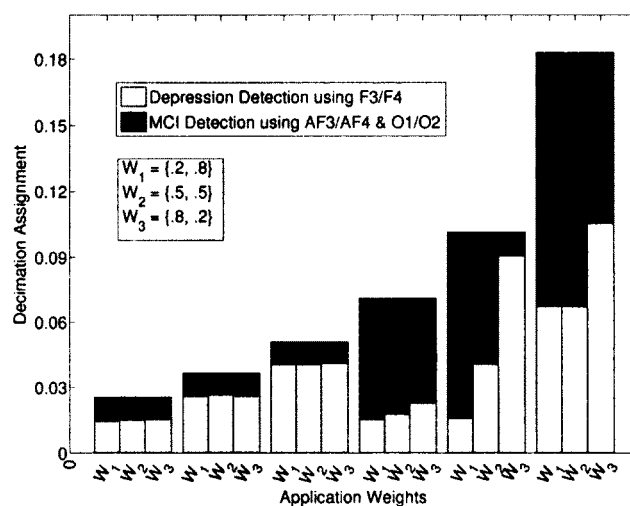


Figure 4.12: Decimation Assignment Computed with Fidelity Maximization Algorithm

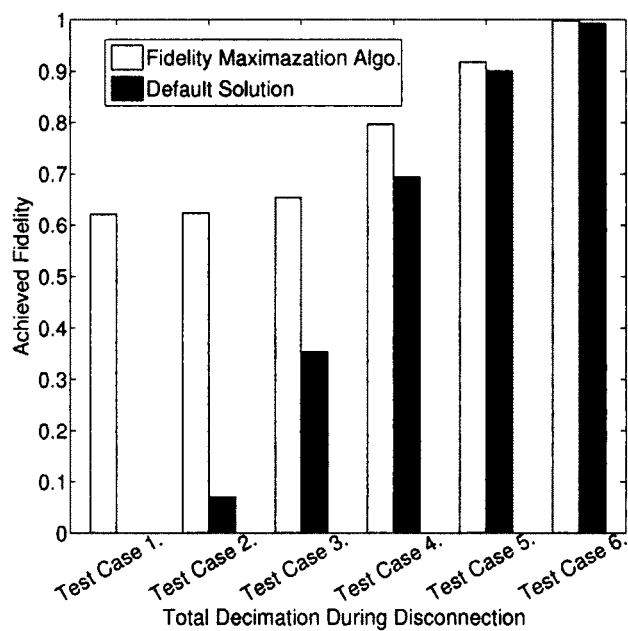


Figure 4.13: Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 = \{.2, .8\}$

When the wireless communication is reliable, we use the energy minimization algorithm to achieve perfect application fidelity with the minimum energy consumption. When the

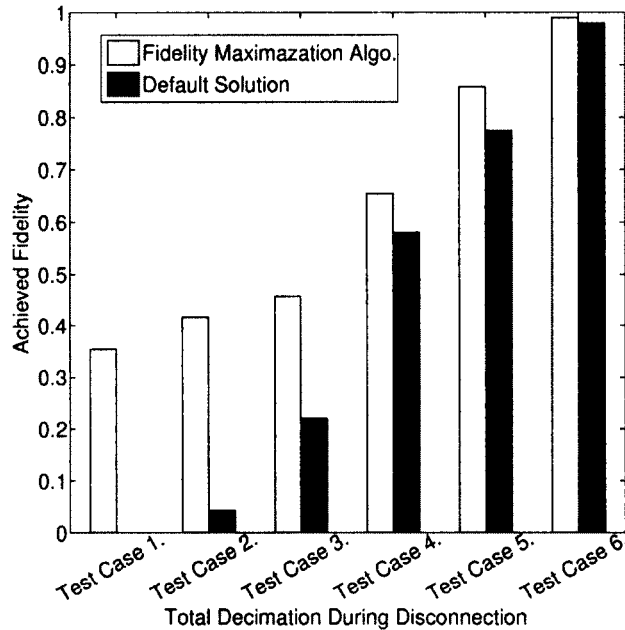


Figure 4.14: Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 = \{.5, .5\}$

wireless communication is disconnected, we incorporate a limited data buffer into the EEG headset to save sampled data temporarily for later delivery.

During the disconnection period, the fidelity maximization algorithm is also called to maximize the total weighted application fidelity according to the limited buffer size.

As demonstrated in Figure 4.10, only a small decimation rate is needed in order to achieve the perfect application fidelity. This suggests that we need a smart way to allow us to focus on a small range of low decimation rate in order to effectively evaluate our fidelity maximization algorithm. Therefore, in this experiment, we first apply our energy minimization algorithm to get the minimally needed decimation rate. Then, we further apply a second round of decimation with six different rates (six test cases), including 12.5%, 18%, 25%, 35%, 50%, and 90%, which correspond to different data buffer sizes.

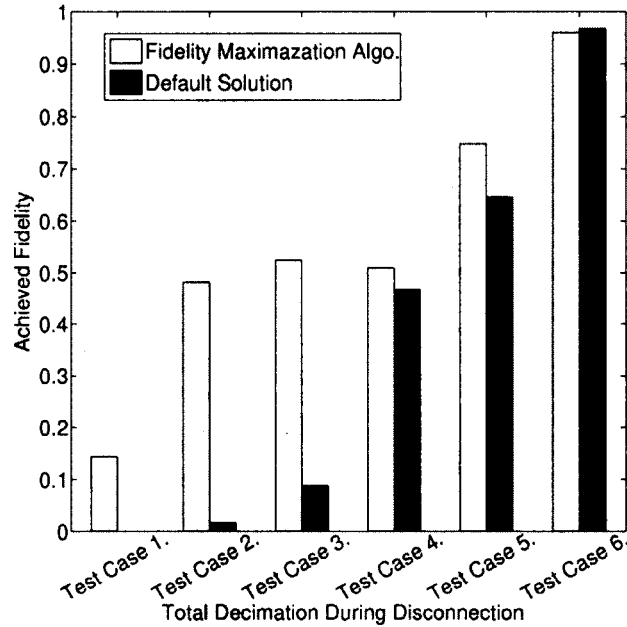


Figure 4.15: Fidelity Achieved with Fidelity Maximization Algorithm: App Weight $W_1 = \{.8, .2\}$

Different application weights are also allowed for the two applications. The weight vector $W_1 = \{.2, .8\}$ means that the depression detection application and the MCI detection application have priority weights of 0.2 and 0.8, respectively. The weight vector $W_2 = \{.5, .5\}$ means that the depression detection application and the MCI detection application each has the same priority weight of 0.5. The weight vector $W_3 = \{.8, .2\}$ means that the depression detection application has the the priority weight of 0.8, and the MCI detection application has the priority weight of 0.2. Figure 4.12 plots the allocated decimation assignment and Figure 4.13 4.14 and 4.15 plots the achieved application fidelity, with the input of different application weight vectors, respectively.

In Figure 4.12, the three bars in each of the six test cases show the decimation assignments computed with one of the six given total decimation rates as well as the three different

combinations of the application weights. The figure shows that the fidelity maximization algorithm is able to adapt the decimation assignment to meet different application fidelity and decimation rate requirements. In the first three test cases, we find that similar decimation assignments are computed for the MCI detection application. This suggests that with a very low total decimation rate ($\leq 25\%$), the application's fidelity cannot be largely improved with a small decimation increase. So, the algorithm assigns more decimation to the depression detection so as to achieve a higher total fidelity. In the last two test cases, we can see that the algorithm assigns more decimation to the MCI detection application when it becomes more important. This is shown in the first bar of each test case that corresponds to the weight vector $W_1 = \{.2, .8\}$. When the depression detection application becomes more important, as shown in the third bar of each test case that corresponds to the weight vector $W_3 = \{.8, .2\}$, the algorithm assigns more decimation to the depression detection application.

From Figure 4.13 4.14 and 4.15, when the buffer size is very small, as shown in the first three test cases of each figure, we observe that the default even decimation assignment algorithm achieves very low fidelity and is always largely outperformed by our proposed algorithm. This is because, as demonstrated in Figure 4.10, the MCI detection application can easily achieve a very high fidelity with a very small decimation rate compared with the depression detection application. Our proposed algorithm is able to automatically learn the sensitivity of different applications' fidelities to the EEG data. Then, based on the learned sensitivity, our algorithm chooses to first satisfy the application that requires less decimation to achieve a high application fidelity. On the other hand, the default algorithm assigns the same decimation to all data streams, ignoring the sensitivity difference between

different applications' fidelities. Thus, more than necessary decimation is assigned to the data streams of one application, but the other application's fidelity is harmed because of inadequate decimation assigned.

From Figure 4.13 4.14 and 4.15, when the buffer size is not very small, as shown in the last three test cases of each figure, we observe that our design is still better than the default solution most of the time. In these three test cases, both algorithms assign high decimations to both applications because the total allowed decimation rate is high. With the high decimations assigned, the MCI detection application's fidelities achieved by both solutions are close to 1, and the depression detection's fidelities achieved by these two solutions do not have obvious difference. This is because, as shown in Figure 4.10, when the decimation is higher than or equal to 20%, a small increase of the decimation does not lead to a large improvement of the depression detection application's fidelity.

4.6 Conclusions

Meeting applications' fidelity requirements as well as saving energy are two central issues of incorporating EEG neurometrics into in-situ and ubiquitous physiological monitoring. In this work, we measure the realistic neuroheadset communication with an off-the-shelf Emotive EPOC Neuroheadset, and reveal a mismatch between the lossy EEG sensor communication pattern and the high neurometric application fidelity requirements. Then, taking the MCI detection and the depression detection as two example neurometric applications, we propose a generic approach to automatically learn the sensitivity of application fidelities to the EEG data. With the learned sensitivity, we propose an energy minimization algorithm

to minimize the energy usage in EEG sampling and networking when the wireless communication is reliable. We also propose a fidelity maximization algorithm to maximize the total weighted applications' fidelities when the wireless communication is poor. Through trace-driven experiments, our proposed solutions are demonstrated to outperform existing ones. In addition to optimizing the data decimation as proposed in this work, we also plan to optimize other network metrics like data resolution, as well as to build network models in our future work, in order to meet applications' fidelity requirements and save energy.

CHAPTER 5

EXPOSING COMMUNICATION QOS AVAILABILITY TO OTHER BSNS

When multiple Body Sensor Networks (BSNs) exist within the communication range of each other, information sharing among BSNs can be very beneficial to applications, such as improving accuracy and saving energy. However, how to provide communication quality assurance in coexisting and shared multiple BSNs is still an open problem. On one hand, a BSN should prevent possible interference from other BSNs' traffic to degrade the quality of intra-BSN links. On the other hand, a BSN should be able to discover its buddy BSNs in the neighborhood, and establish inter-BSN connections with Quality of Service (QoS) assurance. In this work, we propose a framework called BuddyQoS to provide throughput

assurance for coexisting and shared multiple BSNs. BuddyQoS estimates and properly schedules the wireless resources to all inter- and intra-BSN communications. Through trace-driven experiments, we demonstrate that our approach outperforms the default best effort solution using CSMA in TinyOS.

5.1 BuddyQoS Overview

We propose a QoS solution, the BuddyQoS framework, for multiple coexisting and shared buddy BSNs. Our proposed solution receives the throughput requirements from the application layer, and returns admission decisions. The application assigns each requirement with a global priority, which denotes how important the requirement is. When the available resource is not enough to satisfy all the requirements, BuddyQoS uses the priority to decide which requirements are less important and can be rejected.

In addition, BuddyQoS performs the share coordination to establish sensor sharing between different BSNs. To address the privacy concerns for sensor sharing across different BSNs, we assume that only some non-private sensor data, such as environment temperature readings, and background noise samples, are shared between buddy BSNs. Also, we argue that the sensor sharing are more likely to happen when different BSNs are in the communication range of each other. Thus, the physical distance between people wearing the BSNs is short enough for them to visually recognize the activities of others, so the sensor sharing does not release more private information. The application layer is responsible for determine whether to share data to or from another neighboring buddy. When BuddyQoS notifies the application layer of a new neighboring BSN, the application layer should first

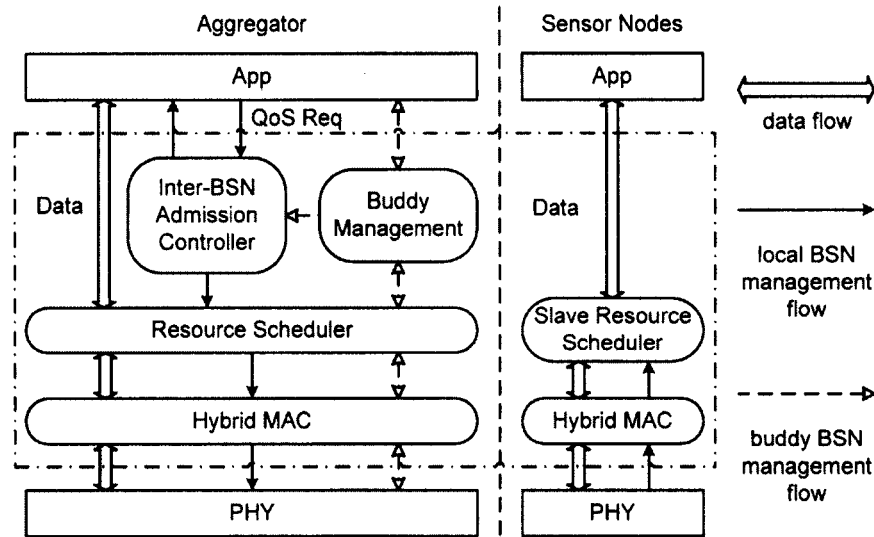


Figure 5.1: BuddyQoS Architecture

decide whether the BSN can be trusted. Then, the applications estimate whether they can benefit from the sensor sharing, with either enhanced performance or energy saving. If the applications allow sharing, they pass the sharing requests or decisions to BuddyQoS, which establishes the sensor sharing between different BSNs.

Figure 5.1 shows the architecture of BuddyQoS, which consists of four main components: the Hybrid MAC, the Resource Scheduler, the Inter-BSN Admission Controller, and the Buddy Management module.

The BuddyQoS Hybrid MAC sits on both the aggregator side and the sensor node side, above the PHY layer. The Hybrid MAC is responsible for transmitting and receiving packets, including data packets and management messages, for upper layers. We especially design the Hybrid MAC to combine the advantages of the TDMA scheme, for easy resource estimation and scheduling, and the CSMA scheme, for its flexibility.

The Resource Scheduler on the aggregator side collaborates with the Slave Resource Scheduler on the node side to schedule resource for communications in the local BSN. 1)

On the aggregator side, the Resource Scheduler first receives a list of admitted throughput requirements from the Inter-BSN Admission Controller, and other BSNs' management information from the Buddy Management module. Then, the Resource Scheduler computes the schedule for the local BSN, and enforces the schedule on aggregator. 2) Correspondingly on the node side, a Slave Resource Scheduler receives the schedule from the local aggregator, and enforces the schedule on node.

The Inter-BSN Admission Controller on the aggregator side is responsible for making admission decisions for QoS requirements from local and neighboring buddy BSNs. First, when the application layer inputs throughput requirement for local data streams, this module estimates the resource needed to provide throughput assurance for communications in the local BSN. Then with the resource schedule of neighboring buddy BSNs from the Buddy Management module, the total resource needed to satisfy requirements for all local and buddy BSNs communications is computed. If the available resource is less than the total resource required, the Inter-BSN Admission Controller rejects some of the QoS requirements, according to their priorities. Otherwise, the QoS requirements are accepted and maintained in a list, which is output to the resource schedule, and the admission decisions are passed to the application layer.

The Buddy Management module on the aggregator side handles the management information from neighboring buddy BSNs. When discovering a new buddy BSN, the Buddy Management module notifies the application layer. Then with sharing requests / decisions from the application layer, the Buddy Management module establishes the sensor sharing between buddy BSNs. Also, the Buddy Management module inputs the neighboring buddy BSNs' resource schedules to the Inter-BSN Admission Controller to help making admission

decisions.

In Figure 5.1, three flows pass through the BuddyQoS framework. The data flow shows how the application data are handled by BuddyQoS to be transmitted between aggregators and sensor nodes. The local BSN management flow indicates how the information for local BSN communication scheduling passes through the BuddyQoS modules. The inter-BSN management flow carries the management information of neighboring buddy BSNs, including the buddy BSN schedules, sensor sharing requests and decisions, to coordinate the communication among multiple BSNs.

5.2 BuddyQoS Hybrid MAC Design

BuddyQoS uses a specific MAC design with the hybrid scheme, whose main idea is borrowed from Z-MAC [57]. Because on one hand, to provide QoS assurance, BuddyQoS needs to estimate the resource needed for the communication and make proper schedules, which is easy to do on a TDMA MAC. On the other hand, in the scenario of multiple BSNs, to eliminate contention from different BSNs which are highly dynamic with the mobility of human beings, the flexibility of a CSMA scheme is desired. The Hybrid MAC design combines the advantages of both CSMA and TDMA schemes, and is especially useful in the multiple BSNs scenario. Based on the Hybrid MAC, we develop the communication paradigm for data delivery, inter- and intra- BSN management information exchange.

Assume that the aggregators and sensor nodes in the neighboring BSNs are all synchronized, and the time is divided into slots. When an aggregator or a sensor node needs to send packets, each packet is sent in a time slot. So, the number of time slots needed for inter-

and intra-BSN communications can be estimated, and the Resource Scheduler assigns the time slots to the local transmitters.

Within each slot, the scheme used for sending the packet resembles the CSMA scheme. When a transmitter has a packet to send in a time slot, it first backs off some time before carrier sensing the channel. Then if the channel is found clear, the packet can be sent. When there are multiple transmitters accessing a same slot, one of them can be assigned as the “owner”, which is the same notion as in the Z-MAC design. The slot owner accesses the slot with the minimum backoff $T_{backMin}$, and other transmitters randomly back off between the minimum and maximum backoffs, $[T_{backMin}, T_{backMax}]$. Given a TDMA schedule, if a time slot is assigned to a transmitter, then the transmitter is the owner of that slot. Here the minimum backoff, $T_{backMin}$, can be set to tolerate time synchronize errors, and the maximum backoff, $T_{backMax}$, affects the setting of the time slots length. This scheme allows other transmitters to utilize the time slot when the owner does not have any packet to send. In addition, the sending priorities of transmitters can be decided by the values of the backoff, i.e., a low priority transmitter should back off longer.

5.3 Inter-BSN Admission Controller and Resource Scheduler

The Inter-BSN Admission Controller and the Resource Scheduler play the key role to provide throughput assurance for communications in coexisting and shared BSNs. In this section, we first develop the communication paradigm based on the Hybrid MAC to support data multicast with throughput assurance in multiple BSNs. With the paradigm, the resource needed to satisfy the throughput requirements from applications can be estimated. Based on

the estimation, the Inter-BSN Admission Controller makes admission decisions to accept or reject throughput requirements. The Resource Scheduler schedules the resource for inter- and intra-BSN communications to provide throughput assurance for all the admitted requirements.

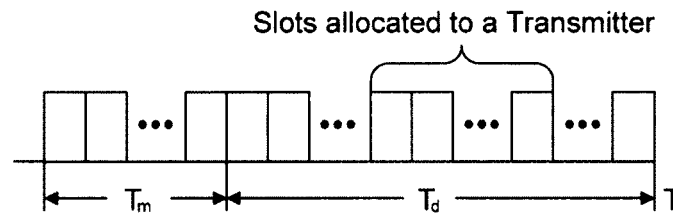


Figure 5.2: Interval T Divided into Two Periods

5.3.1 Communication Paradigm for Shared BSNs

As shown in Figure 5.1, the Inter-BSN Admission Controller and Resource scheduler are responsible for estimating and allocating resources to deliver both the application data, and inter- and intra-BSN management information. These different kinds of communications are scheduled respectively in the two periods of each time interval, as illustrated in Figure 5.2. The first period contains T_m slots, and is used for aggregators to broadcast inter- and intra-BSN management information. The second period contains T_d slots, and is used for data delivery.

In multiple coexisting and shared BSNs, when a sensor node sends out data packets, not only the local aggregator, aggregators from neighboring buddy BSNs sharing the sensor node may also be listening to the data packets. Thus, a communication paradigm to support throughput assurance for the multicast is necessary.

A receiver may use ACK messages to notify the transmitter of the received or lost pack-

ets. If multiple receivers lose data packets at the same time, all of them would ask the transmitter for retransmission; thus, the amount of ACK messages can be large. There has already been extensive studies on reducing the large number of ACK messages, known as the “ACK implosion” problem, in building reliable broadcast [54] [65]. But our problem is different from the reliable broadcast problem, as packet losses are allowed, when we utilize retransmission to provide the data throughput assurance. Similar to the TCP acknowledgment design [19], BuddyQoS uses the selective NACK (SNACK) message instead of ACK message to notify transmitter for retransmission, thus to avoid the collision and reduce cost of sending multiple ACKs. Specifically, before all transmissions, all receivers are informed of the schedule and the number of data packets to be sent during the schedule. Each receiver records the received packets, and uses a bit vector to compactly store the sequence numbers of multiple lost packets. Then, each receiver only need to send one SNACK message containing its bit vector to the transmitter.

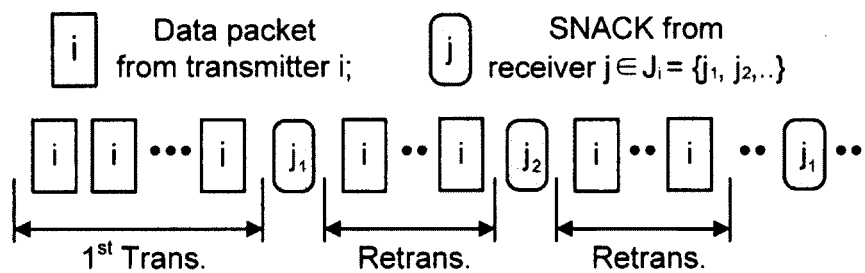


Figure 5.3: Communication Paradigm for Shared BSNs

Figure 5.3 shows the data packet transmissions with SNACKs during the time slots scheduled for a transmitter. The first several slots are assigned to transmitter i to transmit its packets. Then the receivers can use the next time slot to send their SNACK message. The SNACK slot is not assigned to any specific owner, so with the Hybrid MAC, every

receiver has a chance to access the slot. When one of the SNACK messages is transmitted successfully, e.g., j_1 's SNACK in Figure 5.3, the following slots are assigned to the transmitter for data retransmission. After retransmissions, if any receiver still has packets missing, it tries to access the next slot to send out a SNACK.

The use of SNACK greatly reduces the cost of multiple ACKs, and with SNACK suppression, the number of SNACKs can be further reduced. Figure 5.4 shows a SNACK suppression example, where two receivers j_1 and j_2 are both listening to transmitter i . The number in each data packet denotes its sequence number, and numbers in the SNACK message denote the sequence numbers of lost packets stored in the bit vector. After the first batch of transmissions, receiver j_1 loses the 2nd and 4th packets, and receiver j_2 loses the 2nd packet. Utilizing the packet overhearing, a receiver can receive the SNACK packet from other receivers and compare their lost packets. In Figure 5.4, j_2 overhears j_1 's SNACK, and finds out that the set of j_1 's lost packets is

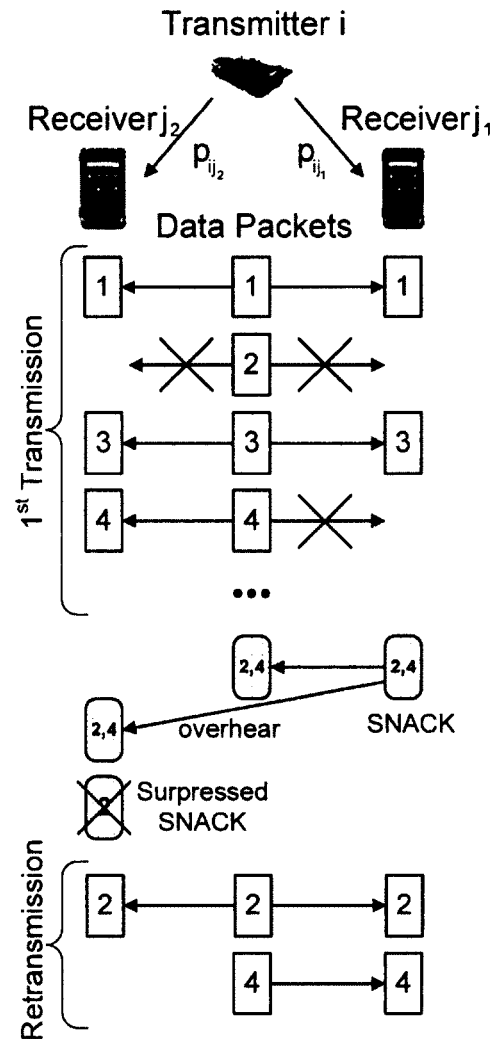


Figure 5.4: SNACK Suppression Example

In Figure 5.4, j_2 overhears j_1 's SNACK, and finds out that the set of j_1 's lost packets is

a superset of its own set of lost packets. Then, j_2 suppresses its own SNACK packet, and receives the lost packets during the following retransmission answering to j_1 's SNACK. In this example, one SNACK message is saved.

To encourage the SNACK suppression, receivers with high packet loss rates, i.e. j_1 in the example, should send their SNACKs before those with low packet loss rates, i.e. j_2 in the example. So with the BuddyQoS Hybrid MAC, each receiver sets its backoff for accessing the SNACK slot according to its packet loss rate. The higher the packet loss rate, the shorter the backoff.

5.3.2 Resource Estimation for Shared BSNs

The resource estimation is needed in both the Inter-BSN Admission Controller and the Resource Scheduler. The total resource needed equals the resource for management information exchange plus the resource for data delivery. In other words, the resource estimation computes the number of slots needed in T_m and T_d periods, respectively.

5.3.2.1 Resource Estimation for Management Information Exchange

During each T_m period, every aggregator sends one management message containing its inter- and intra-BSN management information. Therefore, the number of time slots needed in the T_m period depends on the number of buddy BSNs in the neighborhood, which is flexible. Also, extra slots in T_m should be left open for new buddy BSNs coming to the neighborhood.

At the beginning, T_m is initiated to T_{init} . Then every interval, the length of T_m is reevaluated with the number of neighboring buddy BSNs input from the Buddy Manage-

ment module. If the current length of T_m minus the number of neighboring buddy BSNs is less than a threshold ΔT , T_m is recomputed to be $T_m + \Delta T$. ΔT is a parameter in the system configuration. If a larger number of BSNs come and leave the neighborhood frequently, ΔT can be set higher.

5.3.2.2 Resource Estimation for Data Delivery

For data delivery during each T_d , the resource estimation computes the number of time slots needed each data stream from local nodes, including the slots needed for delivering required data packets, SNACKs from receivers, and data packet retransmissions. The number of data packets to be transmitted is decided by the throughput requirement on each data stream. The number of SNACKs and retransmissions depend on the packet loss rates of each receiver. Following denotations are used in the resource estimation:

- b_i : the throughput requirement for the data stream from sensor node i ;
- S_{pkt} : the effective payload size of each data packet;
- q_{ij} : the packet loss rate for each receiver j from transmitter i ;
- R : the maximum number of transmissions for a data packet. The data packets fail for R times should be discarded, because data collected long ago is meaningless and can queue up in the buffer during congestion, causing buffer overflow;
- J_i : the set of IDs of the receivers listening to node i , $J_i = \{j_1, j_2, \dots\}$. Thus, the number of i 's receivers is cardinality of set J_i , denoted as $|J_i|$

- D_i : the number of data packets needed to be delivered in each interval T from transmitter i to satisfy the throughput requirement from the applications. D_i can be computed from the throughput requirement b_i and the effective packet payload size S_{pkt} , $D_i = \lceil \frac{b_i \times T}{S_{pkt}} \rceil$;
- $E(K)$: let K be the number of data packet transmissions (including retransmissions) for each data packet to be successfully delivered to all receivers; $pr.\{K=k\}$ is the probability of each data packet being transmitted k times before successfully received; then, $E(K)$ stands for the expected number of transmissions, $E(K) = \sum_{k=1}^R k \times pr.\{K=k\}$.
- $E(N_j)$: let N_j be the number of SNACKs sent by receiver j ; $pr.\{N_j=n\}$ is the probability of receiver j sending exactly n SNACKs; then, $E(N_j)$ stands for the expected number of j 's SNACKs, $E(N_j) = \sum_{n=1}^{R-1} n \times pr.\{N_j=n\}$.

To compute the resource needed for the data delivery of each node i , the throughput requirement b_i is given by the application layer, and S_{pkt} and R are from system configurations. Besides, the packet loss rate q_{ij} is computed using the moving average of the history q_{ij} and the q_{ij} measured in current time by each receiver: $q_{ij} = \alpha \times q_{ij.history} + (1-\alpha) \times q_{ij.current}$. Here α is the decay factor. With these inputs, to deliver the required data packets from each sensor node i , the number of slots needed equals the number of data packets to be sent multiplied by the expected number of transmissions for each data packet, $D_i \times E(K)$. For SNACKs from all receivers in J_i , the number of slots needed equals the sum of the expected number of SNACKs from all receivers, $\sum_{j \in J_i} E(N_j)$. Details for computing $E(K)$ and $E(N_j)$ are explained in the following paragraphs of this subsection.

The expected number of data packet transmissions $E(K)$ is computed using Equation 5.1~5.4.

$$E(K) = \sum_{k=1}^R k \times pr.\{K=k\} \quad (5.1)$$

$$pr.\{K=k\} = \begin{cases} pr.\{K \leq k\} - pr.\{K \leq k-1\} & , k < R \\ 1 - pr.\{K \leq R-1\} & , k = R \end{cases} \quad (5.2)$$

$$pr.\{K \leq k\} = \prod_{j \in J_i} pr.\{K_j \leq k\} \quad (5.3)$$

$$pr.\{K_j \leq k\} = 1 - q_{ij}^k \quad (5.4)$$

Firstly, Equation 5.1 shows that $E(K)$ can be computed with $pr.\{K=k\}$. Using Equation 5.2, we compute $pr.\{K=k\}$ with $pr.\{K \leq k\}$. Then, to compute $pr.\{K \leq k\}$, we further look into the number of transmissions needed for the packet to be successfully received by each receiver j , which we denote as K_j . For all receivers $j \in J_i$ to receive the data packet from transmitter i , the number of transmissions needed is $K = \max_{j \in J_i} \{K_j\}$. Assume that the packet loss for each receiver is independent, $pr.\{K \leq k\}$ can be calculated using $pr.\{K_j \leq k\}$ (the probability of transmitting less than or equal to k times for receiver j to successfully receive the data packet) of each receiver j , as shown in Equation 5.3. After that, for each receiver j , we use its packet loss rate q_{ij} to compute $pr.\{K_j \leq k\}$. It is easy to see that if a data

packet is transmitted more than k times before receiver j successfully receives it, the first k transmissions to j must have all failed, and the probability is q_{ij}^k . So $pr.\{K_j \leq k\} = 1 - q_{ij}^k$, thus Equation 5.4 is derived.

Finally, to deliver data packets from transmitter i with throughput assurance, the total number of slots needed equals:

$$D_i \times E(K) = \lceil \frac{b_i \times T}{S_{pkt}} \rceil \times (R - \sum_{k=1}^{R-1} \prod_{j \in J_i} (1 - q_{ij}^k)) \quad (5.5)$$

The expected number of SNACKs sent from a receiver j $E(N_j)$ is computed using Equation 5.6~5.8.

$$E(N_j) = \sum_{n=1}^{R-1} n \times pr.\{N_j = n\} \quad (5.6)$$

$$pr.\{N_j = n\} = \begin{cases} pr.\{N_j \leq n\} - pr.\{N_j \leq n-1\} & , n < R \\ 1 - pr.\{N_j \leq R-2\} & , n = R \end{cases} \quad (5.7)$$

$$pr.\{N_j \leq n\} = (1 - q_{ij}^n)^{D_i} \quad (5.8)$$

Similar to the computation of $E(K)$, according to Equation 5.6 to compute $E(N_j)$ for each receiver j in J_i , we first compute $pr.\{N_j = n\}$. Using Equation 5.7, we compute $pr.\{N_j = n\}$ with $pr.\{N_j < n\}$. Then we calculate $pr.\{N_j < n\}$ using j 's packet loss rate q_{ij} . It is easy to see that when j sends more than n SNACKs for a data packet, the first n

transmissions for this packet must have failed, with the probability of q_{ij}^k . So the probability of j sending less than or equal to n SNACKs for one packet is $1 - q_{ij}^n$. Also, if j sends out less than or equal to n SNACKs, it means that each of the D_i data packets must have failed less than n times. We assume that the loss of each packet is independent, and then Equation 5.8 is derived.

Finally, the total number of slots needed for transmitting SNACKs from all receivers of the D_i data packets to transmitter i equals:

$$\begin{aligned} \sum_{j \in J_i} E(N_j) &= \sum_{j \in J_i} \left(R - \sum_{n=1}^{R-2} (1 - q_{ij}^n)^{D_i} \right) \\ &= R \times |J_i| - \sum_{j \in J_i} \sum_{n=1}^{R-2} (1 - q_{ij}^n)^{D_i} \end{aligned} \quad (5.9)$$

Note that Equation 5.8 does not consider the SNACK suppression, because it would require extra information of the correlation between the packet losses of different receivers, which is too costly to measure in the real scenarios. We leave this to be the future work to find out a lightweight real-time measurement of the packet loss correlations. However, the estimation of Equation 5.8 is always larger than or equal to the number of SNACK slots actually used, so there are still enough resource allocated to each node to satisfy the throughput requirements for them.

As a result of the resource estimation, the total resource needed for the data delivery of each transmitter i is the sum of Equation 5.5 and Equation 5.9.

5.3.3 Admission Decisions

The Inter-BSN Admission Controller in each BSN estimates the resource needed to satisfy the throughput requirements in the local BSN, and makes admission decisions to the application layer.

First, resource estimation is performed to calculate the total resource needed for both information exchange and data delivery for all buddy BSNs in the same neighborhood. The length of T_m is determined as described in Section 5.3.2.1. Meanwhile, the Buddy Management module also inputs the data delivery resource needed by other buddy BSNs to the Inter-BSN Admission Controller. With the local BSN data delivery resource estimation calculated as described in Section 5.3.2.2, the Inter-BSN Admission Controller sums up the total resource needed for the data delivery of each transmitter i in the local BSN and all other neighboring buddy BSNs.

Then admission decisions are made based on whether the total time slots needed for all data deliveries is larger than the T_d slots available, where $T_d = T - T_m$. If the available resource is sufficient, all local throughput requirements, as well as other buddy BSNs' throughput requirements, can be accepted. Otherwise, each aggregator decides on their own to reject some of the throughput requirements. According to the global priority assigned to each throughput requirement the rejection decisions are made. The Inter-BSN Admission Controller rejects requirements with lowest priorities, until the rest of the requirements can be satisfied. Once the Inter-BSN Admission Controller finds out any local requirement need to be rejected, it notifies the application layer that the corresponding requirement can no longer be satisfied. To the applications, this means that for one thing, the data stream is

only delivered with best effort from then on, and for another, its QoS requirement can be degraded in order to be admitted again. Finally, the set of accepted local requirements are passed to the Resource Scheduler.

5.3.4 Resource Scheduler

For each BSN, the Resource Scheduler decides the schedule of its management information broadcasting in T_m , as well as its local data delivery in T_d . In addition, the Resource Scheduler on the aggregator and the Slave Resource Scheduler on each sensor node are responsible for enforcing the local data delivery schedule.

Both inter- and intra-BSN management information for each BSN are sent in a management messages each T_m . The Resource Scheduler decides the slot in T_m to send its local management message. The buddy list maintained by the Buddy Management module is passed to the Resource Scheduler, and the order of the management messages from each buddy BSNs should be the same as the order of the buddy list.

For the local BSN data delivery scheduling, first the Resource Scheduler looks up the buddy list to find out the schedules of other buddy BSNs. The sequence of data delivery schedule of each buddy BSN is the same as that of the management messages sent in T_m . The Resource Scheduler adds up the resource needed by the BSNs before its local BSN is scheduled in the list, and determines when to start its local schedule in T_d . Then, with list of admitted requirements from the Inter-BSN Admission Controller, the Resource Scheduler estimates the resource needed for each local node, according to Equation 5.5 and Equation 5.9.

To enforce the schedule, the Resource Scheduler and Slave Resource Scheduler looks

up the schedules of local and buddy BSNs, in order to find out the time slots allocated to the aggregator and sensor node to send packets, as well as the time slots allocated to the transmitters it wants to listen to. During T_m , all aggregators and sensor nodes listens to all management information sent out. During T_d , each aggregator listens to all local schedules and schedules of other BSNs' nodes, which it wants to share. Each sensor node delivers its data packets according to the communication paradigm described in Section 5.3.1.

5.4 Buddy Management Design

The Buddy Management module maintains the management information of neighboring buddy BSNs, including three lists: 1) the buddy list, which is a list of buddy BSNs in the neighborhood; 2) the listen list, which is a list of sensors from other BSNs that the local aggregator listens to; and 3) the share list, which is a list of BSNs sharing local sensor nodes. From the other BSNs' management messages received during T_m , the Buddy Management module extracts the inter-BSN management information to update its local records. This information is then passed to the Inter-BSN Admission Controller and the Resource Scheduler to help making admission decisions and resource schedules. Also, the Buddy Management module uses the management messages to perform neighbor discovery and sensor sharing coordination.

5.4.1 Neighbor Discovery

During the T_m period, each aggregator listens to the management messages to find out other buddy BSN entering and leaving the neighborhood. The Buddy Management module

is responsible for maintaining a list of neighboring buddy BSNs according to the received management messages.

The Buddy Management module performs neighbor discovery based on the Hybrid MAC. When the Buddy Management module passes the buddy list to the Resource Scheduler, which determines the slot to send its management message, according to the order of the BSN list, and the backoff before accessing that slot. The Resource Scheduler never assigns any owner for the 1st slot in T_m . So, the aggregator of first BSN in the buddy list should access the slot with some backoff in the range of $[T_{backMin}, T_{backMax}]$. If an aggregator successfully sends its management message in the 1st slot of T_m , its buddy list is also sent out in the management message. In this way, this aggregator claims the following time slots for its known neighboring buddies. Then from the 2nd slot of T_m , the aggregators of the BSNs in the buddy list send their management messages one after another, all using minimum backoff $T_{backMin}$. Only after the BSNs in the first BSN's buddy list have all sent out their management messages, the new BSNs (not in the buddy list) can try to access the following slots with no owners, using backoffs between $[T_{backMin}, T_{backMax}]$.

Thus, the neighboring BSNs send management messages in order. When an aggregator hears the management message of a new BSN, its Buddy Management module adds the new BSN to its buddy list. When an aggregator is not heard for several consecutive intervals, it is considered out of the neighborhood and removed from all other BSNs' buddy lists.

According to this process, initially each BSN has no neighbors but only itself in its buddy list, so it tries to access the 1st slot with some backoff. Here we use the aggregator's ID to be the ID of its local BSN. When a new BSN comes to the neighborhood, say BSN j_2 coming to the neighborhood of BSN j_1 , at first, both of them compete the 1st slot. If

j_1 wins the 1st slot, then j_2 adds j_1 to its buddy list before itself, and tries to access the next slot. Then j_1 hears j_2 's management message in a later slot, it adds j_2 to its buddy list after itself. Thus, the two BSNs discover each other.

For the first BSN of the buddy list, it decides the backoff value according to the length of the buddy list. The more buddy BSNs following the local BSN in the buddy list, the shorter backoff will be used for sending its management message. So when a new BSN comes to the neighborhood of multiple BSNs who already know each other, it will fail accessing the 1st slot in T_m , and transmit its management message after all existing BSNs.

5.4.2 Sensor Sharing Coordination

The management message also includes the listen list and the share list from the Buddy Management module. Using these two lists, the sensor sharing coordination is performed. During this process, the listen list acts as a list of sharing requests, and the share list acts as a list of sharing decisions, answering the requests. The details of the sharing coordination are described as follows:

First, the Buddy Management module extracts the available sensor nodes from the schedule contained in the management message of a new neighboring buddy BSN, and reports to the application layer. Then the application layer decides whether it can share the sensor nodes from other BSNs, and inform the Buddy Management module of the share request, which is then add to the listen list and sent out in the management message. When the message is received by aggregators in other buddy BSNs, each aggregator checks the listen list, finds out the share request for its local nodes, and sends it to the application layer to get sharing decisions. If the application layer approves the sharing of the sensor node,

the Buddy Management module adds it to the share list to be sent out in the management message, answering the share request.

When the Buddy Management module finds a buddy BSN left the neighborhood, it goes through its share list and listen list to remove the sharing relations with that BSN.

5.5 Performance Evaluation

We evaluate our proposed framework using trace-driven simulations. The traces of noise and signal strength used in the simulation are collected when sensor nodes wearing on human bodies communicate with aggregators. The traces are collected in a lab environment, where two people wearing two BSNs sit almost back to back, about 2 meters away from each other. The sensor nodes are attached to three positions on the human body: the left chest, the right wrist, and the right ankle. The aggregator is attached to the left waist. The RSSI readings for the noise and signal strength are recorded for 5 minutes for each sensor node communicating with each aggregator respectively. The noise traces are used to generate noise models following the Closest Pattern Matching (CPM) algorithm [35] for each node and each aggregator. The signal strength traces are used directly in the simulation.

We implement BuddyQoS in TinyOS 2.x with NesC, and simulations are run under the TOSSIM simulator [36]. We compare our solution with the default CSMA solution in the standard TinyOS 2.x release. Three performance metrics are used: i) the percentage of delivered throughput, which equals the delivered data throughput over the requested data throughput; ii) the control overhead, which equals the number of control packets, including SNACKs and management messages over the number of data packets requested to send;

Table 5.1: System Parameter Configuration

| Parameter | Value |
|--|-----------------|
| Backoff range $[T_{backMin}, T_{backMax}]$ | [0.3ms, 2.44ms] |
| Slot length | 5ms |
| Interval T | 1000ms |
| Initial T_m value T_{init} | 5 Slots |
| Number of empty slots in T_d ΔT | 3 Slots |
| Data packet payload size S_{pkt} | 32 Bytes |
| Throughput requirement b_i for each node i | 1.2kbps |
| Decay factor α | 0.5 |

iii) the data packet transmission time, the time used to transmit a data packet, including retransmission and excluding queuing time.

Table 5.1 lists the system parameter configurations used in the simulations. The first three parameters are used by the Hybrid MAC, and we borrow some settings in the default CSMA in TOSSIM when we choose the values for them. In the default CSMA in TOSSIM, the initial backoff range is set as [0.3ms, 9.78ms], after that, the backoff range is set as [0.3ms, 2.44ms]. As our Hybrid MAC has already used TDMA schedule to avoid most of the collisions, we use the shorter backoff range to make the data transmission more efficient. $T_{backMin}$ is set as a non-zero value to tolerate time synchronization errors, and in our simulation, we deliberately introduced ≤ 0.1 ms time difference for each node's clock to simulate the synchronization errors. So we set the lower bound of the backoff range to be 0.3ms. Then to decide the slot length, we add up the maximum backoff, the preamble, and the time used to send one packet with 32 bytes data payload. We assume that the size of neighborhood is not very big, considering that $T_{init}=5$ is enough for a family with two parents and several children. Also, we assume that it's not very likely that a lot of people come to the family at the same time, so we set $\Delta T=3$. The decay factor α is set

to .5, considering the history and the current measurement of equally importance when computing the moving average of the packet lose rate.

We first compare the performance of our solution with the default best effort solution using CSMA in TinyOS. Then BuddyQoS is evaluated with an increasing number of buddy BSNs in the neighborhood. With the noise and signal strength traces collected from a real environment, we increase the number of neighboring buddy BSNs from 2 to 4. Same as the trace collection setup, each BSN contains one aggregator and three sensor nodes, and each sharing a sensor node from another BSN. Each aggregator and node can hear all other aggregators and nodes in the neighboring BSNs. The shared nodes are not on the same position on the body.

5.5.1 Performance Comparison with Default CSMA Release in TinyOS

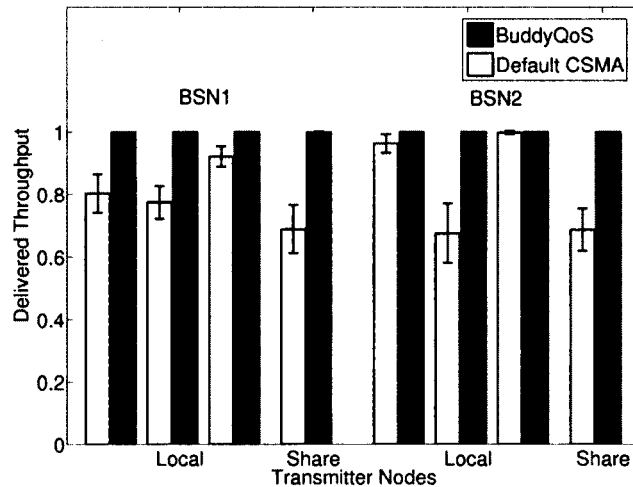


Figure 5.5: Delivered Throughput

First, we measure the percentage of delivered throughput for BuddyQoS and the best

effort solution with two neighboring buddy BSNs, and the results are shown in Figure. 5.5. For each node under BuddyQoS and the default best effort solution, a group of two bars shows the average throughput delivery percentages for the two solutions respectively, and the errorbars show their standard derivation. For each solution, the simulation are run for 5 minutes. Four groups are plotted for each BSN, with the first 3 groups illustrating the results for the 3 nodes in the local BSN, and the last group showing the results for the node shared from another buddy BSN. The results proves that our solution effectively achieves 100% throughput delivery, while the default best effort solution only delivers a portion of the required throughput. On some of the nodes, especially the shared nodes, the delivered throughput can be as bad as only about 70% of the required throughput.

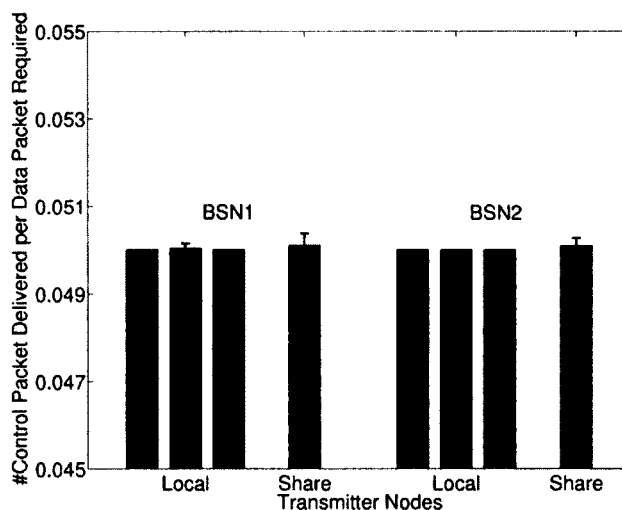


Figure 5.6: Control Overhead

Then we measure the control overhead of BuddyQoS. In BuddyQoS, the control overhead generated by the periodically broadcasted management messages and SNACKs. As one management message is sent every interval for all the local nodes, as well as the node it

share from other BSNs (to maintain the sharing relation), the cost of management messages should be divided among these nodes. In our evaluation setup, one management messages are used by 4 nodes (3 local nodes plus 1 shared node). Then during each interval, for each node we add the number of SNACKs sent to it and the 1/4 share of the management message, and the result is divided by the number of delivered data packets to compute the control overhead. Figure 5.6 plots the control overhead for each node from 2 neighboring buddy BSNs. The results of the mean control overhead are very low, only around 0.05 for each nodes, with very small standard derivation. Also, we can conclude that the cost of SNACKs is minor, and the control overhead is mainly caused by the management messages. This proves that the use of SNACK and SNACK suppression is efficient.

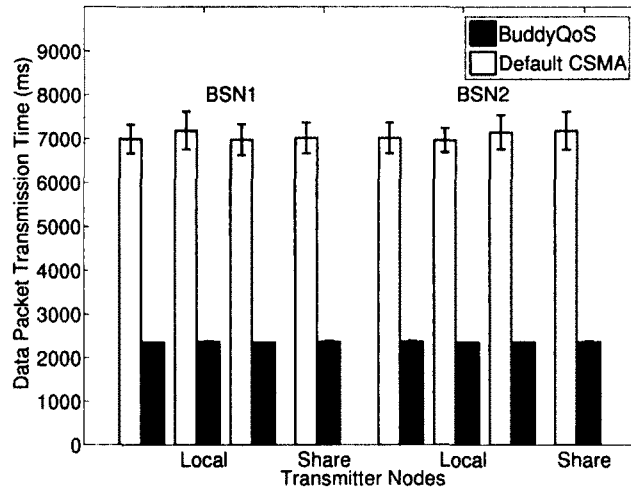


Figure 5.7: Data Packet Transmission Time

Finally, we calculate the time used to transmit a data packet, excluding the queuing delay. The data packet transmission time under BuddyQoS is compared with the results from the default system in Figure 5.7. We observe that BuddyQoS uses much less time to

send a data packet than the default CSMA. Only 1/3 of the time is used by BuddyQoS, and this shows that our solution is more efficient. BuddyQoS uses less time mainly because it uses a shorter backoff than the initial backoff of the default solution.

5.5.2 Performance Results with an Increasing Number of Neighboring BSNs

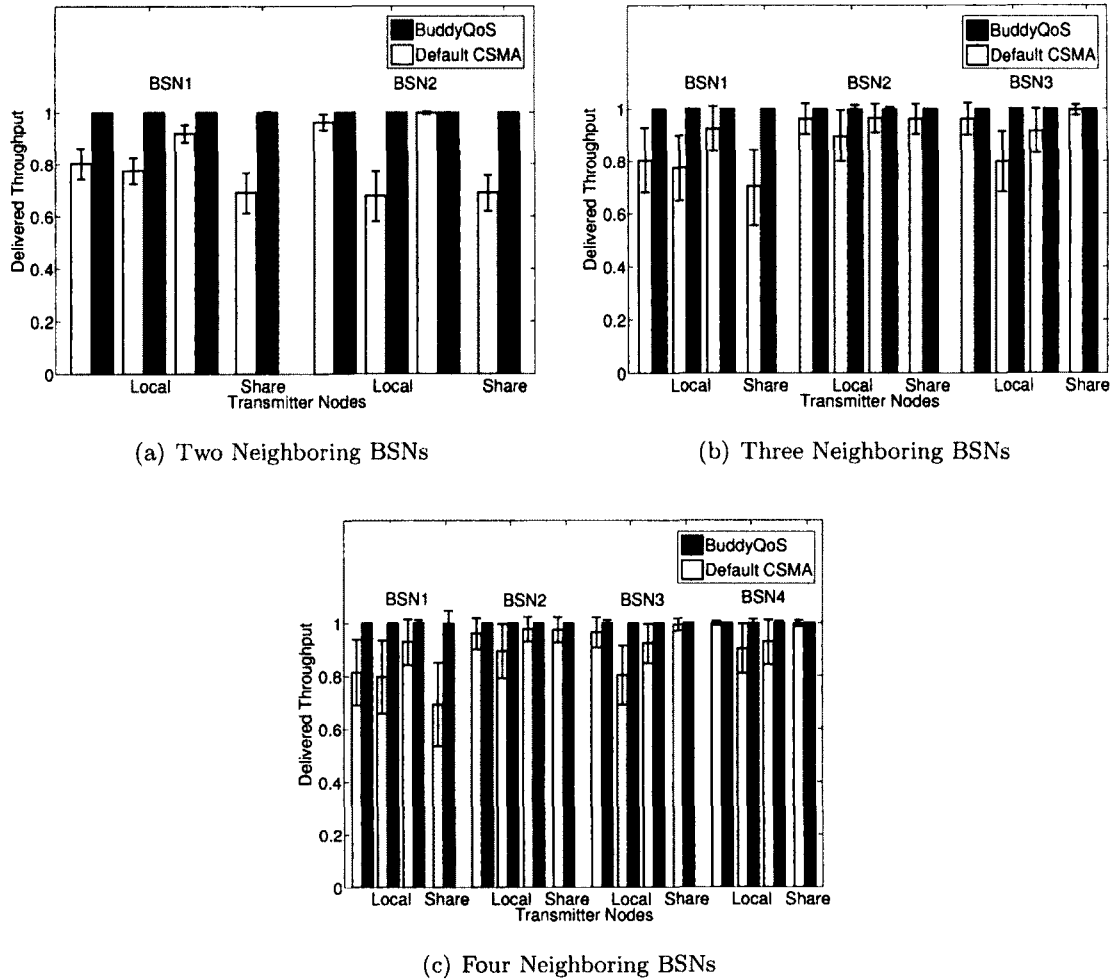


Figure 5.8: Percentage of Delivered Throughput with Increasing Number of BSNs

Figure 5.8 plots the percentage of delivered throughput with an increasing number of

buddy BSNs in the neighborhood. For each solution, the simulation runs for 5 minutes, and the percentage of delivered throughput is measured every 10 seconds. Then we plot the average and standard derivation of the measured results for each node of each BSN (3 local nodes and 1 node shared from another BSN).

Comparing our solution with the default best effort solution, we find that the best effort solution is not able to provide the data throughput assurance. The percentage of delivered throughput under the best effort solution varies with different nodes. This is because the packet losses are mainly due to the signal attenuation when going through human bodies rather than congestion. For some of the links, only about 70% of the data can be delivered. The standard derivation of the delivered throughput is also large (nearly 10%), which shows that the links are very unstable. However, BuddyQoS achieves the QoS goal required by the application for every node when the number of buddy BSNs in the neighborhood increases. 100% of the required throughput are delivered with small standard derivations.

5.6 Conclusion

Multiple BSNs can often coexist in the communication range of each other, when a person wearing a BSN spend time with families, friends and colleagues. In these multiple coexisting BSNs, applications can benefit from sensor data sharing. This paper proposes a novel QoS solution, BuddyQoS, to provide throughput assurance for both inter- and intra-communications in multiple BSNs. Our solution is able to perform resource estimation and allocation across multiple BSNs. Both the management of information exchange and the data delivery are properly scheduled, and with a flexible Hybrid MAC design, the sched-

ules can be effectively enforced. Through a trace-driven simulation, we demonstrate that BuddyQoS outperforms the default best effort solution, with 100% throughput delivery and minor control overhead. In our future work, we will study how to provide both throughput and time delay assurance for multiple coexisting BSNs.

CHAPTER 6

CONCLUSION

The BSN deployment benefits greatly from the wireless communication technology, which makes it possible to install wearable and implantable bio-sensors on human body without constraining normal activities. The ubiquitous applications developed based on the BSN have stringent performance requirements, and they rely on the lower level network to provide certain QoS assurance for the data delivery. However, the low power sensor communication link quality can be easily degraded in the noisy environment. In addition, comparing with the conventional WSN, the distinct characteristics of the BSN [13] require new QoS solutions.

In this dissertation, we develop confident body sensor networking solutions with three parts of works: 1) At the communication level, considering the existing heterogeneous BSN platform, we aim to provide a radio-agnostic QoS solution for the data delivery service

within a BSN. Without assumptions for the underlying MAC implementation, we study the resource scheduling problem in a BSN with both throughput and time delay requirements for all data transmissions in a BSN. 2) At the application level, we aim to connect the communication QoS with the application QoS. As different applications have different requirements, we study how the application fidelity can be affected by the amount of data transmitted to the local station. Then, we develop algorithms to improve the system performance with limited resource. 3) When multiple BSNs coexist and share sensor data with each other, we aim to provide QoS solutions for both inter- and intra-BSN communications. In particular, we expose the communication QoS availability across BSNs.

We address the above challenges through following works:

- *Building Communication QoS.* We propose a novel radio-agnostic approach, BodyT2, to provide joint throughput and time delay assurance for the communication in a BSN. Using the virtual MAC abstraction, we define the resource scheduling problem for the communication in a BSN with throughput and time delay requirements. We theoretically analyze the complexity of this problem and propose a practical solution based on a group-polling scheme. We implement the BodyT2 framework and demonstrate its performance through both TelosB mote lab tests and real on-body experiments.
- *Connecting Communication QoS with Application QoS.* We take neurometric applications as examples, and study the realistic communication pattern of an off-the-shelf commercial wireless EEG headset. We find that the current lossy communication pattern of the EEG headset does not match the high neurometric application fidelity requirements. We define the application fidelity for the two example neurometric ap-

application QoS. We propose to automatically learn the sensitivity of application fidelities to sensory data. With the learned sensitivity, we develop two algorithms to enhance the system performance when the resource is limited. We validate the effectiveness of the proposed solution through trace-based experiments.

- *Exposing Communication QoS Availability to Other BSNs.* We propose a novel QoS solution, BuddyQoS, to provide throughput assurance for both inter- and intra-communications in multiple coexisting and shared buddy BSNs. Our solution is able to perform resource estimation and allocation across multiple buddy BSNs. Both the management information exchange and the data delivery are properly scheduled. A communication paradigm is used to support throughput assurance for the data multicast in shared BSNs, and a specific hybrid MAC design is used to effectively enforce the communication schedules. Through a trace-driven simulation, we demonstrate that BuddyQoS outperforms the default CSMA release in TinyOS, with 100% throughput delivery and minor control overhead. In our future work, we will study how to provide both throughput and time delay assurance for multiple coexisting and shared buddy BSNs.

Overall, we have developed different QoS solutions to provide the confident BSN communication services required from both the communication and application levels, in single and multiple BSNs. In our future work, we first plan to develop QoS solutions that optimize other network metrics, like data resolution and time delay in multiple coexisting BSNs. Taking the heterogeneous radio platforms in BSNs into consideration, we will provide the radio agnostic feature, and will implement and experiment with different standards, such as

ZigBee, Bluetooth, and WiFi. Also, regarding the mobility of BSNs, we plan to study the impact of different environments and human activities on the BSN communication quality. With multiple BSNs, we will develop a lightweight real-time measurement tool for studying the packet loss correlation between a transmitter and multiple receivers. We plan to investigate the communication models not only in BSNs, but also between BSNs and static networks. We will utilize the knowledge of the environment and human activity to build context-aware QoS solutions for BSN communications. Lastly, we are interested in the different fidelity requirements of different applications in the BSN. We plan to study on more time sensitive applications, especially medical applications with specific fidelity requirements, and we will provide generic approaches that automatically adapt resource scheduling to meet those requirements.

BIBLIOGRAPHY

- [1] T. F. ABDELZAHER, S. PRABH, AND R. KIRAN. On Real-time Capacity Limits of Multihop Wireless Sensor Networks. In *IEEE RTSS*, 2004.
- [2] K. ANSARI-ASL, G. CHANEL, AND T. PUN. A Channel Selection Method for EEG Classification in Emotion Assessment Based on Synchronization Likelihood. In *EU-SIPCO*, 2007.
- [3] L. K. AU, W. H. WU, M. A. BATALIN, D. H. MCLNTIRE, AND W. J. KAISER. Microleap: Energy-Aware Wireless Sensor Platform for Biomedical Sensing Applications. In *IEEE BIOCAS*, 2007.
- [4] K. C. BARR AND K. ASANOVIĆ. Energy-Aware Lossless Data Compression. In *ACM TOCS*, 2006.
- [5] M. BARRY, A. T. CAMPBELL, AND A. VERES. Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks. In *IEEE INFOCOM*, 2001.
- [6] A. T. BARTH, M. A. HANSON, H. C. POWELL JR, AND J. LACH. Online Data and Execution Profiling for Dynamic Energy-Fidelity Optimization in Body Sensor Networks. In *IEEE BSN*, 2010.
- [7] M. BÄCHLIN, K. FÖRSTER, AND G. TRÖSTER. SwimMaster: A Wearable Assistant for Swimmer. In *ACM UbiComp*, 2009.
- [8] T. J. DE BOCK, S. DAS, M. MOHSIN, N. B. MUNRO, L. M. HIVELY, Y. JIANG, C. D. SMITH, D. R. WEKSTEIN, G. A. JICHA, A. LAWSON, J. LIANEKHAMMY, E. WALSH, S. KISER, AND C. BLACK. Early detection of Alzheimer's disease using nonlinear analysis of EEG via Tsallis entropy. In *IEEE BSEC*, 2010.
- [9] S.P. BOYD AND L. VANDENBERGHE. *Convex Optimization*. Cambridge University Press, 2004.
- [10] D. G. CACUCI. *Sensitivity and Uncertainty Analysis: Theory, Volume I*. CRC, 2003.

- [11] D. G. CACUCI, M. IONESCU-BUJOR, AND I. M. NAVON. *Sensitivity and Uncertainty Analysis: Applications to Large-Scale Systems, Volume I*. CRC, 2005.
- [12] A. T. CAMPBELL, T. CHOUDHURY, S. HU, H. LU, M. K. MUKERJEE, M. RABBI, AND R. D. S. RAIZADA. Neuro-Phone: Brain-Mobile Phone Interface using a Wireless EEG Headset. In *ACM MobiHeld*, 2010.
- [13] H. CAO, V. LEUNG, C. CHOW, AND H. CHAN. Enabling Technologies for Wireless Body Area Networks: A Survey and Outlook. *IEEE Communications Magazine*, 2009.
- [14] K. K. CHINTALAPUDI AND L. VENKATRAMAN. On the Design of MAC Protocols for Low-Latency Hard Real-Time Discrete Control Applications over 802.15. 4 Hardware. In *ACM/IEEE IPSN*, 2008.
- [15] S. B. EISENMAN, N. D. LANE, AND A. T. CAMPBELL. Techniques for Improving Opportunistic Sensor Networking Performance. In *IEEE DCOSS*, 2008.
- [16] S. B. EISENMAN, E. MILUZZO, N. D. LANE, R. A. PETERSON, G. S. AHN, AND A. T. CAMPBELL. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In *ACM SenSys*, 2007.
- [17] Emotiv EPOC Neuroheadset. <http://www.emotiv.com/>.
- [18] A. ESWARAN, A. ROWE, AND R. RAJKUMAR. Nano-RK: An Energy-aware Resource-centric RTOS for Sensor Networks. In *IEEE RTSS*, 2005.
- [19] K. FALL AND S. FLOYD. Simulation-Based Comparisons of Tahoe, Reno and SACK TCP. *ACM C2R*, 1996.
- [20] E. FARELLA, A. PIERACCI, L. BENINI, L. ROCCHI, AND A. ACQUAVIVA. Interfacing Human and Computer with Wireless Body Area Sensor Networks: the WiMoCA Solution. *Multimedia Tools and Applications*, 2008.
- [21] T. GAO, T. MASSEY, L. SELAVO, D. CRAWFORD, B. CHEN, K. LORINCZ, V. SHNAYDER, L. HAUENSTEIN, F. DABIRI, J. JENG, A. CHANMUGAM, D. WHITE, M. SARRAFZADEH, AND M. WELSH. The Advanced Health and Disaster Aid Network: A Light-weight Wireless Medical System for Triage. *IEEE Transactions on Biomedical Circuits and Systems*, 2007.
- [22] T. GAO, C. PESTO, L. SELAVO, Y. CHEN, J. KO, J. LIM, A. TERZIS, A. WATT, J. JENG, B. R. CHEN, K. LORINCZ, AND M. WELSH. Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results. In *IEEE HST*, 2008.
- [23] L. GU AND J. A. STANKOVIC. *t - kernel*: Providing Reliable OS Support to Wireless Sensor Networks. In *ACM SenSys*, 2006.
- [24] M.A. HANSON, H.C. POWELL, A.T. BARTH, AND J. LACH. Enabling data-Centric Energy-Fidelity Scalability in Wireless Body Area Sensor Networks. In *BodyNets*, 2009.

- [25] CodeBlue. <http://www.eecs.harvard.edu/mdw/proj/codeblue/>.
- [26] T. HE, J. A. STANKOVIC, C. LU, AND T. F. ABDELZAHER. SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks. In *IEEE ICDCS*, 2003.
- [27] J.B. HENRIQUES AND R. J. DAVIDSON. Left Frontal Hypoactivation in Depression. *Journal of Abnormal Psychology*, 1991.
- [28] R. J. HONICKY. Understanding and Using Rendezvous to Enhance Mobile Crowdsourcing Applications. *Computer*, 2011.
- [29] I-H. HOU AND P. R. KUMAR. Admission Control and Scheduling for QoS Guarantees for Variable-bit-rate Applications on Wireless Channels. In *ACM MobiHoc*, 2009.
- [30] C. HU, H. KIM, J. C. HOU, D. CHI, AND S. SHANKAR N. Provisioning Quality Controlled Medium Access in UltraWideBand WPANs. In *IEEE INFOCOM*, 2006.
- [31] H. HUANG, Y. SUN, Q. YANG, F. ZHANG, X. ZHANG, Y. LIU, J. REN, AND F. SIERRA. Integrating Neuromuscular and Cyber Systems for Neural Control of Artificial Legs. In *ACM/IEEE ICCPS*, 2010.
- [32] V. K. INGLE AND J. G. PROAKIS. *Digital Signal Processing using MATLAB*. Cengage Learning, 2011.
- [33] I. JANTUNEN, H. LAINE, P. HUUSKONEN, D. TROSSEN, AND V. ERMOLOV. Smart Sensor Architecture for Mobile-terminal-centric Ambient Intelligence. *Sensors and Actuators A: Physical*, 2008.
- [34] K. LEE, S. HONG, S. J. KIM, I. RHEE, AND S. CHONG. Slaw: A New Mobility Model for Human Walks. In *IEEE INFOCOM*, 2009.
- [35] K. LEE, Y. KIM, S. CHONG, I. RHEE, AND Y. YI. Delay-Capacity Tradeoffs for Mobile Networks with Lévy Walks and Lévy Flights. In *IEEE INFOCOM*, 2011.
- [36] P. LEVIS, N. LEE, M. WELSH, AND D. CULLER. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *ACM SenSys*, 2003.
- [37] Q. LI, J. A. STANKOVIC, M. HANSON, A. BARTH, J. LACH, AND G. ZHOU. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. In *BSN*, 2009.
- [38] N. LITTLESTONE AND M. K. WARMUTH. The Weighted Majority Algorithm. In *IEEE FOCS*, 2002.
- [39] K. LORINCZ, B. R. CHEN, J. WATERMAN, G. WERNER-ALLEN, AND M. WELSH. Pixie: An Operating System for Resource-aware Programming of Embedded Sensors. In *IEEE EmNets*, 2008.
- [40] C. LU, B. M. BLUM, T. F. ABDELZAHER, J. A. STANKOVIC, AND T. HE. Rap: A Real-time Communication Architecture for Large-scale Wireless Sensor Networks. In *RTAS*, 2002.

- [41] H. LU, N. D. LANE, S. B. EISENMAN, AND A. T. CAMPBELL. Bubble-sensing: Binding sensing tasks to the physical world. *Pervasive and Mobile Computing*, 2010.
- [42] R. MANGHARAM, A. ROWE, R. RAJKUMAR, AND R. SUZUKI. Voice Over Sensor Networks. In *IEEE RTSS*, 2006.
- [43] M. MIKHAIL, K. EL-AYAT, R. EL KALIOUBY, J. COAN, AND J. J. B. ALLEN. Emotion detection using noisy EEG data. In *ACM AH*, 2010.
- [44] A. MILENKOVIC, C. OTTO, AND E. JOVANOVIĆ. Wireless Sensor Networks for Personal Health Monitoring: Issues and an Implementation. *Computer Communications*, 2006.
- [45] E. MILUZZO, N. D. LANE, A. T. CAMPBELL, AND R. OLFATI-SABER. CaliBree: A Self-calibration System for Mobile Sensor Networks. In *IEEE DCOSS*, 2008.
- [46] M. NABI, M. BLAGOJEVIC, M. GEILEN, T. BASTEN, AND T. HENDRIKS. MCMAC: An Optimized Medium Access Control Protocol for Mobile Clusters in Wireless Sensor Networks. In *IEEE SECON*, 2010.
- [47] V. NAMBOODIRI AND A. KESHAVARZIAN. Alert: An Adaptive Low-latency Event-driven MAC Protocol for Wireless Sensor Networks. In *ACM/IEEE IPSN*, 2008.
- [48] D. NARAYANAN AND M. SATYANARAYANAN. Predictive Resource Management for Wearable Computing. In *ACM MobiSys*, 2003.
- [49] A. NATARAJAN, B. D. SILVA, K. K. YAP, AND M. MOTANI. Link Layer Behavior of Body Area Networks at 2.4 GHz. In *ACM MobiCom*, 2009.
- [50] M. J. NEELY. Delay Analysis for Maximal Scheduling in Wireless Networks with Bursty Traffic. In *IEEE INFOCOM*, 2008.
- [51] A. S. PENTLAND. Healthwear: Medical Technology Becomes Wearable. *Computer*, 2004.
- [52] A. PYLES AND G. ZHOU. Implementing USB Host Mode on the Android Platform. In *College of William and Mary Technical Report WM-CS-2010-06*, 2010.
- [53] V. RAGHUNATHAN, V. BORKAR, M. CAO, AND P. R. KUMAR. Index Policies for Real-time Multicast Scheduling for Wireless Broadcast Systems. In *IEEE INFOCOM*, 2008.
- [54] B. RAJAGOPALAN. Reliability and scaling issues in multicast communication. In *ACM SIGCOMM Computer Communication Review*, 1992.
- [55] Z. REN, G. ZHOU, A. PYLES, M. KEALLY, W. MAO, AND H. WANG. BodyT2: Throughput and Time Delay Performance Assurance for Heterogeneous BSNs. In *IEEE INFOCOM*, 2011.
- [56] I. RHEE, M. SHIN, S. HONG, K. LEE, AND S. CHONG. On the Levy-walk Nature of Human Mobility: Do Humans Walk like Monkeys? In *IEEE INFOCOM*, 2008.

- [57] I. RHEE, A. WARRIER, M. AIA, J. MIN, AND M. L. SICHITIU. Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 2008.
- [58] A. ROWE, R. MANGHARAM, AND R. RAJKUMAR. RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks. In *IEEE SECON*, 2006.
- [59] J. A. RUIZ AND S. SHIMAMOTO. Novel Communication Services Based on Human Body and Environment Interaction: Applications Inside Trains and Applications for Handicapped People. In *IEEE WCNC*, 2006.
- [60] R. C. SHAH, L. NACHMAN, AND C.-Y. WAN. On the Performance of Bluetooth and IEEE 802.15.4 Radios in a Body Area Network. In *BodyNets*, 2008.
- [61] E. I. SHIH AND J. J. GUTTAG. Reducing Energy Consumption of Multi-Channel Mobile Medical Monitoring Algorithms. In *ACM HealthNet*, 2008.
- [62] E. I. SHIH, A. H. SHOEB, AND J. V. GUTTAG. Sensor Selection for Energy-Efficient Ambulatory Medical Monitoring. In *ACM MobiSys*, 2009.
- [63] A. SHOEB AND J. GUTTAG. Application of Machine Learning To Epileptic Seizure Detection. In *ICML*, 2010.
- [64] S. SRINIVASA AND S. KRISHNAMURTHY. CREST: an Opportunistic Forwarding Protocol Based on Conditional Residual Time. In *IEEE SECON*, 2009.
- [65] F. STANN AND J. HEIDEMANN. RMST: Reliable Data Transport in Sensor Networks. In *The First International Workshop on Sensor Network Protocols and Applications*, 2003.
- [66] M. VERLOOP AND S. BORST. Heavy-Traffic Delay Minimization in Bandwidth-Sharing Networks. In *IEEE INFOCOM*, 2007.
- [67] N. VERMA, A. SHOEB, J. V. GUTTAG, AND A. P. CHANDRAKASAN. A Micro-power EEG Acquisition SoC with Integrated Seizure Detection Processor for Continuous Patient Monitoring. In *IEEE Symposium on VLSI Circuits*, 2009.
- [68] Y. WU, G. ZHOU, AND J. A. STANKOVIC. ACR: Active Collision Recovery in Dense Wireless Sensor Networks. In *IEEE INFOCOM*, 2010.
- [69] Y. YANG AND R. KRAVETS. Achieving Delay Guarantees in Ad Hoc Networks Through Dynamic Contention Window Adaptation. In *IEEE INFOCOM*, 2006.
- [70] G. ZHOU, T. HE, S. KRISHNAMURTHY, AND J. A. STANKOVIC. Models and Solutions for Radio Irregularity in Wireless Sensor Networks. In *ACM TOSN*, 2006.
- [71] G. ZHOU, C. HUANG, T. YAN, T. HE, J. A. STANKOVIC, AND T. F. ABDELZAHER. MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks. In *IEEE INFOCOM*, 2006.

- [72] G. ZHOU, J. LU, C.-Y. WAN, M. D. YARVIS, AND J. A. STANKOVIC. BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks. In *IEEE INFOCOM*, 2008.
- [73] G. ZHOU, J. A. STANKOVIC, AND S. H. SON. Crowded Spectrum in Wireless Sensor Networks. In *IEEE EmNets*, 2006.

VITA

Zhen Ren

Zhen Ren received her Bachelor of Science degree in Computer Science for Nanjing University, China, in 2003. From August 2006, she began perusing her Doctor of Philosophy degree under the supervision of Dr. Gang Zhou and Dr. Haining Wang. Her research interests are on the area of wireless communication and networking, sensor networks, ubiquitous computing, Quality of Service (QoS), Voice over Internet Protocol (VoIP).