Multi-Body, Multi-Function Body Sensor Networks

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Qiang Li

May 2017

Abstract

With the cost of health care increasing, Body Sensor Networks (BSNs) have been proposed to provide low-cost health care solutions for the senior population. Reliability is essential for BSNs because they are mainly used for medical purposes such as detecting health-detrimental accidents and monitoring health status. The reliability of BSNs is twofold: reliable communication and reliable applications. The former focuses on data collection, while the latter focuses on data processing. Though existing work studies some of the reliability issues, the state of art lacks systematic approaches to implement reliable BSNs, especially when multiple BSNs coexist.

Therefore, in this dissertation we systematically study how to develop reliable multi-body, multi-function BSNs. First, we perform an empirical study to investigate the challenges for developing reliable BSNs. Then, we propose a QoS framework that guarantees reliable communication and fidelity of BSN applications. Reliable communication is achieved by dynamically grouping nearby BSNs into one group and using different frequency channels and time sharing to schedule transmission within each group. Fidelity of BSNs is guaranteed by profiling BSN platforms and applications, and finding the optimal configuration to accommodate multiple BSN systems. We also develop two representative BSN applications, fall detection (accident detection) and social activity detection (health monitoring), to study how to develop reliable BSN applications. At last, we build a BSN platform to unify our proposed QoS framework with both fall detection and in-person interaction monitoring applications.

The evaluation demonstrates the effectiveness of our proposed methods for developing reliable multi-body, multi-function BSNs. BSN systems implemented using our framework achieves over 97% overall Packet Reception Ratio (PRR) even when multiple BSNs are within the interference range of each other, while PRR of systems not considering nearby BSNs drops to below 50%. Profiling BSNs and automatically switching between BSN settings guarantee data fidelity in terms of sample frequency and sample delivery. Our fall detection application detects over 96% of falls and our in-person interaction monitoring application achieves over 82% accuracy.

Approval Sheet

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

Doctor of Philosophy (Computer Science)

Qiang Li

This dissertation has been read and approved by the Examining Committee:

John Stankovic, Adviser

Kamin Whitehouse, Committee Chair

Jack Davidson

Gabriel Robins

John Lach, Minor Representative

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

May 2017

To my mother Yunxia Wang and my father Dianwen Li for their love and support.

Acknowledgments

Foremost, I want to express my sincere gratitude to my adviser Professor John Stankovic for his patience, support, and guidance. During the past few years, there were numerous times that I felt so frustrated about the progress of finishing the dissertation, but John has always been there encouraging me and pushing me forward. John is my role model both as a distinguished researcher in academia and as a compassionate person in life.

I would also like to thank my committee members, Professor Kamin Whitehouse, Professor John Lach, Professor Gabriel Robins, and Professor Jack Davidson for their guidance and invaluable comments on this dissertation.

I am very fortunate and honored to work with outstanding colleagues from whom I received a lot of help. I would like to take this opportunity to thank them all: Zhiheng Xie, Yafeng Wu, Adam T. Barth, Mark A. Hanson, Shanshan Chen, Gang Zhou, Jingyuan Li, Shahriar Nirjon, Robert F. Dickerson, and Shuai Che.

Last but not least, I want to thank my family. The unconditional love from my parents and my sister has always been the momentum to push me forward. Without the understanding and support of my wife Ting-Yao Huang, I would not have been able to complete this dissertation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Challenges

According to the U.S. Department of Health and Human Services [1], the population of 65 and older reached 46.2 million in 2014. They represented 14.5% of the U.S. population. By 2040, older population is projected to be 82.3 million, 21.7% of the population. People older than 85 is projected to increase from 6.2 million in 2014 to 14.6 million in 2040. At the same time, the healthcare costs for older people are also increasing: in 2014 older people averaged out-of-pocket health care expenditures of $5,849, an increase of 50% since 2004. Therefore, during the past few years, Body Sensor Networks (BSNs) have been proposed as a low-cost health care solution for this large population.

The main functionality of BSNs includes accident detection (such as fall detection) and health status monitoring (such as electrocardiography, ECG). In both cases reliable solutions are essential: an accident detection application like heart attack detection needs to detect accidents accurately and send the emergency alarm in real time to healthcare professionals for help; a health monitoring application like ECG needs to log a large amount of physiological data continuously with acceptable fidelity.

However, there are many challenges to implement reliable healthcare systems using BSNs:

- Body shadowing effect. With motes mounted on the body, BSNs suffer from radio attenuation caused by the impermeability of the human body. In addition, when people perform different activities, the body shadowing effect changes between two positions on the body, which results in highly variable link conditions.

- Variable operation environment. People wearing BSNs move across different environments, e.g. people may move from an indoor home to an outdoor open area. The radio reflection changes across different environments (such as indoor vs. outdoor), and thus the variable operation environment has great impacts on BSN communications.

- Power constraint. Since the motes in BSNs are worn by people, their size must be small enough for comfortable wearing, thus the batteries of BSN motes are limited in size and capacity. At the same time, the lifetime of the system should last as long as possible. Therefore, how to reduce power consumption to prolong system lifetime is also a major challenge for BSNs.

- Two scalablility problems. Most existing work only focuses on a single BSN. However, in a real world deployment, there can be a number of BSNs working at the same time. For example, in a hospital or an assisted-living facility, every patient or older person can wear a BSN, and there will be interference between these BSNs. Therefore, how to coordinate multiple BSNs is an critical problem.

  Moreover, future BSNs can have multiple applications running on them. For example, a BSN can perform fall event detection and daily activity monitoring at the same time. BSNs with multiple functions feature heavier traffic load, and needs a way to coordinate the traffic of each function.

- Developing reliable BSN applications. Most BSN applications are used for medical purposes which requires high-level reliability. However, because of challenges of BSNs and the difficulty to collect real data for medical events such as falls, it is difficult to develop reliable BSN applications.

Though there is existing work trying to address some of the challenges listed above, the state of art lacks systematic approaches to implement reliable BSN systems, especially when multiple BSNs coexist.

1.2 Contributions

To tackle the challenges, this dissertation focuses on developing reliable *multi-body*, *multi-function* BSNs. *Multi-body* means that the BSN systems are deployed onto multiple persons, and BSNs worn by these people can be present in the same vicinity at the same time. Because of the mobility of BSNs, the distance and interference between BSNs vary over time. *Multi-function* means that

each BSN system worn by one single person can perform more than one function. For example, a BSN system can detect falls and monitor heart rates at the same time. The data traffic of different functions has different data rates and different fidelity requirements. When multiple BSNs, each of which hosts multiple applications, coexist, the traffic needs to be controlled and scheduled to guarantee the delivery and fidelity of data.

The research in the dissertation is divided into four parts. First, an empirical study was carried out to investigate challenges for reliable BSN communications. To tackle the challenges, we then studied the design and implementation of a Quality of Service (QoS) solution for multi-body, multi-function BSNs. After this, we designed and implemented two representative BSN applications: fall detection and in-person interaction monitoring. At last, we unified the QoS framework and the two BSN applications to implement a reliable multi-body, multi-function BSN system.

The main contributions of the dissertation are listed below:

- We designed and implemented a QoS solution for multi-body, multi-function BSN systems that guarantees not only reliable communications between BSN nodes but also data fidelity for BSN applications. Our solution coordinates the traffic of nearby BSNs using channel assignment and time sharing to reduce packet collisions and thus lower power consumption used for retransmission. To tackle the body shadowing effect with minimum energy cost, we used a power adaptation scheme to automatically adapt transmission power according to link conditions. The data fidelity requirements for BSN applications are considered in our QoS solution when scheduling transmissions. QoS solutions across multiple BSNs and multiple applications have not been studied before.

- We proposed a profiling approach to separate BSN hardware platform details and BSN application requirements. Nowadays, BSN hardware and software are indivisible: a BSN system is developed to host one specific application such as fall detection, while an application is at the same time confined to a specific BSN platform because it is developed to work with data collected from specific sensors, in specific rates, etc. This intertwined approach dramatically increases the difficulty to develop BSN systems, because it requires expertise of both hardware and software. In the dissertation, BSN hardware characteristics and application data requirements are described using profiles, and we provide algorithms to automatically coordinates the data collection from sensors and the data processing by the applications. In this way, developing platform independent BSN applications become possible. This separation has not been studied before.

- We implemented a grammar-based, posture- and context-cognitive fall detection application which not only better distinguishes fall-like activities from real falls, but also emphasizes the detection of slow falls. We utilize posture information extracted from on-body sensors and context information collected from sensors deployed in the house to reduce false positives. A fall in our framework is detected as a sequence of sensor events. We provide a context-free grammar to define these sequences so that the framework can be easily extended to detect more kinds of falls. Our case study shows that our method can distinguish various fall-like activities from real falls and can also effectively detect both fast falls and slow falls. The integration evaluation shows that our method achieves both high sensitivity and high specificity.

- By using state-of-art smartphones and on-body sensors, we implemented a multi-modal system that collects a battery of features to monitor in-person interactions. Unlike existing work that monitors interactions only based on data collected from one person, we emphasize that in-person interactions intrinsically involve multiple participants, and thus we aggregate information from nearby people to identify more interaction details. In addition, unlike most existing work our in-person interaction monitoring solution does not require any in-situ infrastructure. Evaluation shows our solution accurately detects various in-person interactions and provides insights absent in existing systems.

- A reliable BSN system requires both reliable communication (data collection) and reliable applications (data processing). In the dissertation, we unified our QoS solution and the two applications we developed onto one multi-body, multi-function BSN system. The unification of the QoS framework and two BSN applications demonstrate how our system matches BSN hardware details and application software requirements, and how the data transmission of multiple BSNs are coordinated so that data fidelity required by applications are guaranteed. This holistic approach to address reliability of BSNs are absent from existing work.

All the work listed above consists of a framework for developing reliable multi-body, multi-function BSNs in both communication level and application level. In the dissertation, we accomplished this framework, and a BSN system that detects falls and monitors in-person interactions using the framework.

1.3 Organization

In the dissertation, we first review related work in Chapter 2. Then an empirical study of BSN link characteristics is performed in Chapter 3 to identify various challenges of building reliable BSN systems. In Chapter 4, we proposed an BSN QoS framework that addresses how to achieve reliable communication when multiple BSNs coexist. Chapter 5 and Chapter 6 implemented two typical BSN applications, respectively: fall detection and in-person interaction monitoring. These two applications are not only important by themselves, but their data requirements such the sample frequency and maximum data update interval also serve as real examples of BSN application requirements in Chapter 7. The unification of the BSN QoS framework and two BSN applications is the main topic of Chapter 7. Chapter 8 concludes the dissertation and identifies potential future work.

Chapter 2

Related Work

With the cost of heath care increasing, BSNs are proposed to provide low-cost solutions for monitoring health status [2, 3] and detecting detrimental events [4, 5]. In both cases the reliability of BSNs is essential. Accident detection systems need to report dangerous events in time, so it requires real-time transmission. Health status monitoring systems usually require high data bandwidth because of the high sampling frequency of medical sensors, e.g. the ECG (Electrocardiography) sensor built by Park et al. [6] samples with a frequency of 1KHz. As a result, reliable communication, in terms of high Packet Reception Ratio (PRR), is of great importance for BSNs. This reliability is twofold: reliable communication and reliable applications. The former focuses on data collection, while the latter focuses on data processing. In this chapter, we first review literature on challenges to achieve reliable BSN communication, then discuss existing work on QoS (Quality of Service) solutions for BSN. The BSN framework implemented in the dissertation includes two representative BSN applications: fall detection and in-person interaction monitoring. Existing work on these are also reviewed.

2.1 Challenges to BSN Communication

BSNs are a specific type of wireless sensor networks with some distinguishing qualities: nodes of BSNs are attached onto human body, so the distance and shadowing between nodes change with body movements and postures; environmental factors, such as the interference from WiFi networks and radio reflections caused by obstacles, change frequently because of the locomotion of people wearing BSNs. These make communication characteristics of BSNs different from those of traditional wireless sensor networks. Despite that there are many existing work studying the communication of wireless sensor networks, the study of radio characteristics of BSNs is quite limited.

Rahul et al. [7, 8] explore the Packet Reception Ratio (PRR), goodput, and latency for different activities (sitting, standing, walking), and various sensor locations. However, they only use maximum sending power, which can dramatically shorten battery life of sensors and cause serious interference when multiple BSNs are close to each other.

Emiliano et al. [9] use Radio Signal Strength Indicator (RSSI), Link Quality Indicator (LQI), and throughput as metrics to show human body has a significant effect on BSN link qualities. Their results show that the body factor can reduce the throughput between a transmitter and a receiver to zero for some relative positions in an open soccer field. However, most BSNs operates indoors.

David et al. [10] exploit PRR under different combinations of antenna settings (removed, flattened and circled, normal), sensor locations, radio output power, and different environments (home, car, office). They use one node to broadcast messages to receiver nodes fixed at different on-body locations. This broadcast model eclipses the interference problem at the aggregator, because it does not comply with the traffic pattern of BSNs: nodes usually aggregate data to a single aggregator in BSNs, not vice versa.

Anirudh et al. [11] study the performance of different network architectures for BSNs in different environments. They show that a star architecture with nodes operating at low power levels suffice in a cluttered indoor environment, however, in an outdoor setting the sensor nodes need to operate at higher power levels or change to a multihop architecture to achieve acceptable PRR. Their later work [12] investigates the link layer behavior of BSNs. They collect data from 12 nodes in three different environments (office, residence, and open space) over a long period to reveal link layer characteristics. However, their experiments are not well controlled, which makes it difficult to understand how various factors such as body postures affect link qualities.

Wan et al. [13] evaluate the performance of IEEE 802.15.4 sensor networks in terms of packet delivery rate (PDR), latency, and energy consumption against traffic load and beacon tracking modes. Their study shows that nonbeacon-enabled mode provides lower latency and energy consumption, while beacon-enabled mode offers higher PDR. However, their performance study is based on simulation, not real world measurements. In addition, their study doesn't address special characteristics of BSNs such as body shadowing.

Wafa et al. [14] compare various broadcast strategies for BSNs. Their work shows that body postures and movements have great impact on packet reception ratio, thus dramatically affect the performance of broadcast strategies. However, their study didn't consider the interference between nearby BSNs, and as in [13] the evaluation is purely based on simulation.

In [15], Feeney et al. show that interference among co-located 802.15.4 BSNs significantly lowers the packet reception ratio and increases mean latency.

In addition to studying the communication characteristics of BSNs, there is work trying to model BSN channels. A literature review on the channel modeling for BSNs by Attaphongse et al. [16] shows that the path loss is very high in BSNs, and the lognormal distribution better fits the samll-scale fading in BSNs than the traditional Rayleigh and Ricean distributions in other wireless sensor networks.

The channel modeling subgroup of IEEE 802.15.6 (Body Area Network) characterize the path loss of body area network devices taking into account possible shadowing due to the human body and postures of human body [17, 18]. The measurement of path loss at frequency 2.36GHz shows the dominant factor affecting fading in the channel is the movement of human body.

Comparing to path loss, the variance of PRR is a more explicit indicator of link quality. Ankur et al. [19] use a multilevel approach to model the long and short time scale behavior of links in wireless sensor networks. They use Hidden Markov Models (HMM) to model the long-term evolution of the packet reception pattern, and use another HMM or Multivariate Bernoullis (MMB) to model the short-term dynamics. However, there is no similar work that study the packet reception pattern in BSNs.

IEEE 802.15.4 shares the 2.4GHz ISM band with other technologies such as WiFi and Bluetooth, so the interference from other kinds of radios deteriorates BSN link qualities.

Rahul et al. [7] study the co-existence of Bluetooth and 802.15.4 radios. Their result shows that 802.15.4 takes a much bigger hit in performance when the two types of radios are operated simultaneously.

The impact of WLAN interference on BSN is studied by Hauer et al. [20]. They find that WLAN can be a major cause for substantial packet loss in the IEEE 802.15.4 BSNs when the transmission power of the BSN is low. The transmission failures are negligible when the transmission power is higher than -10 dBm. The empirical study by Ghayvat et al. [21] shows the interference from other 2.4GHz technologies such WiFi deteriorates Zigbee link qualities in smart building.

During the past few years, IEEE 802.15.6 is proposed to supplement IEEE 802.15.4 to support traffic of higher data rate in body area sensor networks. However, according to study by Dhafer et al. [22], IEEE 802.15.6 suffers similar challenges such as body shadowing and changing environments to achieve reliable communication.

The attenuation caused by body shadowing and the interference from other communication sources in the vicinity make BSN link qualities highly variable. To provide reliable data transmission

for BSNs, people in the field propose different methods.

2.2 Solutions for Reliable BSN Communication

There is existing work [23, 24] on assigning transmission power according to the resulting signal strength. However, body shadowing effect changes fast when people perform dynamic activities such as walking, and existing power assignment methods cannot respond fast enough.

Buddhika et al. [25] try to reduce the interference when multiple BSNs are in the same vicinity. They use a fixed network infrastructure to identify nearby BSNs that are likely to interfere with each other, then these nearby BSNs are configured to operate on different frequencies to reduce interference. In their work, a BSN sends packets periodically to the fixed network, and the location of the BSN is determined by the RSSI values. However, localization using RSSI values is not accurate especially when there is interference from other networks such as WiFi. Other assisted living networks such as AlarmNet [26] include motion sensors, and can provide better location information of BSNs. However, the frequencies are limited, so if there are too many BSNs near each other, we need other techniques to reduce interference.

In BSNs, the sensors collect physiological data and send these data back to the aggregator. These traffic patterns are periodic, so TDMA-based MAC protocols [27, 28, 29] are usually used in BSNs to reduce packet collisions. However, these protocols cannot avoid the interference between nearby BSNs. There are MAC protocols for wireless sensor networks that can adapt to the traffic automatically. For example, Injong et al. propose a hybrid MAC protocol called Z-MAC [30] which behaves like CSMA under low contention and like TDMA under high contention. But there is no similar work for BSNs.

BSN specific MAC protocols are mainly derived from IEEE 802.15.4 [31]. Long et al. [32] propose a MAC protocol to guarantee delivery of critical data. In their method, one superframe is divided into CSMA/CA period and TDMA period. Guaranteed time slots in TDMA period are allocated to data according to their priority. Rae et al. [33] and Sabin et al. [34] propose similar solution despite the superframe structure is different. Adnan et al. [35] propose to dynamically select beacon order (BO) and superframe order (SO) to increase BSN throughput and minimize latency. These solutions lack coordination of multiple BSNs, and thus cannot allocate slots across sensors in different BSNs to avoid collision.

To overcome body shadowing effects, Jie et al. [36] propose a two-hop routing strategy by adding relay nodes to forward packets to coordinator. Their study shows that using relays significantly improve BSN link qualities, especially when there is interference from BSNs in the vicinity. By

contract, in [37] Javaid et al. do not allow multi-hopping of data among in-body sensors. They use minimum power needed to transfer data from body sensors to in-body relay (aggregator), which in turn send data to coordinator and finally to end station.

Sidrah et al. [38] propose an incremental routing protocol which automatically switches from one-hop to two-hop to transfer data from sensor to aggregator. In their solution, nodes first directly send data to aggregator. If the transmission fails, nodes will try to relay the packet though other nodes in the BSN.

It is hard to guarantee both delay and reliability at the same time, and it may not be required to guarantee both. Javed et al. [39] divide BSN traffic into three categories: delay sensitive data, reliability sensitive data, and other data without any constraint. Their routing algorithm treats different types of data with different routing strategy.

To collect data from multiple BSNs, Umer et al. [40] propose a routing approach that first divides nearby BSNs into clusters, then elects cluster heads to rely data to base stations. Though the routing protocol claims to improve the reliability of collecting data from multiple BSNs, it doesn't address inter-BSN interference.

Quality of Service (QoS) in wireless sensor networks is studied to guarantee the network performance [41, 42, 43]. The network topology of a BSN is usually a simple one-hop star topology, therefore the traditional QoS policies such as data fusion and making better routing decisions are not as effective in BSNs. Gang Zhou et al. [44, 45] propose BodyQoS to guarantee the delivery of the most critical data in BSNs. In their work, the mote to aggregator traffic is divided into reserved communication and best-effort communication. Reserved communication is scheduled according to the reservation priority first, then the remaining bandwidth is shared by other motes using a best-effort policy. The effective bandwidth that the QoS system uses is determined by the packets received by the aggregator during a time period. In their later work, they propose a radio-agnostic QoS solution BodyT2 [46] to provide joint throughput and time delay assurance. However, their solutions cannot efficiently schedule multiple BSNs influencing each other.

Zhiqiang et al. [47] observe that when link quality is very poor, fixed transmission rate cannot guarantee lower packet loss rate (PLR). Therefore, they propose a transmission rate allocation policy (TRAP), which lowers transmission rate when link quality is poor, to achieve lower PLR. Based on this, they optimize BSN transmission to minimize energy consumption as well as meet QoS requirements including PLR, throughput, and delay. When arranging transmission schedule, a TDMA-like MAC protocol is used to avoid interference between nodes. However, possible interference from other BSNs is not considered in their work.

To accommodate QoS constraints, Sharbani et al. [48] propose a MAC protocol that transfers critical packets and reliability constrained packets in the beginning of superframe using TDMA, then transfers delay constrained packets and normal packets using CSMA. During TDMA period, pre-determined minislots are reserved to transfer urgent packets.

As mentioned before, health status monitoring is one of the two major application areas of BSNs. Reliability of this kind of applications depends on reliably streaming data from sensor nodes to aggregator. Nedal et al. [49] propose a QoS protocol that considers nodes' remaining energy when constructing routing tree and then allocates bandwidth to nodes accordingly. By solving the joint routing tree construction and bandwidth allocation problem, they maximize the network utility while satisfying the QoS requirements. However, their solution failed to address characteristics specific to BSNs such body shadowing and attenuation.

As discussed in this section, existing work has studied physical layer, MAC layer, network layer, and QoS framework to achieve reliable BSN communication. However, literature lacks holistic solutions that address communication challenges by coordinating multiple network layers. Moreover, there is no existing solutions to deal with the interference caused by multiple BSNs.

2.3 Work on Fall Detection

Besides reliable communication, developing reliable applications is also crucial, because most BSN systems are used for medical purposes. However, most diseases lack quantitative definitions, and their diagnosis and treatment are largely based on doctors' experience. This makes it difficult to develop reliable BSN applications to detect detrimental events or monitor disease trajectories accurately. For example, falls and depression are two common health problems for senior people, and existing BSN applications cannot deal with them reliably.

Falls are one of the most detrimental events for the aged population. About one out of three senior people 65 years or older have at least one fall per year, and falls are the leading cause of injury-related hospitalization for elderly people [50]. Detecting falls accurately can reduce the severe consequences, and thus is of great importance.

However, falls are difficult to accurately detect due to three reasons. First of all, certain fall-like activities are hard to distinguish from real falls. Existing solutions mainly use accelerometers [51, 52, 53, 54, 55] to detect falls, because falls are usually characterized by larger accelerations compared to normal daily activities. However, focusing only on large accelerations can result in many false positives by detecting fall-like activities, such as sitting down quickly, or vigorous walking or jumping, as

falls. Some other fall detection algorithms assume that falls are often accompanied with a prominent change of body orientation, e.g. a typical fall can be detected if one's body orientation changes from standing upright to lying prone horizontally on the floor [4]. However, these kinds of systems also cause false positives when detecting activities ending in horizontal position, such as lying onto bed quickly.

Second, falls have different characteristics according to their causes. Rubenstein et al. [56] summarized major causes of falls in elderly people and their relative frequencies. The first four causes for falls in elderly are: falls stemming from environmental hazards (31%), gait/balance disorder (17%), dizziness and vertigo (13%), and drop attacks (9%). In these categories, falls caused by environmental hazards and gait problems usually happen faster than those caused by dizziness and drop attacks. However, previous work on fall detections only focuses on fast falls, yet cannot recognize slow falls, which causes lots of false negatives.

Third, unlike activities of daily living (ADL), falls are accidents, so it is extremely difficult to collect data of real falls of the elderly. Therefore, existing work on fall detection is mainly based on falls simulated by younger people [57, 58]. However, not all types of falls can be handled by simulation, e.g. falls caused by dizziness cannot be simulated. Therefore, it is important to have a generic fall detection framework which can be adjusted and extended as more real falls are reported.

Existing fall detection solutions mainly analyze acceleration data to detect falls. Prado et al. [59, 60] use a four-axis accelerometer, fixed to the back, at the height of the sacrum, to detect falls. They fix the sensor to the skin to minimize the influence of the displacement of the sensor. Then the system determines if there is an impact according to high frequency acceleration components. Mathie et al. [61] utilize a single, waist-mounted, tri-axial accelerometer to detect falls. In their method, if the peek acceleration exceeds a preset threshold, an abnormal event like a fall, a stumble or a collision may have happened. Lindemann et al. [51] integrate a tri-axial accelerometer into a hearing aid housing which is fixed behind the ear, and use thresholds for acceleration and velocity to differentiate falls and Activities of Daily Living (ADL). Kangas et al. [62] study the acceleration of the waist, wrist, and head for falls and ADL, and show that measurements from the waist and head are more useful for fall detection. Their results also show that the acceleration value ranges are overlapping for falls and ADL, which means simple thresholds alone are not optimal for practical fall detection. Bourke et al. [57] place two tri-axial accelerometers at the trunk and thigh, and derive four thresholds, upper and lower thresholds for both the trunk and thigh. Exceeding any of the four thresholds indicates a fall occurred. Jeong et al. [63] implement an acceleration monitoring system for convenient monitoring of activity volume and recognition of emergent situations such as

falls. They use DSVM (differential signal vector magnitude) to distinguish falls from other dynamic states like running and walking. Mobile phones with accelerometers are also used to detect falls, e.g. Jiangpeng et al. [64] use an HTC G1 phone to detect falls by examining if the acceleration exceeds predefined thresholds. Lisowska et al. [65] compares various supervised, unsupervised, and hybrid methods to detect falls from acceleration data, and propose the best hybrid approach is the combination of Convolutional Neural Network (CNN) and the one-class Support Vector Machine (SVM). The problem with these methods is that some normal activities such as sitting down quickly and jumping also feature large accelerations. Therefore, only using acceleration for fall detection causes many false positives. In addition, these methods cannot detect slow falls which do not feature large accelerations.

To improve fall detection accuracy, some solutions utilize both acceleration and body orientation information to detect falls. Noury et al. [66] develop a fall detector consisting of three sensors: a tilt sensor to monitor body orientation, a piezoelectric accelerometer to monitor vertical acceleration shock, and a vibration sensor to monitor body movements. A fall is detected if the body position is standing and the acceleration exceeds the threshold. In their later work [67], they develop a sensor with two orthogonally oriented accelerometers and use this sensor to monitor the inclination and inclination speed to detect falls. Chen et al. [4] monitor the body orientation before and after an impact, and detect falls based on the change in orientation. Purwar et al. [68] use a chest worn triaxial accelerometer to record the acceleration and the tilt angle between the sensor and the vertical direction. A fall is detected if both the acceleration and the angle cross the thresholds. Leijdekkers et al. [69] present a prototype system for remote healthcare monitoring. In this system, the body orientation is analyzed after a large acceleration, and a fall is detected if the orientation is horizontal or not upright after some period of inactivity. Bourke el al. [70] develop a threshold-based fall detection algorithm using a bi-axial gyroscope sensor. They put the gyroscope at the sternum, and measure the angular velocity, angular acceleration, and change of the trunk angle to detect falls. Postures are detected using preset inclination thresholds in [71]. Body orientation can help improve the fall detection accuracy, but using one single device can only monitor the orientation of the trunk, more posture information cannot be collected using this kind of method. Qiang et al. [5] propose a three-phase fall detection algorithm. Two TEMPO nodes [3], each of which features a triaxial accelerometer and a triaxial gyroscope, are attached on the chest and thigh. Using these two sensors, the activity volume, body postures, and the transition process from a dynamic state to a static posture are monitored. If the acceleration and rotational rate before a lying posture exceed thresholds, a fall is detected. This method uses preset thresholds to recognize postures and evaluate

the transition process, which makes it less applicable for different people and not able to detect slower falls.

There is also work trying to detect falls before they really happen. Nyan et al. [58] attach two sensors on torso and thigh. Each sensor contains a 3-D accelerometer and a 2-D gyroscope. They show that falls can be detected with an average lead-time of 700ms before the impact occurs. Sabatini et al. [72] use both accelerometer and altimeter to detect falls with a mean prior-to-impact time of 157ms. These work still mainly depends on thresholds of accelerations or rotational rates to determine whether a fall is to happen.

Some other work uses cameras or image sensors to confirm the fall detection results generated by accelerometers. Hansen et al. [73] use a camera phone to communicate with elderly people by the emergency service when a fall is detected. Tarbar et al. develop a home care network [74]. In their system, a user wears a badge node providing user-centric event sensing functions such as detecting falls, and the appropriate location of the user is detected by measuring the Received Signal Strength Indicator (RSSI). When a fall is detected by the badge, the most nearby image sensor node is activated for further posture analysis. Chua et al. [75] detect falls by the shape change of the human silhouette through extraction of three human body points (upper, mid-, and lower body parts) from video frame. Though using cameras can improve the fall detection accuracy, it is not feasible when the user is in an open area. In addition, privacy is a big concern for people regarding using a camera to monitor them all the time.

Stone et al. [76] use depth image frames captured using Microsoft Kinect to detect falls. They first calculate a person's vertical state to determine if the person is on the ground. If that is the case, they go back to check how fast the on ground event happens to detect falls. Though better than cameras, depth image sensors still raise major privacy concerns.

Contextual information can be used to improve fall detection accuracy as well as address the privacy concerns. Backere et al. [77] develop a context-aware multi-sensor fall detection platform. In the platform, a commercial fall detector is used to detect *possible* falls, then contextual information such as activity and position extracted from infrared motion sensors and pressure sensors are used to confirm if a real fall happened. This can significantly reduce false positives, for example, a possible fall can be easily falsified if motion sensors detect the subject moving around in the kitchen right after the "fall".

There are also commercial products able to detect falls. Many are based on a watch device that includes accelerometers using thresholds as discussed above. WellAWARE [78] markets a floor vibration sensor used to detect falls in specific locations such as near a bed. This system may consider

many normal activities such as stamping as falls. Philips' Lifeline [79] uses a help button to issue medial alerts when a fall happens. However, when a really serious fall happens, people may not be able to push the button. GE's QuietCare [80] system detects falls using motion sensors: if an elderly person stays in one place like the bathroom for a long time, a fall might have happened. This system lacks fast response. While these products are being used, unfortunately, as far as we know, there is no substantial data published as to the false alarm rates for these products.

2.4 Work on In-Person Interaction Monitoring

Depression is a common health problem for the senior population. According to Joseph et al. [81], depression affects 15%-20% of senior people. Many studies show that a lack of social activities plays an important role in the initialization and development of depression [82] and are one of the most important indicators of physical and mental health in aging patients [83]. Therefore, social activity monitoring is needed for psychiatric clinicians or geriatric professionals to perform more complete and accurate diagnosis, treatment, and evaluation of psychological problems for elderly people. The initial status and longitudinal trajectory of depression can be diagnosed by monitoring ADL (activities of daily living) activities (such as eating, walking) and social interactions with other people. According to Hong et al.[84], eight kinds of social activities are related to depression: working, volunteering, attending religious services, exercising regularly, getting together with others such as family, friends, or neighbors, talking on the phone with others, going to movie or sport events, and eating out.

However, though there are plenty of existing work on recognizing daily activities [85, 86, 87, 88, 89, 90], work focusing on detecting social activities is still very limited. These work [91, 92, 93, 94] usually detects social interactions by analyzing daily activities, location information, and the presence of conversation. As shown in gerontological research [82, 95], more detailed information, such as getting together with other people and the sentiment of a conversation, are exceedingly important for psychiatrists or caregivers to monitor and assess mental health. There are some recent work that detects when people are together [96, 97] using on-body sensors. However, lacking communication between BSNs in these work makes detection of BSN interactions unnecessarily difficult and less accurate.

There is quite a few existing work on recognizing daily activities using BSNs. Raghu et al. [98] attach five accelerometers to a jacket, and perform activity recognition by using Hidden Markov Models to analyze acceleration data. Quwaider et al. [99] distinguish activities with different activity intensity levels, such as run, walk, and sit, by transforming collected acceleration data to the frequency

domain. Low activity postures, such as sit and stand, are distinguished by using Hidden Markov Models to analyze RSSI values from multiple sensors. Pham et al. [100] compute the relative energy distribution over the body according to the acceleration data collected from right-hand top, right-hand bottom, left-hand top, left-hand bottom, and waist. Then the activity recognition is performed using a naive Bayes learner. He et al. [101] use Discrete Cosine Transform (DCT) to transform acceleration data to the frequency domain and then use Support Vector Machine (SVM) to classify different activities. Faye et al. [90] detect micro- (such as jumping, heart beat, and taking a step) and macro-activities (such as walking, running, and sport competition) using data collected from smart glasses, smart watch, and smartphone. However, these methods mainly focus on physical activities (such as walking, running, etc) instead of social interactions (such as talking with friends, watching movies, etc), and thus are not useful for depression detection.

Existing work on social activity detection can mainly be divided into two categories according to whether they use environmental sensors.

In the first category, only on-body sensors, such as GPS, accelerometer, microphone, and camera, are used. Emiliano et al develop a system called CenceMe [93, 94] which automatically detects activities of individuals and shares the sensing results through social networks such as Facebook. In their work, various types of data such as location, acceleration, and audio signals are collected by sensors on mobile phones. Social activities such as partying or dancing can be detected based on sound volume and activity level, respectively. Hong Lu et al develop a system for modeling sound events called SoundSense [92], which can distinguish speech from music and ambient sound. According to Gill et al [95], the sentiment of conversation is strongly related to the mental health of elderly people, however, existing work such as CenceMe and SoundSense only detects the presence of conversation and cannot provide sentiment. Besides microphones, cameras are also used to detect social interactions. Pierluigi at al [102] build a badge which includes a triaxial accelerometer and a JPEG camera. The camera is used to detect the presence of other people, which is an indicator of social interactions. Lane et al. [103] build a match-box sized sensing device called Zoe which detects physical activities and conversation through inertial sensors and microphone. Social activities are intrinsically involved with multiple participants, but solutions in the first category focus on the subject alone without caring about what other participants do during interactions. This dramatically limits their ability to describe and detect high-level social activities.

Today's smartphones usually come with sensors such as GPS, microphone, gyroscope, accelerometer, and camera. Therefore, some work uses smartphones to detect social interactions. Nicholas et al. [104] review phone sensing applications. Emiliano at al. implement the CenceMe [105] application

on Apple iPhone platform, which can detect people's certain social activities such as conversation, parting and dancing. SoundSense [106] use microphone to collect sound data, and use supervised and unsupervised learning techniques to classify general sound types (such as music and voice) and person-specific sound events (such as driving cars and walking), respectively. Brett Adams et al. [107] use smartphones to detect social context, which includes social sphere (where people go), social rhythms (what people do), and social ties (who people know). Flutura et al. [108] use microphone and accelerometer on smartphone to detect laughter, a social signal. Though these smartphone-based applications provide insights about people's social activities, they still have problems: they cannot understand social activities in detail (e.g. CenceMe can recognize conversations, but they cannot determine if people are talking with friends or just responding to a salesman); most of exiting work uses supervised learning techniques, which is a hindrance for unobtrusive sensing. Moreover, the activities existing method try to detect lack proved relationships with depression, which makes existing social activity detection methods not suitable for detecting depression.

Solutions in the second category use environmental sensors to understand interactions of multiple people. Datong et al [83] placed four video cameras and audio collectors in public areas such as dining room, living room and hallway. By analyzing collected audio/video signals, their work can detect high-level social interactions such as greeting, standing conversation, and walking together. Diane et al [109] detected social interactions in a smart home which is equipped with motion sensors distributed about every meter throughout the space. The main problem with solutions using environmental sensors is that they require "purposeful" sensing systems deployed according to nature of a given application, which severely impairs the practicality of the solutions.

Envisaging the prevalence of personal sensing devices, Paul et al [110] discussed possibilities of socially aware computing, in which every device not only performs its own task, but also collaborates with each other to monitor and analyze social interactions. Brett et al [91] equipped each subject with a GPS, and then determined how they spent time at the same location together. For indoor localization, AlarmNet [26], which is an assisted-living and residential monitoring network developed at the University of Virginia, uses context sensors to detect the location of each BSN. The localization method in [111] can tell which room the person is in. However, localization information retrieved from GPS or context sensors are limited to provide only general information on proximity of people, which is not enough to monitor social activities in detail.

Chapter 3

BSN Link Characteristics

3.1 Introduction

In this Chapter, we perform an empirical study on the wireless communication characteristics of a Zigbee BSNs which will be used later in the dissertation. The goal of this study is to understand the factors that impact the reliability of communication in BSNs.

There are different wireless technologies to connect sensors. Lee et al. [112] compare four most popular protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. Figure 3.1 shows their comparison results of power consumption of these four protocols. From Figure 3.1 we can see that ZigBee consumes the least power, which makes it most suitable for BSNs, because most BSN applications require long lifetime. In our study, we choose ZigBee sensors as the hardware platform.

The source to sink communication within a single BSN is subject to complex impacts from real

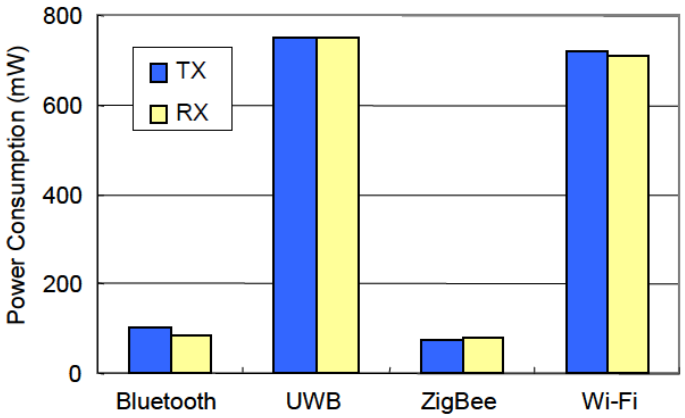


Figure 3.1: Comparison of the power consumption for different wireless protocols.

world communication realities, including the transmission power level adopted by individual nodes, different placements of sensor nodes on the human body, body postures and activities , the number of nodes within one BSN, and the network traffic load. With the presence of multiple BSNs in the vicinity of each other, the interference among BSNs should also be accounted for.

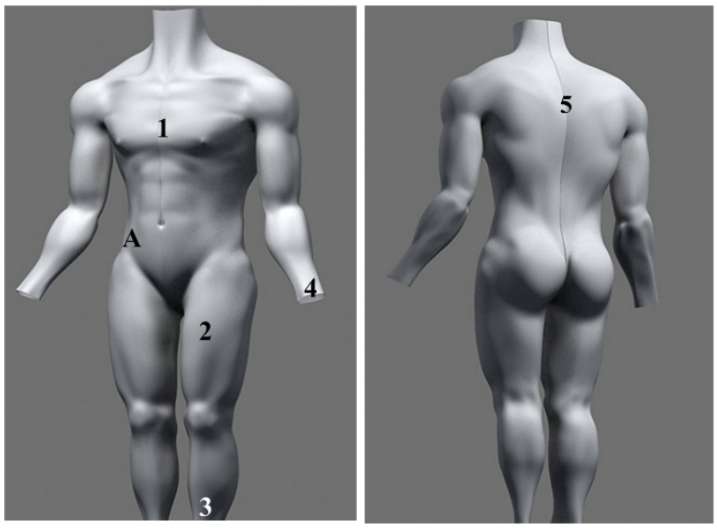
To understand the impact of different factors on source to sink communication within BSNs, we designed and carried out extensive experiments. In our experiments, we choose a typical home environment, and carefully select a ZigBee radio channel (Channel 26) for the nodes to avoid interference from WiFi networks [113]. The devices being used in this experiment are TelosB motes. We chose TelosB motes because they do not have a protruding antenna, which makes them more wearable as body sensors. During the experiments, the BSN on human body assumes a star topology, with all sensor nodes transmitting to a single aggregator placed at a fixed position on the body.

Using higher transmission power levels can improve the link qualities in BSNs, but it also consumes more energy and shortens the life time of BSNs. Therefore, in reality we want to overcome the body shadowing effect with minimum power. The CC2420 radio chip used by TelosB motes can be programmed to transmit at different power levels as shown in Table 3.1.

We carried out our experiments with both single and two BSN settings according to the main factors that affect BSN link qualities. All experiments were carried out three times. Within one BSN, the main factors that affect link qualities are the body shadowing effect and the interference between intra-BSN links. When there are multiple BSNs close to each other, the interference between them needs to be evaluated. In the following sections, we study the body shadowing effect, the interference within one BSN, and the interference between BSNs, respectively.

Power Level	Output Power (dBm)	Current (mA)
31	0	17.4
27	-1	16.5
23	-3	15.2
19	-5	13.9
15	-7	12.5
11	-10	11.2
7	-15	9.9
3	-25	8.5

Table 3.1: Output power of the CC2420 chip.



(a) (b)

Figure 3.2: (a) Frontside sensor placement; (b) Backside sensor placement.

3.2 Body Shadowing Effect

To study the body shadowing effect, we make all sensors send packets to the aggregator at different times so that there will be no intra-BSN interference. Then we observe PRR values for sensors attached on different body locations when a person performs different activities.

3.2.1 Sensor Placement

Figure 3.2 shows the positions where we put sensors in our study, the numbers in the figure correspond to node ID. We place sensors at both front and back at several locations, so that we can study the body shadowing effects due to node placement. The aggregator is placed at fixed position A (waist).

In this experiment, the date rate of each node is 50 packets/second. Every node sends messages using power levels from low to high as shown in Table. 3.1.

Figure 3.3, Figure 3.4, and Figure 3.5 show the PRR values of each node when the person is standing, sitting, and walking, respectively.

Comparing these figures, we can see that placement influences PRR. In Figure 3.3, the PRR of Node 1 to 4 is much better than that of Node 5, because Node 5 is attached on the back, and the link quality between Node 5 and the aggregator deteriorates because of body shadowing. When the person changed the posture to sit down, the nodes attached to the left ankle (Node 3) and the left wrist (Node 4) are out of the line of the sight of the aggregator, so the link quality between them and the aggregator becomes worse: their PRR decreases from about 100% when standing to about 7%

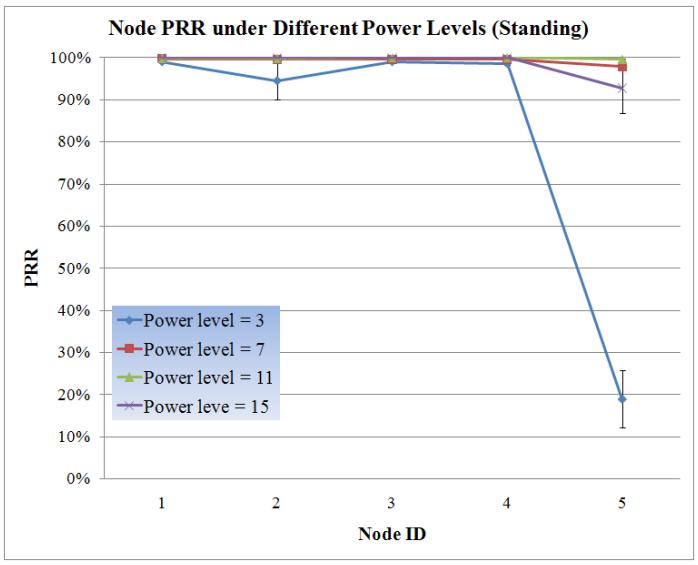


Figure 3.3: The PRR values of the nodes at different body locations when the person is standing.

and 14% when sitting. In Figure 3.5, the PRR of Node 3 and Node 4 is lower then that in Figure 3.3 but higher than that in Figure 3.4 when the power level is 3. This is because the ankle and wrist move into and out of the line of sight of the aggregator periodically when the person is walking, so the PRR falls between the PRR of standing and sitting.

We can also see that the body shadowing effect is much more evident when the transmission power is low. From Figure 3.3–Figure 3.5, the body shadowing is negligible when the power level larger than 11. However, when the power level is as low as 3, the link quality is much more sensitive to body shadowing effect.

3.2.2 Body Posture and Activity

Body postures and activities affect the source to sink distance and change the body shadowing effect for each sensor node, thus impacting the quality of source to sink communication within BSNs. In our experiments, standing, sitting, and walking are the three typical postures and activities considered.

Figure 3.6–Figure 3.10 shows the impact of body postures and activities on PRR. In this experiment, five nodes are attached to the body as shown in Figure 3.2. We record the PRR values under different body postures and activities including standing, sitting, and walking against 8 different power levels. Each node sends 50 packets to the aggregator per second in turn.

Figure 3.6, Figure 3.8 and Figure 3.10 show the PRR values of the sensors on the chest (Node 1), ankle (Node 3), and back (Node 5) under different postures and activities. The results of Node 2 (Figure 3.7) and Node 4 (Figure 3.9) are similar to those of Node 1 and Node 3, respectively.

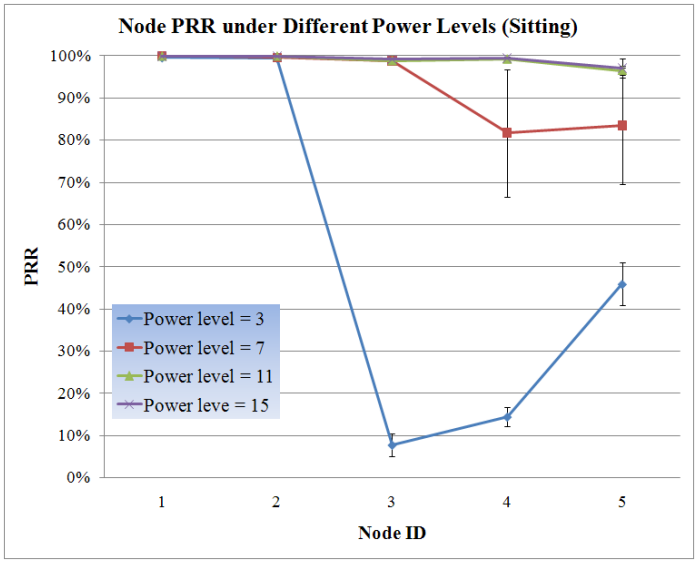


Figure 3.4: The PRR values of the nodes at different body locations when the person is sitting.

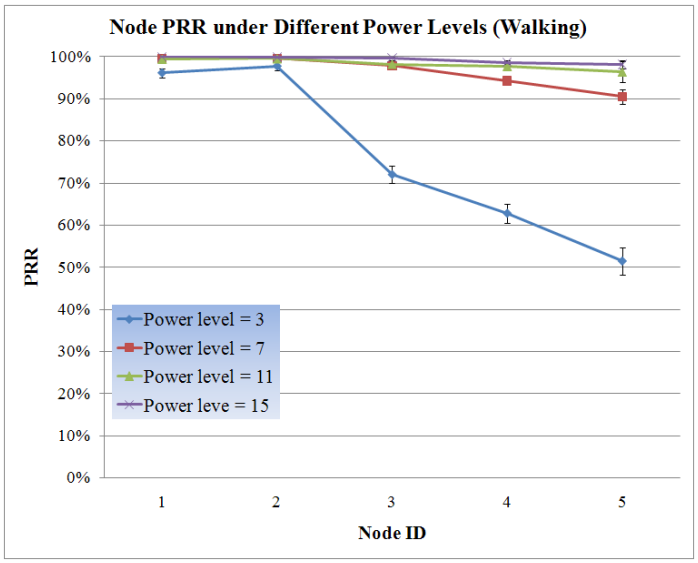


Figure 3.5: The PRR values of the nodes at different body locations when the person is walking.

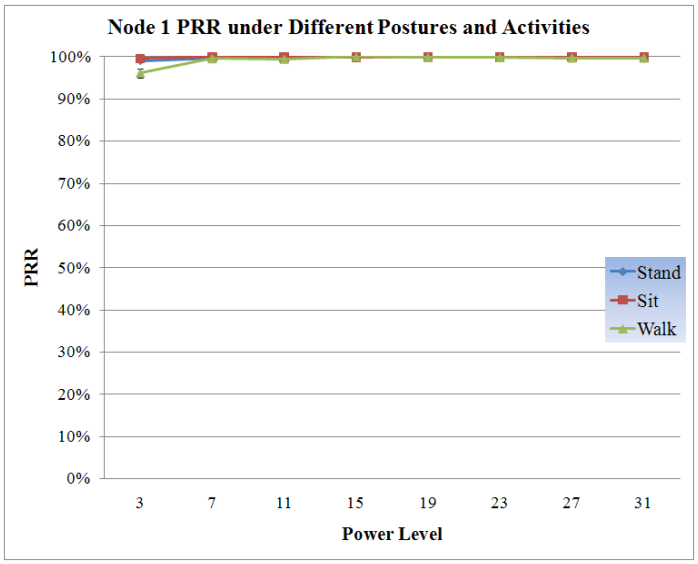


Figure 3.6: The PRR values of the sensor attached on the chest (Node1) when the person is standing, sitting, and walking.

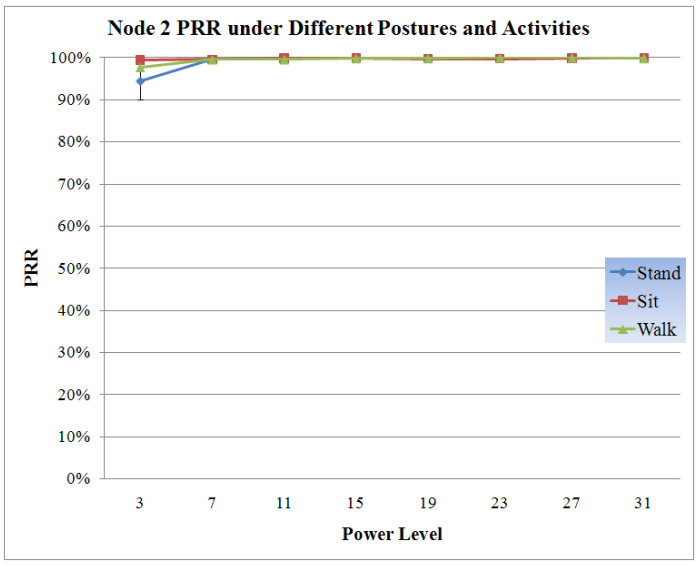


Figure 3.7: The PRR values of the sensor attached on the thigh (Node 2) when the person is standing, sitting, and walking.

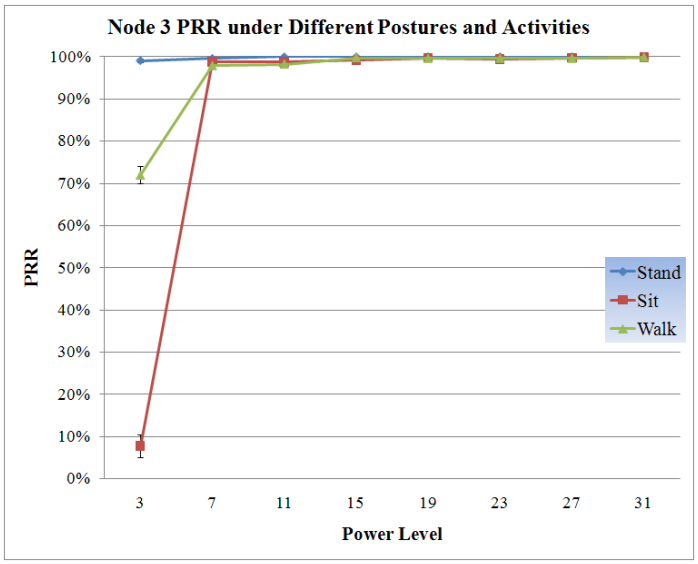


Figure 3.8: The PRR values of the sensor attached on the ankle (Node 3) when the person is standing, sitting, and walking.

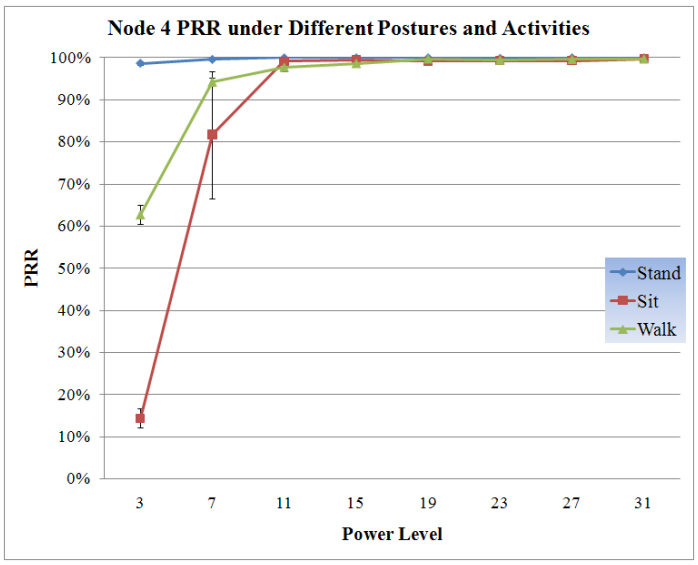


Figure 3.9: The PRR values of the sensor attached on the wrist (Node 4) when the person is standing, sitting, and walking.

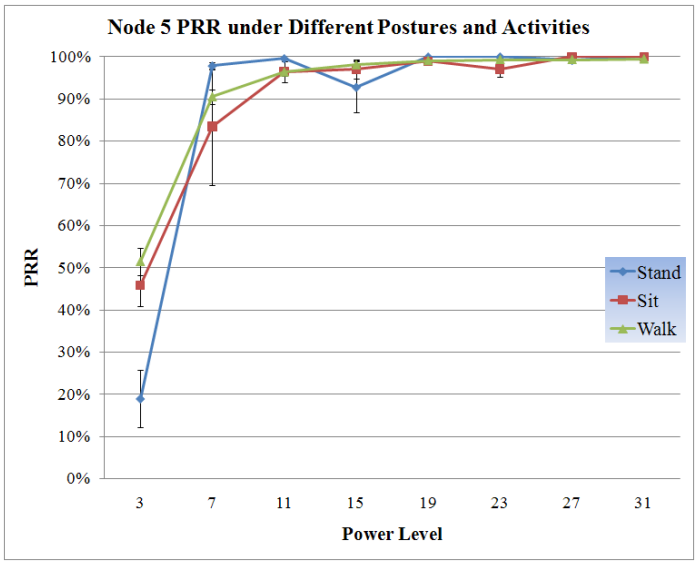


Figure 3.10: The PRR values of the sensor attached on the back (Node 5) when the person is standing, sitting, and walking.

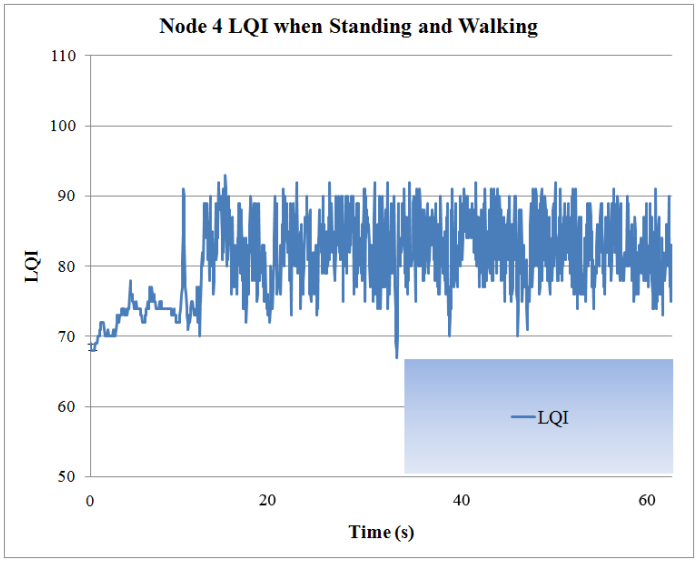


Figure 3.11: The LQI of Node 4 (wrist) when standing and walking.

From Figure 3.6, we can see that the PRR of Node 1 is always close to 100% no matter what postures and activities the person is performing. However, the PRR of other nodes such as Node 3 and Node 5 changes with the postures and activities. These different situations are caused by the body shadowing effect: the link between Node 1 and the aggregator is hardly impeded by the body, while the body often intervenes the links between Node 3 and Node 5 to the aggregator.

Figure 3.11 shows the Link Quality Indicator (LQI) value of the link from Node 4 (on the wrist) to the aggregator when the transmission power level is 3. During the first 10 seconds, the person is standing, and the LQI is between 70 and 80. Then the person begin to walk, and the LQI varies between 70 and 90. The LQI is lower in the beginning is because when standing the node on the wrist is out of the line of sight of the aggregator. When the person begin to walk the node moves into and out of the line of sight of the aggregator periodically, so the LQI becomes better.

However, when the transmission power is above 11, all three nodes become insensitive to postures and activities. This is consistent with our observation in the previous section.

In our experiment, the interference from WiFi networks is avoided by selecting Channel 26, the intra-BSN interference is avoided by making nodes transfer data at different times, and there is also no interference from other BSNs. Under these conditions, Figure 3.6–Figure 3.10 show that power level 11 is enough to achieve reliable communication in a home environment, because the PRR is close to 100% for all sensor nodes under different postures and activities. However, if the nodes transfer data at the same time, there will be interference between them, and we need to evaluate their impact on link qualities.

3.3 Interference within One BSN

To study the intra-BSN interference, all sensors send packets simultaneously, and the number of sensors and the data rate of each sensor are changed to see their impact on PRR.

3.3.1 Number of Nodes within One BSN

To study intra-BSN interference, all nodes send data at the same time. B-MAC[114] is used as the baseline for media access control. Nodes are programmed to generate constant traffic on a periodic basis. The number of nodes within a single BSN relates to the degree of conflict that each node experiences, thus will also affect source to sink link qualities.

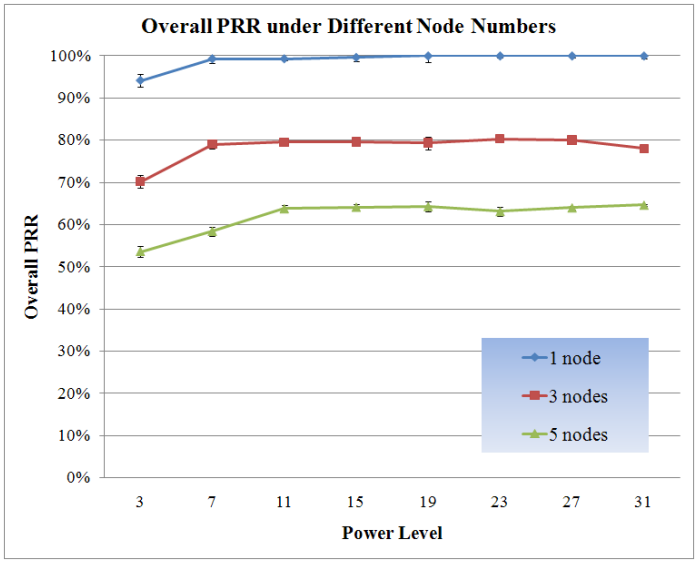


Figure 3.12: Overall PRR when there are different number of sensors within one BSN.

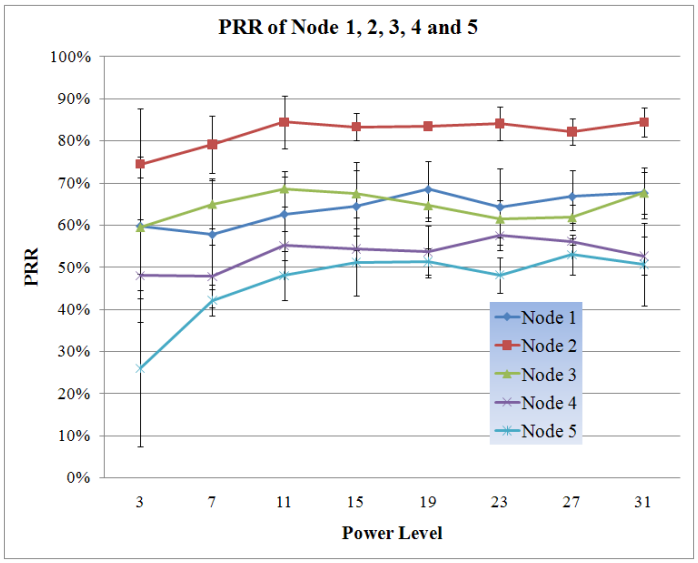


Figure 3.13: PRR of Node 1, 2, 3, 4 and 5 when they transfer data at the same time.

Figure 3.12 shows the impact of the number of nodes on the overall PRR of the BSN. In this experiment, the person stands still with different number of nodes attached. Each node sends 25 packets to the aggregator per second simultaneously. The placement of the nodes are:

- 1 node: Position 1 (as in Figure 3.2)
- 3 nodes: Position 1, 2, and 3
- 5 nodes: Position 1, 2, 3, 4, and 5

From Figure 3.12, we can see that when there is only one source node within the BSN, the PRR is about 100%. But the PRR deteriorates with the increasing number of nodes. When there are three nodes, the overall PRR drops to around 80%. Moreover, the overall PRR with five nodes is lower than that with three nodes. This is not caused by lower PRR of the added Node 4 and Node 5, but by the more severe interference when 5 nodes are present, because our previous results show that with power level higher than 11, every node achieves PRR close to 100%.

The packet loss is not likely caused by network congestion, because the aggregator can accept about 150 packets per second, but the overall traffic in the network is 75 packets per second when there are three nodes and 125 packets per second when there are five nodes.

To further investigate the impact of interference, Figure 3.13 shows the PRR of Nodes 1, 2, 3, 4 and 5 when only these nodes send messages at the same time. Unlike the more steady overall PRR shown in Figure 3.12, the PRR of each node fluctuates because of collisions. Node 2 is closer to the aggregator than other nodes, so its PRR is the highest. Node 5 is farther from the aggregator than other nodes, so its PRR is always lower than others even under maximum transmission power.

3.3.2 Data Rate of Each Node

As mentioned before, nodes are programmed to generate constant traffic on a periodic basis. The data rate of each node affects the total traffic load that the BSN experiences. Network traffic load also has impact on PRR due to collisions.

Figure 3.14 shows the impact of the data rate of each node on the overall PRR of the BSN. In this experiment, the person stands with three nodes attached (Node 1, 2, and 3 as shown in Figure 3.2). All nodes transfer data simultaneously. We record the overall PRR of the BSN against different power levels and different data rates of each node. The data rates studied are 10 packets/second, 25 packets/second, and 40 packets/second.

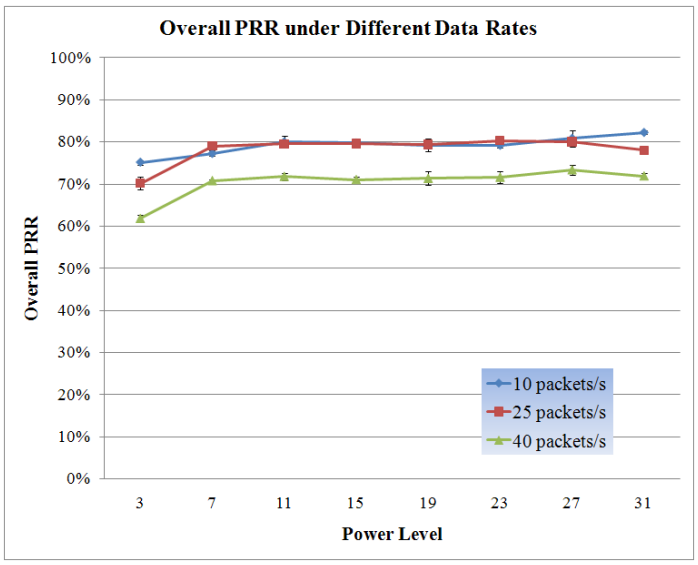


Figure 3.14: Overall PRR when 3 nodes transfer data at different data rates.

From Figure 3.14, we can see that the overall PRR is almost the same when the date rate of each node is 10 packets/second and 25 packets/second, however, with data rate increasing to 40 packets/second, the overall PRR becomes worse, because more traffic load in the network causes more conflicts.

Comparing the experimental results in this section and Section 3.3.1 with those in previous sections, we can see that the intra-BSN interference affects the link qualities badly. Considering the star topology of BSN, TDMA-based MAC protocols are better than CSMA for BSNs.

However, the mobility of BSNs makes existing TDMA-based MAC protocols insufficient for BSNs, because they are not aware of the changes of environmental network conditions. For example, when multiple BSNs are near each other, even if all of them use TDMA-based MAC protocols, there will still be interference between them. Therefore, in the following section, we study the interference between nearby BSNs.

3.4 Interference between BSNs

In addition to the factors accounted for in one single BSN, we evaluate the interference between two BSNs in this section. Two BSN experiments are easy to control and also sufficient to identify the factors that affect link qualities. In our experiments, three factors are examined: the distance between the two BSNs, the power level of the two BSNs, and the data rate of each node. We use 4 TelosB motes in each BSN: 3 sensor nodes and 1 aggregrator. The sensor nodes are placed at

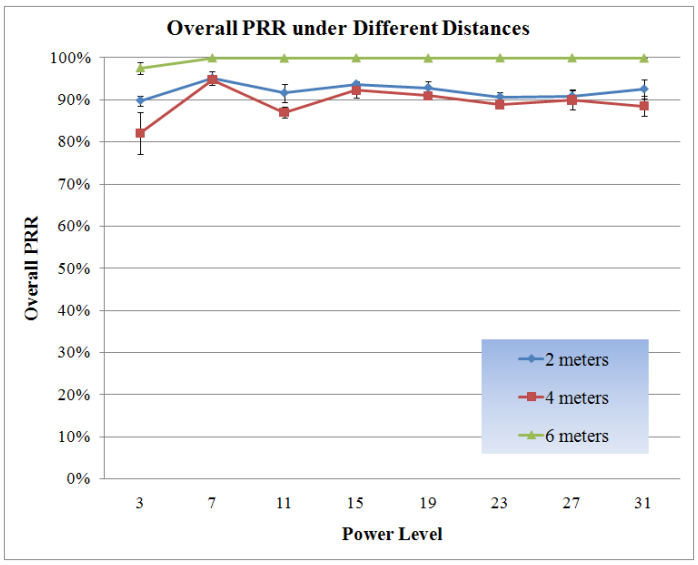


Figure 3.15: Overall PRR when the distances to the interference source are 2m, 4m, and 6m.

positions 1, 2, and 3 as shown in Figure 3.2, and the aggregator is placed at position A. We record the PRR of each node in one BSN (the subject BSN), and use the other BSN (interference BSN) as an interference source. Both of the two are configured to run in a TDMA-based scheme: no two nodes in the same BSN transfer data at the same time, so that we can isolate the interference between BSNs from the interference caused by intra-BSN links.

3.4.1 Distance between Two BSNs

When multiple BSNs are near each other, there are interferences among them. In this set of experiments, we vary the distance between the two BSNs and study the packet delivery within the subject BSN. Each BSN has three source nodes, each of which sends 50 packets to the aggregator per second.

Figure 3.15 shows the PRR values of the subject BSN when two persons are standing, face to face. The power level of the interference BSN is 11 constantly. This power level is enough to achieve reliable communication when there is no interference according to our discussion in the previous section. The power levels of the subject BSN changes from 3 to 31. The distances we measured are 2m, 4m, and 6m. During this experiment, we find that when the distance between two BSNs is larger than 6m, the impact of distance on PRR becomes minimal.

From Figure 3.15, we can see that the overall PRR of the subject BSN becomes better with the increase of the distance from the interference BSN. When the subject BSN is 6m away from the

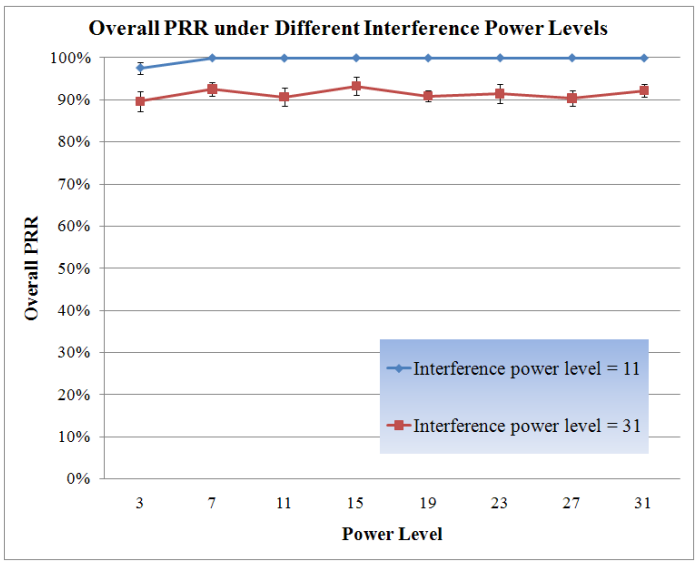


Figure 3.16: Overall PRR when the interference power levels are 11 and 31.

interference BSN, it almost achieves 100% PRR when the power level is larger than 7. However, when the BSN is nearer to the interference source, the PRR fluctuates around 90%.

The overall PRR of the subject BSN when it is 2m and 4m away from the interference BSN is close to each other. This is because the interference signals are still very effective when the two BSNs are 4 meters away.

3.4.2 Power Level of the Interference BSN

The inference among multiple BSNs is related to their power levels, because when the power levels of the interference BSNs are high, their signal is stronger at the aggregrator of the BSN being interfered with. This reduces the value of the capture effect, thus resulting in more packet loss.

In our experiment to study the impact of the power level of the interference BSN, two persons stand face to face. The distance between the two BSNs is 6m. All nodes in the two BSNs send 50 packets per second. The power level of the subject BSN changes from 3 to 31, and the power level of the interference BSN is 11 and 31.

The diamond dot line in Figure 3.16 shows the overall PRR value of the subject BSN when the power level of the interference BSN is 11. The square dot line shows the PRR value when the interference power level is 31, the maximum radio transmission power.

From Figure 3.16, we can see that the lower the power level adopted by the interference BSN the better PRR the subject BSN can achieve. When the power level of the interference BSN is 11, the

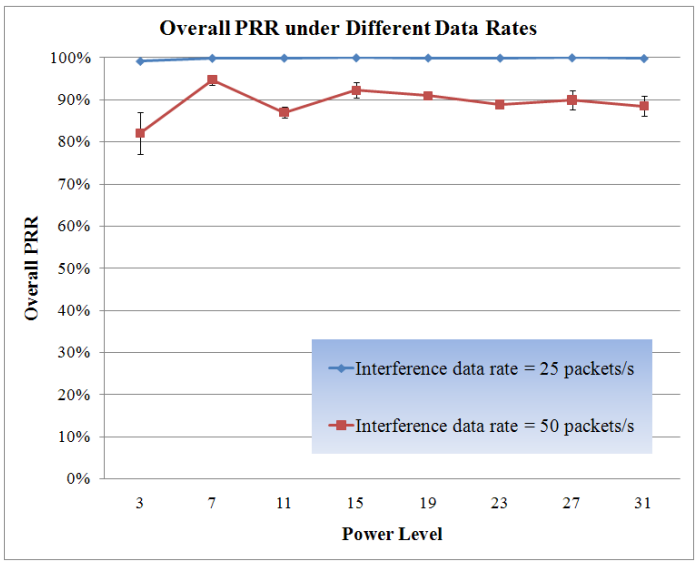


Figure 3.17: Overall PRR when the interference source sends data at 50 packets/s and 100 packets/s.

overall PRR of the subject BSN is close to 100% most of time. While the PRR is never larger than 95% when the interference power level is 31.

When two BSNs are near each other, and the interference BSN transfers data using a high power level, the interference is not able to be overcome for the subject BSN. For example, the square dot line never approaches 100% no matter what transmission power the subject BSN uses.

3.4.3 Data Rate of the Interference BSN

As we discussed in the single BSN experiments, the data rate affects the total traffic load, and has impact on PRR due to conflicts. The change of the data rate of the interference BSN is especially common in multi-BSN scenario: people wearing BSNs can walk around and gather closer to each other to communicate, the BSNs on people can enter and leave the interference range of other BSNs frequently, and thus resulting in rapid change of network traffic.

In this section, we study the impact of the data rate of the interference BSN on the overall PRR of the subject BSN. In this experiment, two persons stand face to face. The transfer mission power of the interference BSN is 11 constantly. The distance between the two BSNs is 4m.

Figure 3.17 shows the overall PRR of the subject BSN when it uses different transmission power levels. The square dot line shows the PRR of the subject BSN when the interference source sends 50 packets per second, and the diamond dot line is the PRR when the interference data rate is 25 packets per second.

From Figure 3.17 we can see when the interference source sends data at a higher rate, the PRR deteriorates because of more collisions. The square dot line is always lower than the diamond dot line.

Comparing the results we get in this section with those in Section 3.3, we can see the interference between BSNs deteriorates link qualities badly. In addition, in our experiments in this section, only two BSNs interference with each other, but in reality, there can be more BSNs near each other, resulting in more nodes transmitting data at the same time, and in that case, there is more interference. Therefore, we need to minimize the interference between BSNs to achieve reliable communication in BSNs.

3.5 Discussion

As discussed in Section 2.1, body shadowing and locomotion impose severe challenges on BSN link qualities. In this chapter, we studied various factors that affect the link qualities of Zigbee based BSNs which we will use later in the dissertation.

We studied the effect of body shadowing on link qualities by measuring PRR and LQI while performing basic activities including standing, sitting, and walking. In an indoor environment, power level 11 is generally enough to achieve acceptable PRR when there is no interference.

However, in reality, there is interference from both intra- and inter-BSN nodes. Our study shows that within one BSN, three nodes transferring packets simultaneously (each at the rate of 25 packets per second) already lowers the PRR to around 80%. With the number of nodes and data rate increasing, the PRR becomes lower.

We also studied the interference between BSNs. Our experiments show that interference between BSNs are ignorable when they are more than 6 meters away and operating at the normal power level of 11. However, the interference range increases with power level, and the severity increases with the data rate of interference BSNs.

We performed all the experiments indoors, because most BSN applications are used for indoor monitoring as discussed in Section 2.3 and Section 2.4. Our study shows the inter-BSN interference already affects link qualities severely when only two BSNs are in the vicinity. With more BSNs joining, the interference will become worse.

With the study results in this section, we propose our BSN QoS framework in the next chapter to address both intra- and inter-BSN interference and overcome body shadowing effects.

Chapter 4

Quality of Service for Multiple BSNs

4.1 Introduction

As discussed in Section 2.1, challenges to BSN communication are mainly from body shadowing and interference. To overcome body shadowing, existing work usually increases transmission power or utilizes relay nodes, both of which result in increasing energy consumption. We choose not to use relay nodes, because it dramatically complicates transmission scheduling.

According to Chapter 3, using enough transmission power makes body shadowing effect negligible. To achieve better lifetime, our solution tries to use the minimum energy needed to overcome body shadowing by dynamically adjusting the transmission power based on the link condition.

Section 2.2 discussed many existing work trying to reduce interference by using TDMA-like protocols. However, existing work only deals with intra-BSN interference. In reality, multiple BSNs can be present in the same vicinity at then same. For example, patients in hospitals or elderly people in assisted living facilities can all wear on-body monitoring systems while they are near to each other. In this scenario, interference between BSNs (inter-BSN) must be avoided to achieve better BSN communication. In this dissertation, we dynamically group BSNs in the vicinity into one group, and schedule transmission slots across BSNs in the same group to avoid inter-BSN interference.

Most importantly, the ultimate goal of developing QoS solutions to achieve reliable BSN communication is making BSN applications reliable. However, few if any existing work relates real application requirements to QoS implementations. In this dissertation, we propose a generic way to profile BSNs

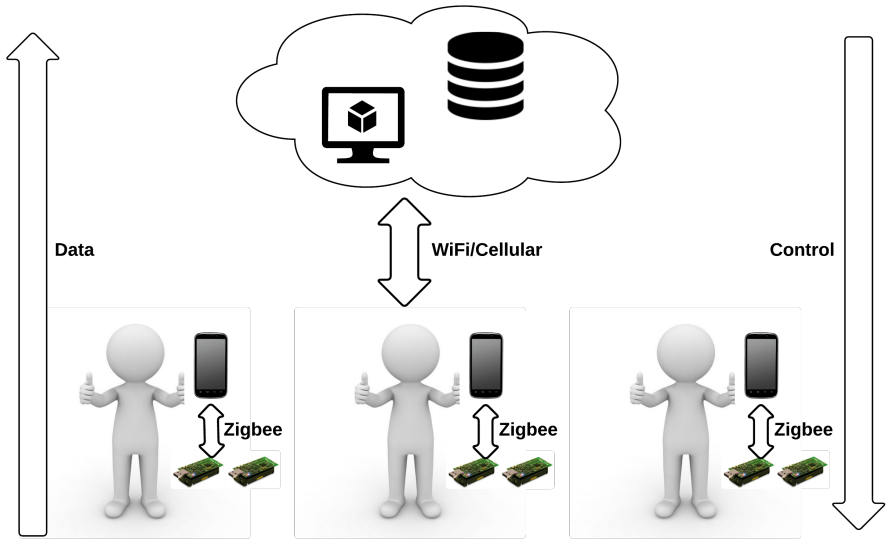


Figure 4.1: Thee-level BSN QoS topology.

and applications, then our QoS algorithm matches profiles of BSNs and applications to guarantee the reliability of BSN applications. By adopting the concept of profiling, our work separates lower level data collection in BSNs from upper level data processing in applications. The result is a system in which BSNs and applications can dynamically register and update themselves, and our QoS algorithm allocates bandwidth and determines transmission schedule for the whole system to achieve desired fidelity requirements automatically.

In the following sections, we will discuss how to group nearby BSNs into the same group, how to describe BSN hardware platforms and BSN applications using profiles, how to schedule data transmission based on these information, and how to adjust transmission power to overcome the body shadowing effect. This chapter focuses on the QoS solution, the whole system is evaluated in Chapter 7 after introducing our BSN applications of fall detection in Chapter 5 and in-person interaction monitoring in Chapter 6. The reason to do this is that many parts of the solutions in this chapter such as profiling can be better evaluated using real applications when the whole system including both BSN hardware and application software are integrated.

4.2 Three-Layer BSN QoS Topology

As shown in Figure 4.1, a typical BSN system can be divided into three layers: on-body sensor nodes, the aggregator, and the base station. Sensor nodes are attached on the human body to collect data.

An aggregator is usually a smartphone which can run applications to process the data collected from sensor nodes and then sends processed data to base stations on the premises or in cloud for storage or additional processing. This process is shown as the *data* flow shown in Figure 4.1.

In our dissertation, to accommodate multiple BSNs and to achieve optimal configuration for the whole system, we need to be aware of the characteristics and status of each BSN. As shown in Figure 4.1, the base station is the sink of the data flow, and has all the information about every BSN. Therefore, we naturally compute how each BSN operates at the base station, and then pass the *control* instructions down to aggregators and then to sensor nodes.

In our system, we use TelosB motes as the sensor platform. The reason to choose the TelosB mote is because it features an embedded antenna, which is non-intrusive and more appropriate for on-body use. A Pololu LSM303DLM sensor board with a 3D compass and a 3D accelerometer is connected to the TelosB node through expansion connector for collecting acceleration data. Smartphones are ideal to be used as aggregator. However, since most smartphones cannot communicate with Zigbee sensor nodes, we use laptops connected with TelosB nodes as aggregators. The base station can be a normal on-prem host or a virtual machine hosted in the cloud. IEEE 802.15.4 radio is used to transfer data between sensor nodes and aggregators, while WiFi/cellular is used to communicate between aggregators and the base station.

4.3 Automatic Grouping of BSNs

As shown in Section 3.4, interference between BSNs becomes negligible at a certain distance which is related the transmission power. To coordinate multiple BSNs to reduce interference as well as fully utilize bandwidth, our solution first puts BSNs close to each other into the same group. This grouping is automatically adjusted with the locomotion of people wearing BSNs.

In spite of the concern about the radio channel uncertainty, RSSI has been widely used for distance estimation, because, rough as the estimation it provides, it does not require extra hardware and is easy to implement [115]. In our solution, RSSI is used as an indicator of *equivalent distance* between BSNs: if one BSN receives packets from the other with low RSSI, the reason can be the other BSN is spatially far away, the environment features fast path loss, or the BSN sends packets with lower power. In any case, lower RSSI means the other BSN is not a major source of inter-BSN interference, so grouping BSNs by RSSI is both sufficient and efficient for our purpose of reducing interference between BSNs.

Algorithm 1 Group nearby BSNs based on RSSI: Aggregator A_i.

$m \leftarrow$ beacon broadcast interval
$w \leftarrow$ moving average window size
while do
    Aggregator A_i broadcasts beacon packet b_i
    (Events: Aggregator A_i receives beacon packet b_j $(j \neq i)$ from Aggregator A_j)
    Aggregator A_i calculates the moving average of RSSI $\overline{r_j}$ from A_j during window w:

$$\overline{r_j} = \frac{\sum_1^{n_j} r_j}{n_j}$$

    , where n_j is the number of packets received from A_j in last w seconds
    Aggregator A_i sends RSSI vector $\vec{r_i} = (\overline{r_1}, \cdots, \overline{r_n})$ to base station
end while

Algorithm 2 Group nearby BSNs based on RSSI: Base station.

while Base station receives RSSI vector $\vec{r_i}$ from Aggregator A_i **do**
Base station update RSSI similarity matrix $R \begin{bmatrix} \vec{r_1} \\ \vec{r_2} \\ \vdots \\ \vec{r_i} \\ \vdots \\ \vec{r_n} \end{bmatrix}$, in which higher RSSI average means nodes are closer to each other and thus more similar.
Use hierarchical clustering to automatically group nearby BSNs. Use the average distance between elements of each cluster as the distance (linkage) between two clusters, $L(A, B) = avg\{d(x, y)|x \in A, y \in B\}$.
end while

4.3.1 Grouping Algorithm

We group BSNs in the vicinity together and make sensors in the same group of BSNs transfer data one after another. RSSI and hierarchical clustering are used in our algorithm to dynamically group BSNs.

The first part of the grouping algorithm is shown in Algorithm 1. Because of using power adaptation to overcome the body shadowing effect, the power levels used to transfer data from sensor nodes to the aggregator can be different from node to node depending on sensor locations and body postures. To measure the interference range of the whole BSN, each aggregator broadcasts beacon packets every m seconds using the *maximum* power level used by the sensor nodes within the BSN. Moving averages of RSSI values of received packets from other BSNs are calculated with a window size of w seconds. Then the aggregator sends the moving average vector to the base station. In Algorithm 1, m controls how frequently the system updates location information of BSNs, w determines how sensitive the algorithm is to radio channel uncertainty.

Figure 4.2 shows an example of grouping three BSNs. In this case, each node broadcasts to two

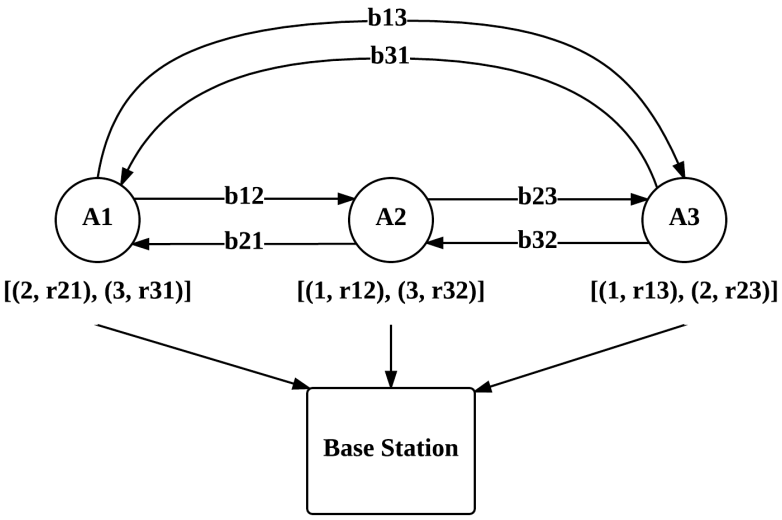


Figure 4.2: Example of grouping 3 BSNs.

	A1	A2	A3
A1			
A2	(r12+r21)/2		
A3	(r13+r31)/2	(r23+r32)/2	

Table 4.1: Example of similarity matrix.

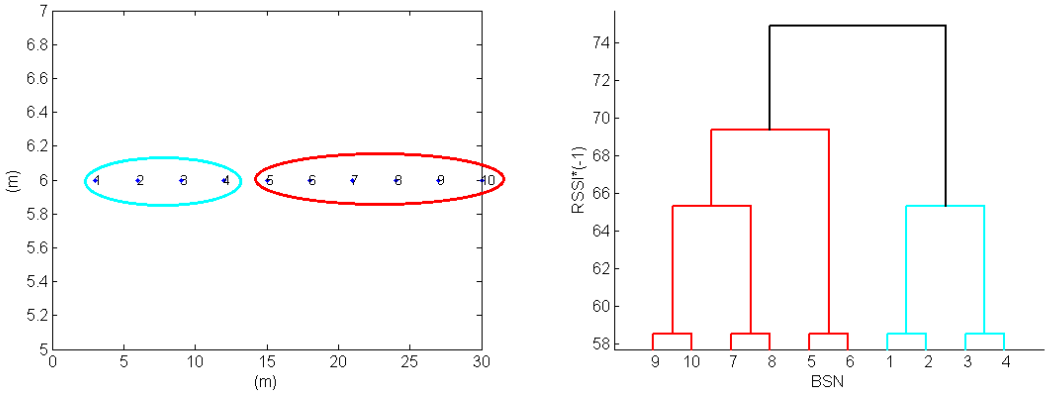
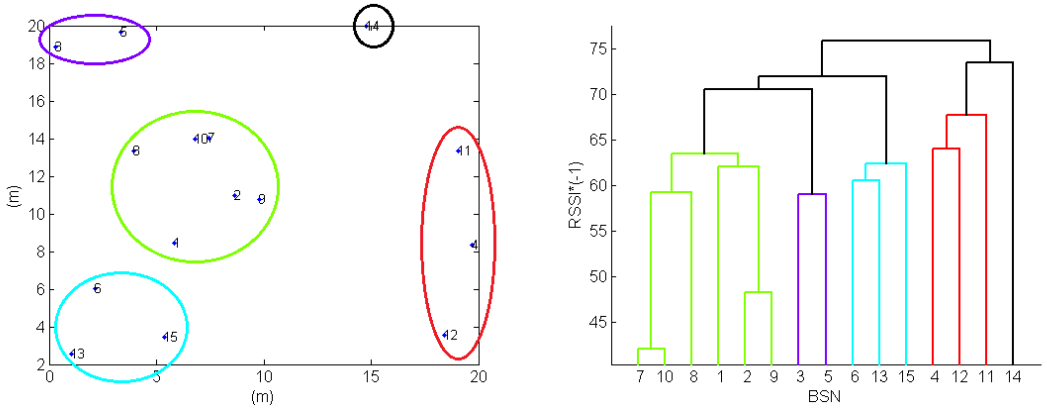


Figure 4.3: Simulation of grouping algorithm with 10 BSNs: (a) location of BSNs, (b) grouping result.



(a) (b)

Figure 4.4: Simulation of grouping algorithm with 15 BSNs: (a) location of BSNs; (b) grouping result.

nearby BSNs. Then each node sends the vector containing RSSI moving averages of packets received from two other BSNs to the base station.

At the base station, RSSI averages received from BSNs are put together into a similarity matrix R as shown in Algorithm 2. R is usually a sparse matrix, since one BSN can only receives beacon packets from nearby BSNs. If two aggregators A_i and A_j don't receive beacon packets from each other, R_{ij} and R_{ji} can be considered infinitely large. If the link between A_i and A_j is asymmetric, e.g. aggregator A_i doesn't receive beacon packets from aggregator A_j, but A_j receives beacon packets from A_i, R_{ji} can be assigned a value representing very low power such as $-120dBm$ which is beyond the sensitivity of the radio chip to reflect the weak link between two aggregators. Based on the similarity matrix R, we use hierarchical clustering [116] to automatically group BSNs. The distance between clusters is defined as the average distance between elements of each cluster. The number of groups (clusters) can be controlled by changing distance threshold between clusters. Table 4.1 shows the similarity matrix of the example in Figure 4.2. Figure 4.3 and Figure 4.4 show a simulation of grouping 10 BSNs and 15 BSNs, respectively. In the simulation, the distance threshold for RSSI value we chose was $-70dBm$, and the transmission power level is 11 ($-10dBm$) for all BSNs.

4.3.2 REST API to Publish RSSI Vectors

The communication between aggregators and the base station is through REST APIs. RSSI vectors discussed in Section 4.3.1 are sent to the base station using the POST method shown in Listing 4.1. The payload contains a list of `RSSI` values received from BSNs identified by `BSNId`, which is a

Universally Unique IDentifier (UUID). The payload is in JSON (JavaScript Object Notation) format, which is a lightweight data-interacting format commonly used by Web APIs these days.

Listing 4.1: Uri and payload to publish RSSI vectors to the base station.

```
Request Uri:
POST http://<server>/bodyqos/bsns/<bsn-id>/rssivector

Request Paylod:
{
    "RSSIVector" : [
        {
            "BSNId" : "6fede02d-c1c4-44a2-9b61-8d91b2f87516",
            "RSSI" : 40
        },
        {
            "BSNId" : "21c16c54-abf1-4d14-a193-b69cf1ed8213",
            "RSSI" : 50
        }
    ]
}
```

4.4 Profiling BSN and Application

Nowadays BSN systems are developed using a monolithic approach: BSN applications are developed on top of specific BSN hardware platforms. The result of this approach is that to build a BSN application such as fall detection, full stack knowledge is needed from choosing sensors and mote platforms, to embedded programming, to networking, to smartphone application development, and to back end storage. This dramatically complicates the development of BSN systems. Android Wear from Google and watchOS from Apple try to simplify this development process by running a unified software interface on top of various hardware platforms, but their work is still mainly limited to smart watches, which have far more resources than a normal sensor node.

In addition, this coupling of BSN applications and hardware platforms make it difficult to develop generic MAC, routing, or QoS protocols for BSNs, since they are all tied to the specific hardware choices of each BSN.

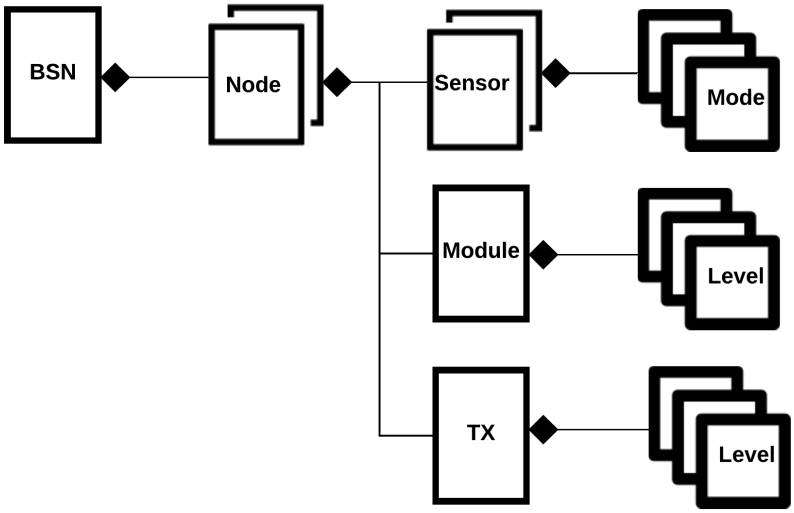


Figure 4.5: Composition of BSN profiles.

In the dissertation, to develop reliable multi-body, multi-function BSNs, we propose a profiling approach that can separate the BSN hardware platforms from the applications running on top of them, so that our QoS solution can be applied to BSNs with different sensors and different applications.

After grouping BSNs, the transmission schedule for each group are determined based on predefined information called *profiles* about each BSN and application. We define two levels of profiles: BSN profiles which are used to describe BSN hardware platforms, and application profiles which are used to define the requirements of BSN applications.

4.4.1 BSN Profile

Figure 4.5 shows the composition of BSN profiles. A `BSN` consists of a list of `Nodes`. For example, in our experiments to study BSN link qualities in Chapter 3, we used a BSN consisting of five TelosB nodes. Besides `Nodes`, each BSN is identified by `Id` which is a UUID, and contains the `Name` of the BSN and the `Priority` of the data generated by the BSN. The `Priority` is used when scheduling transmission. `BSN` properties are listed in Table 4.2.

`Node` properties are listed in Table 4.3. As in BSN profiles, a node has a `Name` and is identified by its `Id`. In addition, a node also has properties of `Type` indicating the kind of the node (TelosB, MICAz, etc) and `Location` indicating where the node is mounted (wrist, chest, etc). Each `Node` contains a list of `Sensors`. For example, a TelosB node contains a visible light sensor, an IR sensor, a humidity sensor, and a temperature sensor. A node also contains `Module` and `TX`. `Module` on a node

Property	Type
Id	UUID
Name	String
Priority	Int
Nodes	List of `Node`

Table 4.2: BSN profile.

Property	Type
Id	UUID
Name	String
Type	String
Location	String
Module	List of `Level`
TX	List of `Level`
Sensors	List of `Sensor`

Table 4.3: Node profile.

includes various components such as processor, RAM, flash memory, etc. It's difficult to separate the energy consumption of these components from each other, but most of the time `Module` as a whole can operate in different modes such as active or sleep, and cost different amount of energy. `TX` is the RF transceiver component on a node. `Module` and `TX` usually can work under different power `Levels`. Each `Level` draws different amount of `Current`. `Level` properties are listed in Table 4.4.

`Sensor` properties are listed in Table 4.5. It has its own `Id` and `Name`. `Type` specifies the kind of the sensor such as accelerometer or gyroscope. `Rate` is the sample frequency range of the sensor. Each `Sensor` can operates in different `Mode`, which includes an `Id`, the measurement `Range` of the sensor, sample `Resolution` and `Accuracy`, and `Bit` (the number of bits for each data sample) as shown in Table 4.6.

4.4.2 REST API to Manage BSN Profiles

BSN profiles discussed in Section 4.4.1 can be published, updated, and deleted using the REST APIs shown in Listing 4.2.

The example payload for POST and UPDATE methods are shown in Listing 4.3. The payload for DELETE method is empty.

Property	Type
Level	Int
Current	Float
Unit	String

Table 4.4: Level profile.

Property	Type
Id	UUID
Name	String
Type	String
Rate	[Int, Int]
Modes	List of `Mode`

Table 4.5: Sensor profile.

Property	Type
Id	UUID
Range	[Float, Float]
Resolution	Float
Accuracy	Float
Unit	String
Bit	Int

Table 4.6: Sensor sampling Mode profile.

Listing 4.2: Uri to publish, update, and delete BSN profiles.

```
Request Uri:
POST/UPDATE/DELETE http://<server>/bodyqos/bsns/<bsn-id>/profile
```

Listing 4.3 shows an example of BSN profile with two TelosB nodes. Each TelosB node has a humidity sensor, a temperature sensor, and an external 3D accelerometer. BSN profiles are described using JSON format.

Listing 4.3: Example of BSN profile with two TelosB nodes.

```
{
    "Id" : "8a1c5d14-e916-4f90-b200-5b69cfffc7e1",
    "Name" : "Sample BSN Profile with Two Nodes",
    "Priority" : 1,
    "Nodes" : [
        {
            "Id" : "f89d6ece-7ae5-4eee-b378-1917e8e90e57",
            "Name" : "Chest Node",
            "Type" : "TelosB",
            "Location" : "Chest",
            "Module" : [
                {
                    "Level" : 0,
                    "Current" : 1.8,
                    "Unit" : "mA"
                },
```

```
                {
                    "Level" : 1,
                    "Current" : 5.1,
                    "Unit" : "pA"
                }
            ],
            "TX" : [
                {
                    "Level" : 0,
                    "Current" : 23,
                    "Unit" : "mA"
                },
                {
                    "Level" : 1,
                    "Current" : 21,
                    "Unit" : "pA"
                },
                {
                    "Level" : 2,
                    "Current" : 1,
                    "Unit" : "pA"
                }
            ],
            "Sensors" : [
                {
                    "Id" : "b3eff0a4-7479-48db-a071-78bf241830be",
                    "Name" : "Humidity Sensor",
                    "Type" : "Humidity",
                    "Rate" : [0, 100],
                    "Modes" : [
                        {
                            "Id" : "07e7122e-a512-4b33-9b77-5d161b491794",
                            "Range" : [0, 100],
                            "Resolution" : 0.03,
                            "Accuracy" : 3.5,
                            "Unit" : "1%",
                            "Bit" : 8
                        }
                    ]
                },
                {
```

```
                    "Id" : "aefe476f-8237-4fd9-90f1-d3377a2d748e",
                    "Name" : "Temperature Sensor",
                    "Type" : "Temperature",
                    "Rate" : [0, 100],
                    "Modes" : [
                        {
                            "Id" : "f14844cb-a30b-4fa2-be4f-e1865544d085",
                            "Range" : [-40, 123.8],
                            "Resolution" : 0.01,
                            "Accuracy" : 0.5,
                            "Unit" : "C",
                            "Bit" : 16
                        }
                    ]
                },
                {
                    "Id" : "ca0933cf-7961-45f3-9975-496f16a2643b",
                    "Name" : "Acceleration Sensor",
                    "Type" : "3D Accelerometer",
                    "Rate" : [0, 400],
                    "Modes" : [
                        {
                            "Id" : "b41d19db-38c0-45aa-9a10-f8472d6f4a37",
                            "Range" : [-8.0, 8.0],
                            "Resolution" : 0.01,
                            "Accuracy" : 0.05,
                            "Unit" : "g",
                            "Bit" : 48
                        }
                    ]
                }
            ]
        },
        {
            "Id" : "40c09357-a7f5-49e2-b8c9-4c3d5b700d79",
            "Name" : "Left Thigh Node",
            "Type" : "TelosB",
            "Location" : "Left Thigh",
            "Module" : [
                {
                    "Level" : 0,
```

```
                    "Current" : 1.8,
                    "Unit" : "mA"
                },
                {
                    "Level" : 1,
                    "Current" : 5.1,
                    "Unit" : "pA"
                }
            ],
            "TX" : [
                {
                    "Level" : 0,
                    "Current" : 23,
                    "Unit" : "mA"
                },
                {
                    "Level" : 1,
                    "Current" : 21,
                    "Unit" : "pA"
                },
                {
                    "Level" : 2,
                    "Current" : 1,
                    "Unit" : "pA"
                }
            ],
            "Sensors" : [
                {
                    "Id" : "0ea21780-9e54-48ea-8db9-905121e1c25c",
                    "Name" : "Humidity Sensor",
                    "Type" : "Humidity",
                    "Rate" : [0, 100],
                    "Modes" : [
                        {
                            "Id" : "ed8edceb-0dbf-4382-b081-2f4c149de35d",
                            "Range" : [0, 100],
                            "Resolution" : 0.03,
                            "Accuracy" : 3.5,
                            "Unit" : "1%",
                            "Bit" : 8
                        }
```

```
                    ]
                },
                {
                    "Id" : "307ae9ff-7b06-4753-9237-7f2b3b514725",
                    "Name" : "Temperature Sensor",
                    "Type" : "Temperature",
                    "Rate" : [0, 100],
                    "Modes" : [
                        {
                            "Id" : "a1a13512-c549-4d39-9792-bde42ba514df",
                            "Range" : [-40, 123.8],
                            "Resolution" : 0.01,
                            "Accuracy" : 0.5,
                            "Unit" : "C",
                            "Bit" : 16
                        }
                    ]
                },
                {
                    "Id" : "f97b180d-458f-4c5c-970c-894ee39f1b40",
                    "Name" : "Acceleration Sensor",
                    "Type" : "3D Accelerometer",
                    "Rate" : [0, 400],
                    "Modes" : [
                        {
                            "Id" : "930ee077-5f05-4964-a388-897f49e9fc99",
                            "Range" : [-8.0, 8.0],
                            "Resolution" : 0.01,
                            "Accuracy" : 0.05,
                            "Unit" : "g",
                            "Bit" : 48
                        }
                    ]
                }
            ]
        }
    ]
}
```

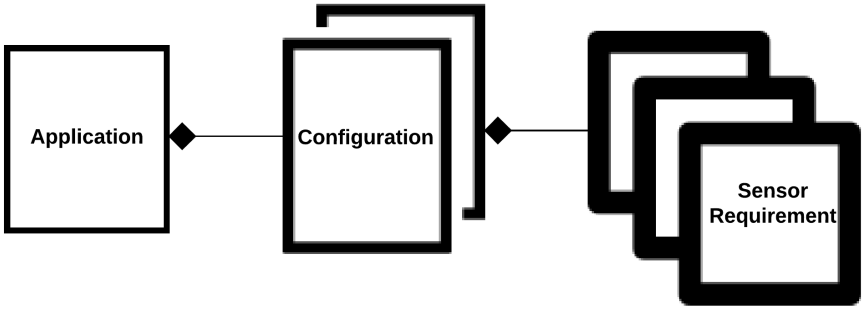


Figure 4.6: Composition of application profiles.

4.4.3 Application Profile

Figure 4.6 shows the composition of application profiles. `Application` properties are listed in Table 4.7. Each BSN `Application` has a descriptive `Name` and is identified by a unique `Id`. The `Priority` is used to determine the level of emergency of the application: the lower the `Priority`, the more urgent to transfer the data used by the application. BSN profiles also contain a `Priority` as shown in Table 4.2, which reflects the priority of the person wearing the BSN. For example, BSNs deployed onto people in an Intensive Care Unit (ICU) should be given higher priorities than normal patients. The priority of an application is used to specify which application should take precedence over other applications worn by the same person. An `Application` also consists of a list of `Configurations`.

`Configuration` properties are listed in Table 4.8. Each `Configuration` contains the a list of `SensorRequirements`, which specify the requirement of each sensor used by the application. Besides sensor requirements, each `Configuration` has a `Preference`. The lower the `Preference`, the more preferable is the `Configuration`. In general, a `Configuration` is more preferable if the `SensorRequirements` require data of higher fidelity such as higher sampling rate or lower delay requirement.

The requirements for a sensor is specified in `SensorRequirement`. `SensorRequirement` properties are list in Table 4.9. `Location` designates where the sensor should be attached such as chest or

Property	**Type**
Id	UUID
Name	String
Priority	Int
Configurations	List of `Configuration`

Table 4.7: Application profile.

Property	Type
Id	UUID
Preference	Int
SensorRequirements	List of `SensorRequirement`

Table 4.8: Configuration profile.

Property	Type
Location	String
Type	String
Rate	Float
Delay	Float
Range	[Float, Float]
Resolution	Float
Accuracy	Float
Unit	String

Table 4.9: SensorRequirement profile.

ankle. `Type` is the kind of the sensor, such as "3D accelerometer" or "Gyroscope". `Delay` specifies the maximum time interval between updates of data from the sensor to reach the aggregator. `Rate`, `Range`, `Resolution`, `Accuracy`, and `Unit` are the same as in Table 4.5 and Table 4.6. The main difference is that `Rate` in the application profile is a single Float instead of a range of Int, because applications have determined sensing rates and may sample data less frequent than 1Hz, which means `Rate` can be smaller than 1.

4.4.4 REST API to Manage Application Profiles

Application profiles discussed in Section 4.4.3 can be published, updated, and deleted using the REST APIs shown in Listing 4.4.

Listing 4.4: Uri to publish, update, and delete application profiles.

```
Request Uri:
POST/UPDATE/DELETE http://<server>/bodyqos/apps/<application-id>/profile
```

The example payload for POST and UPDATE methods are shown in Listing 4.5. The payload for DELETE method is empty.

Listing 4.5 shows an example profile of the fall detection application which uses two nodes. Each node has a 3D accelerometer sensor. Application profiles are described using JSON format.

Listing 4.5: Example of Application profile for fall detection.

```
{
    "Id" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
```

```
    "Name" : "Fall Detection",
    "Priority" : 1,
    "Configurations" : [
        {
            "Id" : "03a0701e-9b47-49eb-9b52-0851e113313c",
            "Preference" : 1,
            "SensorRequirements" : [
                {
                    "Location" : "Chest",
                    "Type" : "3D Accelerometer",
                    "Rate" : 100,
                    "Delay" : 100,
                    "Range" : [-8.0, 8.0],
                    "Resolution" : 0.01,
                    "Accuracy" : 0.05,
                    "Unit" : "g"
                },
                {
                    "Location" : "Left Thigh",
                    "Type" : "3D Accelerometer",
                    "Rate" : 100,
                    "Delay" : 100,
                    "Range" : [-8.0, 8.0],
                    "Resolution" : 0.01,
                    "Accuracy" : 0.05,
                    "Unit" : "g"
                }
            ]
        },
        {
            "Id" : "2ecaa316-18f1-427b-bfda-a14e8a30e6e9",
            "Preference" : 2,
            "SensorRequirements" : [
                {
                    "Location" : "Chest",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 500,
                    "Range" : [-8.0, 8.0],
                    "Resolution" : 0.01,
                    "Accuracy" : 0.05,
```

```
                    "Unit" : "g"
                },
                {
                    "Location" : "Left Thigh",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 500,
                    "Range" : [-8.0, 8.0],
                    "Resolution" : 0.01,
                    "Accuracy" : 0.05,
                    "Unit" : "g"
                }
            ]
        }
    ]
}
```

The REST APIs used to register, update, and get the mappings between BSNs and applications is shown in Listing 4.6. The payload contains a list of application `Id`s.

Listing 4.6: Uri and payload to publish, update, and get mappings between BSNs and applications.

```
Request Uri:
POST/UPDATE/GET http://<server>/bodyqos/bsns/<bsn-id>/apps

Request Payload:
{
    "Applications" : [
        "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
        "6dc56b98-df75-4ed9-a599-fa35bec61729"
    ]
}
```

4.5 Compute BSN Settings

In Section 4.4, both BSN sensing platforms and applications running on top of the platforms are defined using profiles. The profiling approach provides a way to separate hardware level details from the requirements of the software. After a BSN system registers its hardware details and the applications to the base station through the REST APIs, our solution automatically computes

possible BSN *settings* based on these profiles. Each *setting* contains the configurations of all the nodes in the BSN when running a given list of applications.

The process to compute BSN settings can be divided into two steps: validate configurations of each application profile against the BSN profile to get a list of possible settings assuming only this application is to be run on the BSN platform; then we combine these settings to compute settings that can satisfy the requirements of all the applications hosted by the BSN.

4.5.1 Validate Application Configurations

The algorithm to validate application configurations against BSN profiles is shown in Algorithm 3. Given an application profile A and a BSN profile B, the algorithm iterates all configurations of A. For each configuration, every sensor requirement is examined, and sensors with modes that satisfy the requirement is added to the setting. The output of the algorithm is a list of possible settings for all sensors used by the application.

Listing 4.7 shows the settings got from the BSN profile shown in Listing 4.3 and the application profile shown in Listing 4.5 through Algorithm 3.

Listing 4.7: Example of BSN settings for the application of fall detection.


```
{
    "BSNId" : "8a1c5d14-e916-4f90-b200-5b69cfffc7e1",
    "ApplicationId" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
    "Settings" : [
        {
            "ConfigurationId" : "03a0701e-9b47-49eb-9b52-0851e113313c",
            "Sensors" : [
                {
                    "NodeId" : "f89d6ece-7ae5-4eee-b378-1917e8e90e57",
                    "SensorId" : "ca0933cf-7961-45f3-9975-496f16a2643b",
                    "ModeId" : "b41d19db-38c0-45aa-9a10-f8472d6f4a37",
                    "Rate" : 100,
                    "Bit" : 48,
                    "Delay" : 100
                },
                {
                    "NodeId" : "40c09357-a7f5-49e2-b8c9-4c3d5b700d79",
                    "SensorId" : "f97b180d-458f-4c5c-970c-894ee39f1b40",
                    "ModeId" : "930ee077-5f05-4964-a388-897f49e9fc99",
                    "Rate" : 100,
```

Algorithm 3 Validate each configuration of Application A against BSN B.

```
1: A ← Application profile
2: B ← BSN profile
3: SettingList ← [[[]]]
4: for all Configuration C in A.Configurations do
5:     Result ← [[]]                          ▷ Result keeps possible sensor settings for Configuration C
6:     for all Sensor requirement R in C.SensorRequirements do
7:         T ← []                                                    ▷ T is the sensor list that satisfy R
8:         for all Node N in B.Nodes do
9:             for all Sensor s in N.Sensors do
10:                if N.Location = R.Location ∧ s.Type = R.Type ∧ R.Rate ∈ s.Rate then
11:                    for all Mode m in s.Modes do
12:                        if R.Range ⊆ m.Range ∧ R.Resolution ≥ m.Resolution ∧ R.Accuracy ≥
    m.Accuracy then
13:                            T.Add([N.Id, s.Id, m.Id, R.Rate, m.Bit, R.Delay])
14:                        end if
15:                    end for
16:                end if
17:            end for
18:        end for
19:
20:        Temp ← []
21:        for all t in T do
22:            for all r in Result do
23:                if t[1] is not used in r then                          ▷ t[1] is the sensor Id
24:                    Temp.Add(r.Add(t))
25:                end if
26:            end for
27:        end for
28:
29:        Result = Temp
30:    end for
31:
32:    for all r in Result do
33:        if len(r) == len(C.SensorRequirements) then
34:            SettingList[B.Id][C.Id].Add(r)
35:        end if
36:    end for
37: end for
```

```
                    "Bit" : 48,
                    "Delay" : 100
                }
            ]
        },
        {
            "ConfigurationId" : "2ecaa316-18f1-427b-bfda-a14e8a30e6e9",
            "Sensors" : [
                {
                    "NodeId" : "f89d6ece-7ae5-4eee-b378-1917e8e90e57",
                    "SensorId" : "ca0933cf-7961-45f3-9975-496f16a2643b",
                    "ModeId" : "b41d19db-38c0-45aa-9a10-f8472d6f4a37",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 500
                },
                {
                    "NodeId" : "40c09357-a7f5-49e2-b8c9-4c3d5b700d79",
                    "SensorId" : "f97b180d-458f-4c5c-970c-894ee39f1b40",
                    "ModeId" : "930ee077-5f05-4964-a388-897f49e9fc99",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 500
                }
            ]
        }
    ]
}
```


4.5.2 Combine BSN Settings

The algorithm to combine BSN settings is shown in Algorithm 4. Given BSN settings S_1 and S_2 which are calculated through Algorithm 3 as in the last section, Algorithm 4 iterates all settings to check which settings of the two applications can be combined. If there are no conflicts on how to configure sensors such as sensing mode, we combine the list of sensors in the settings for both applications. If there are conflicts, the algorithm checks if the sensor settings can be combined. The output of Algorithm 4 is a list of possible settings for all sensors used by both applications. Combining more than two application settings can be done through repeating Algorithm 4.

Algorithm 4 Combine BSN settings.

```
1: S_1 ← Settings of BSN B running Application A_1
2: S_2 ← Settings of BSN B running Application A_2
3: S ← {}                                   ▷ S is the settings of BSN B running both A_1 and A_2
4: S.BSNId ← B.Id
5: S.Settings ← []
6: for all setting_1 in S_1.Settings do
7:     for all setting_2 in S_2.Settings do
8:         if ∃s_1, s_2 | s_1 ∈ setting_1.Sensors ∧ s_2 ∈ setting_2.Sensors ∧ s_1.SensorId == s_2.SensorId ∧
   s_1.ModeId ≠ s_2.ModeId then
9:             Continue                                                    ▷ Sensor conflicts
10:        else
11:            apps ← [[A_1.Id, setting_1.ConfigurationId], [A_2.Id, setting_2.ConfigurationId]]
12:            sensors ← []
13:            for all s | s ∈ settings_1.Sensors ∧ s ∉ settings_2.Sensors do
14:                sensors.Add(s)
15:            end for
16:            for all s | s ∉ settings_1.Sensors ∧ s ∈ settings_2.Sensors do
17:                sensors.Add(s)
18:            end for
19:            for all s_1, s_2 | s_1 ∈ setting_1.Sensors ∧ s_2 ∈ setting_2.Sensors ∧ s_1.SensorId ==
   s_2.SensorId do
20:                s ← s_1
21:                s.Rate ← max(s_1.Rate, s_2.Rate)
22:                s.Delay ← min(s_1.Delay, s_2.Delay)
23:                sensors.Add(s)
24:            end for
25:            S.Settings.Add([apps, sensors])
26:        end if
27:    end for
28: end for
```

4.5.3 REST API to Download Computed BSN Settings

Aggregators can download the combined BSN settings from the base station through the REST API shown in Listing 4.8.

Listing 4.8: Uri to get BSN settings.

```
Request Uri:
GET http://<server>/bodyqos/bsns/<bsn-id>/settings
```

Listing 4.9 is an example response payload showing the combined settings of one BSN running two applications. Based on these BSN settings and the grouping results from Section 4.3, the transmission schedule for each group of BSNs can be calculated.

Listing 4.9: Example of combined settings to run two applications.

```
{
    "BSNId" : "8a1c5d14-e916-4f90-b200-5b69cfffc7e1",
    "Settings" : [
        {
            Id : "67551dd7-219d-45c1-84d2-bb3f0558d68a",
            "Applications" : [
                {
                    "ApplicationId" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
                    "ConfigurationId" : "03a0701e-9b47-49eb-9b52-0851e113313c"
                },
                {
                    "ApplicationId" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
                    "ConfigurationId" : "03a0701e-9b47-49eb-9b52-0851e113313c"
                }
            ],
            "Sensors" : [
                {
                    "NodeId" : "f89d6ece-7ae5-4eee-b378-1917e8e90e57",
                    "SensorId" : "ca0933cf-7961-45f3-9975-496f16a2643b",
                    "ModeId" : "b41d19db-38c0-45aa-9a10-f8472d6f4a37",
                    "Rate" : 100,
                    "Bit" : 48,
                    "Delay" : 100
                },
                {
                    "NodeId" : "40c09357-a7f5-49e2-b8c9-4c3d5b700d79",
                    "SensorId" : "f97b180d-458f-4c5c-970c-894ee39f1b40",
```

```
                    "ModeId" : "930ee077-5f05-4964-a388-897f49e9fc99",
                    "Rate" : 100,
                    "Bit" : 48,
                    "Delay" : 100
                }
            ]
        },
        {
            "Id" : "52f08d54-794b-4c29-a42b-6b103f454b7d",
            "Applications" : [
                {
                    "ApplicationId" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
                    "ConfigurationId" : "2ecaa316-18f1-427b-bfda-a14e8a30e6e9"
                },
                {
                    "ApplicationId" : "e98c1fe5-32e8-494c-baf4-be6216d4a75b",
                    "ConfigurationId" : "03a0701e-9b47-49eb-9b52-0851e113313c"
                }
            ],
            "Sensors" : [
                {
                    "NodeId" : "f89d6ece-7ae5-4eee-b378-1917e8e90e57",
                    "SensorId" : "ca0933cf-7961-45f3-9975-496f16a2643b",
                    "ModeId" : "b41d19db-38c0-45aa-9a10-f8472d6f4a37",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 500
                },
                {
                    "NodeId" : "40c09357-a7f5-49e2-b8c9-4c3d5b700d79",
                    "SensorId" : "f97b180d-458f-4c5c-970c-894ee39f1b40",
                    "ModeId" : "930ee077-5f05-4964-a388-897f49e9fc99",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 500
                }
            ]
        }
    ]
}
```

Algorithm 5 Algorithm to compare BSN settings

```
1: S_i ← Setting i of BSN B running multiple applications
2: S_j ← Setting j of BSN B running multiple applications                ▷ i ≠ j
3: Sort S_i.Applications and S_j.Applications by each application's priority in increasing order,
   applications with the same priority are sorted by their Ids.  ▷ Sorting by Id guarantees unique
   order
4: for t ← 0; t < len(S_i.Applications); t++ do
5:     if S_i.Applications[t].Configuration.Preference < S_j.Applications[t].Configuration.Preference
   then
6:         return -1                                                 ▷ S_i is more preferable
7:     end if
8:     if S_i.Applications[t].Configuration.Preference > S_j.Applications[t].Configuration.Preference
   then
9:         return 1                                                  ▷ S_j is more preferable
10:    end if
11: end for
```

4.6 Transmission Scheduling

In Section 4.3 nearby BSNs are grouped together. Then in Section 4.5 the settings for a BSN to host multiple applications are computed. Each BSN setting corresponds to a list of applications with specific configurations. Based on these information, we can obtain the transmission schedule for each BSN.

When scheduling transmission, we first assign BSNs of the same group to different frequency channels. If there are more BSNs in the group than the number of channels, the remaining BSNs are assigned to the channels with most bandwidth left. If the bandwidth requirements of BSNs assigned to the same channel exceeds the limit, we switch BSN settings to those with lower bandwidth requirements. After bandwidth allocation, within each channel, aggregators poll sensor nodes for data, the polling is scheduled with the delay requirement considered. Within each group, we randomly choose one master BSN to moderate the data transmission for the whole group.

4.6.1 Preference and Bandwidth of BSN Settings

As shown in Listing 4.9, each BSN can operate under various settings. Before scheduling transmission for each BSN, we need to define how to determine which setting is more preferable for a BSN, since we should always use the more preferable setting if possible. BSN settings also contain sensor configurations and thus determine the bandwidth requirement, which is a key input to transmission scheduling. This section discusses the calculation of preference and bandwidth of BSN settings.

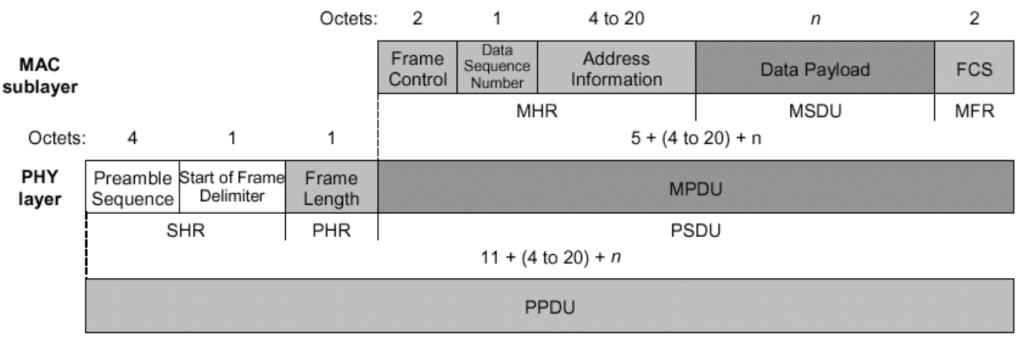


Figure 4.7: IEEE 802.15.4 data frame structure.

Preference

Algorithm 5 shows how to determine which one of two BSN settings is more preferable. In the algorithm, we sort the applications in each setting first by their priority and then by their Id. Sorting by Id is to guarantee the same sorting result for the applications with the same priority level. After this we go through the sorted list to check the first preference difference of application configurations. A less preferable configuration chosen for a higher prioritized application means the setting is less preferable than the other.

Bandwidth

Figure 4.7 shows the data frame structure of IEEE 802.15.4. The maximum length of MPDU (MAC Protocol Data Unit) is 127 bytes, which matches the 1-byte length of the frame length field.

To transfer n bytes of data, the packet size is $11 + \Delta + n$ ($4 \leq \Delta \leq 20$), where Δ is the number of bytes of MAC address information. The value of Δ is platform specific. According to the CC2420 packet structure shown in Listing 4.10, $\Delta = 6$ for TelosB motes which use CC2420. A `type` byte is also added by TinyOS to specify message type.

Listing 4.10: CC2420 packet structure.

```
typedef nx_struct cc2420_header_t {
  nxle_uint8_t length;
  nxle_uint16_t fcf;
  nxle_uint8_t dsn;
  nxle_uint16_t destpan;
  nxle_uint16_t dest;
  nxle_uint16_t src;
  nxle_uint8_t type;
} cc2420_header_t;
```

Algorithm 6 Algorithm to calculate the bandwidth requirement of a BSN setting.

$S \leftarrow$ Setting of BSN B running multiple applications
$G \leftarrow$ Group S.Sensors by their NodeId
$Bandwidth \leftarrow 0$
for all Group g in G **do**
    $minDelay \leftarrow \infty$ ▷ $minDelay$ is the maximum time interval to send data from the node to the aggregator, it is in the unit of milliseconds
    $payloadBits \leftarrow [0, 0]$
    for all Sensor s in g **do**
        $minDelay \leftarrow min(minDelay, s.\text{Delay})$
        $payloadBits \leftarrow payloadBits + s.\text{Rate} \times s.\text{Bit}$ ▷ $payloadBits$ is bits per second as s.Rate is samples per second
    end for
    $minPacket \leftarrow \lceil 1000/minDelay \rceil$ ▷ Minimum packets per second considering delay requirements
    $minPacket \leftarrow max(minPacket, \left\lceil \frac{payloadBits}{maxPayloadSize} \right\rceil)$ ▷ Minimum packets per second considering delay and maximum payload size
    $Bits \leftarrow payloadBits + minPacket \times 8 \times packetOverhead$
    $Bandwidth \leftarrow Bandwidth + Bits$
end for

```
10
11 typedef nx_struct cc2420_packet_t {
12   cc2420_header_t packet;
13   nx_uint8_t data[];
14 } cc2420_packet_t;
```

Though not shown in Listing 4.10, a 2-byte CRC is auto-appended to each outbound packet by the CC2420 radio hardware. The maximum size of a packet is 128 bytes including its headers and CRC, which matches the 802.15.4 specifications.

Increasing the packet size will increase data throughput in TinyOS applications, but it will also increase the probability that packets are destroyed by interference and need to be retransmitted. Therefore, the default maximum payload size for CC2420 is 28, which is defined by `TOSH_DATA_LENGTH` preprocessor variable. Though we can modify this value to increase the maximum payload size, it causes compatibility problems between packets and the Java tools provided by TinyOS. Therefore, in our system, we keep the maximum length the data payload in a packet $maxPayloadSize = 28$.

Based on these information, we can calculate the bandwidth needed for a specific BSN setting using Algorithm 6. In Algorithm 6, to transfer $payloadBits$, at least $minPacket$ packets are needed, and $packetOverhead = 11 + 6 + 1 = 18$ for TelosB motes, because $\Delta = 6$ and 1 byte is used by TinyOS to specify message type.

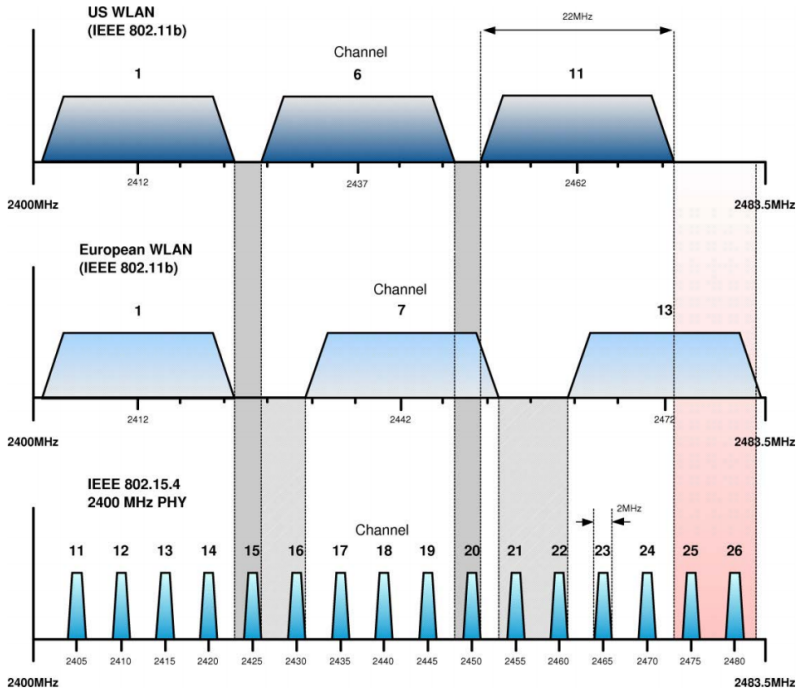


Figure 4.8: Frequency of 802.15.4 channels and WiFi channels

4.6.2 Channel Selection and Bandwidth Allocation

IEEE 802.15.4 has 27 operating frequency channels. Channel 0 to 10 operate at 868MHz/915MHz. Channel 11 to 26 operate at 2.4GHz (shown in Figure 4.8). Most Zigbee sensor motes including TelosB and MICAz used in our system operate at 2.4GHz, because this frequency offers higher bit rate of 250kbps.

2.4GHz is an unlicensed ISM band which is also used by other technologies such as Bluetooth and WiFi. According to the study on the coexistence of IEEE 802.15.4 and other technologies [117], Bluetooth doesn't impact IEEE 802.15.4 data transmissions very much because of the retry mechanism employed by IEEE 802.15.4. After interfering a first transmission, Bluetooth usually hops to a different part of the spectrum and won't interfere with the retry.

However, WiFi interference cannot be ignored as shown in [117, 118]. In our system, we make all the environmental WiFi networks operate on Channel 1 to minimize WiFi interference with Zigbee Channel 15 to 26 as shown in Figure 4.8. Then we use Zigbee Channel 15 for inter-BSN communication between aggregators, e.g. packets used for getting RSSI vectors in Section 4.3 are broadcast at this channel. Channel 16 to 26 are used for intra-BSN communication between nodes and aggregators.

Letting WiFi networks operate on Channel 1 is for the sake of simplicity. Actually we can detect

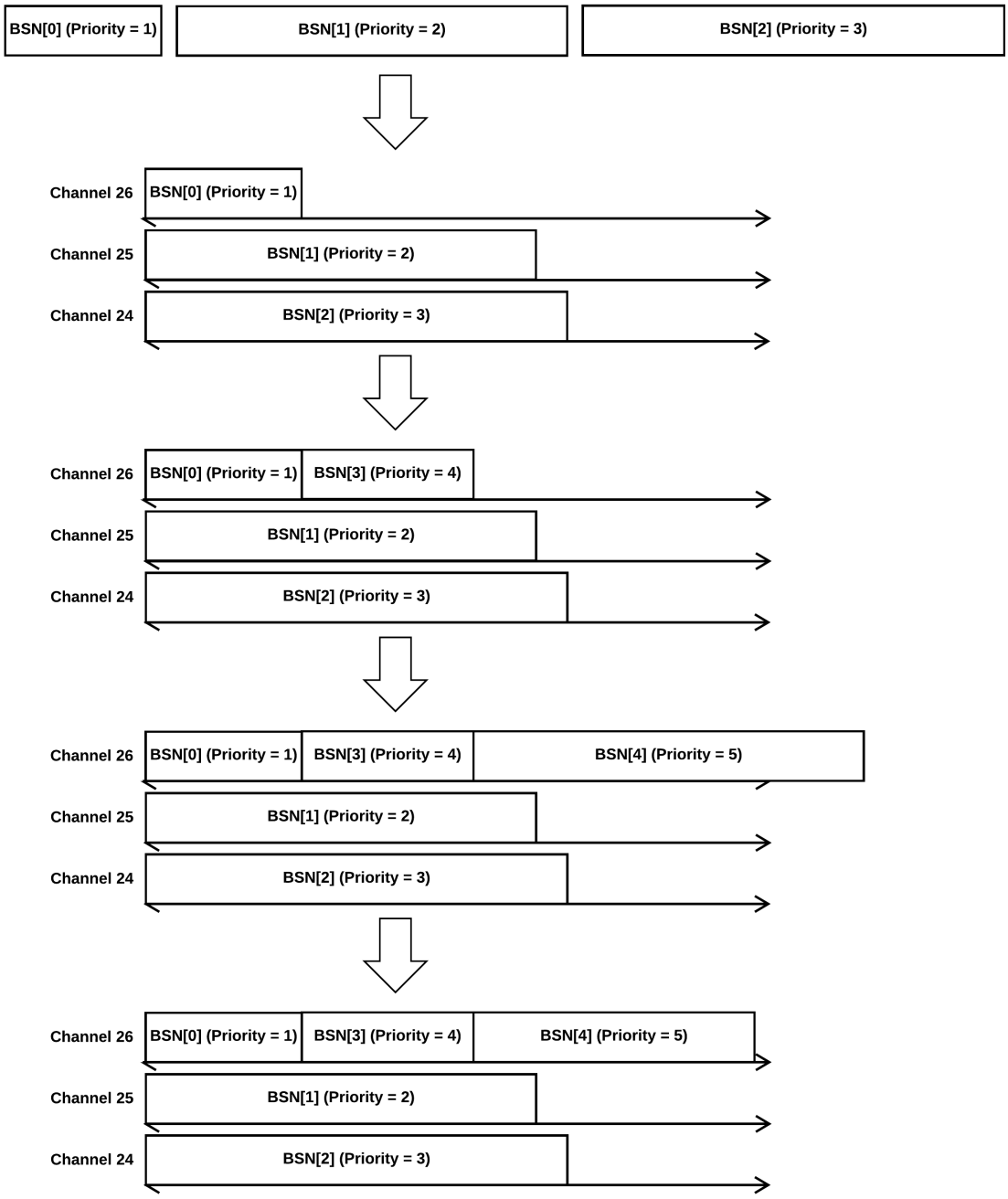


Figure 4.9: Example of channel allocation and bandwidth allocation.

the presence of WiFi interference on a Zigbee channel by performing RSSI noise floor reading, and then choose clear channels for BSN communication. There is also existing work on how to survive WiFi interference for Zigbee networks [119].

When selecting channels for BSNs in the same group, BSNs are first sorted according to their priority defined in BSN profiles (lower value means higher priority), so that in the list B of BSNs, if $i < j$, then $B[i]$.Priority $\leq B[j]$.Priority. BSN $B[i]$ $(0 \leq i \leq 10)$ is assigned to Channel $(26 - i)$. Figure 4.9 shows the process to allocate BSNs to three IEEE 802.15.4 channels. In the beginning, there are three BSNs in the group, and each one is assigned to a different channel based on their priority.

However, if there are more BSNs in the same group than available channels, remaining BSNs will be allocated to the channel with most bandwidth left. To determine the used bandwidth, we first choose the most preferable setting for allocated BSNs according to Algorithm 5. The bandwidth of the chosen setting can be calculated using Algorithm 6. Figure 4.9 shows when $BSN[3]$ joins the group, it was assigned to Channel 26 because it has the most bandwidth left.

With more BSNs joining the group, some channels become too crowded for all the BSNs on the channel to use their most preferable settings. For example, Figure 4.9 shows when $BSN[4]$ joins the group, Channel 26 becomes short of bandwidth. In this case, we adjust BSNs with the lowest priority on the channel (in this case, $BSN[4]$), and change the setting for this BSN to a setting with lower bandwidth requirement, so that there will be enough bandwidth for all BSNs on the channel. In the adjustment process, we always adjust the BSN with the lowest priority first, until it is already using the setting with lowest bandwidth requirement, then we adjust the BSN with the second lowest priority.

If the number of BSNs keep increasing, there will not be enough bandwidth to accommodate all BSNs even though every BSN uses the setting with lowest bandwidth requirement. In this case we can either only schedule slots for BSNs with higher priority, or we can lower the sampling rate of sensors and lower the bandwidth used, though this may fail the fidelity requirement of BSN applications. In reality, since there are 11 channels to be used by nearby BSNs, in most cases this is more than enough to accommodate all BSNs in the same group.

4.6.3 Delay-Aware Real-Time Scheduling

According to [32, 120, 121], IEEE 802.15.4 provides two operating modes: non-beacon mode and beacon-enabled mode. The non-beacon mode simply uses CSMA/CA mechanism for transmission.

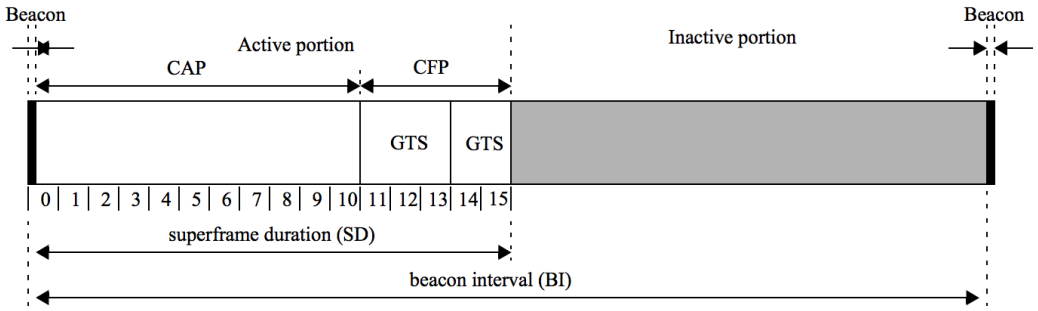


Figure 4.10: IEEE 802.15.4 superframe structure.

The beacon-enabled mode uses a superframe structure shown in Figure 4.10. The superframe is divided into 16 time slots. These slots belong to either Contention Access Period (CAP) or Contention Free Period (CFP). During CAP, nodes access channel with slotted CSMA/CA. During CFP, nodes reserve Guaranteed Time Slot (GTS) and conducts TDMA for communication. Most existing BSN QoS solutions choose to use beacon-enabled mode to utilize GTS for transferring data of high priority.

However, two major disadvantages make the beacon-enabled mode insufficient for coordinating traffic for multiple BSNs. Firstly, a coordinator is needed to use beacon-enabled mode. This may not be a problem when there is only one single BSN system, but it is not feasible to assume there exists a node that can reach all other nodes within the same BSN group. Secondly, in a superframe as shown in Figure 4.10, the number of GTS slots is limited to 7, which can be much less than the number of nodes in a BSN group. Insufficient GTS slots can fail reservation and increase transmission delay.

Given the settings of BSNs assigned to the same channel, the delay requirement $d = minDelay$ and the sample bits generated by sensors per second $p = payloadBits$ of each node can be calculated as between Line 5 and Line 10 in Algorithm 6. Therefore, the number of packets to transfer n during time d for a node can be calculated as:

$$n = \left\lceil \frac{p \times d}{1000 \times maxPayloadSize} \right\rceil \tag{4.1}$$

where $maxPayloadSize$ is maximum payload length in a packet for a specific platform. For TelosB motes, $maxPayloadSize = 28$ as we discussed in Section 4.6.1. Then, the time c needed to send the n packets can be calculated as:

$$c = n \times (T_{cca} + T_{backoff} + T_{system} + T_{air}) \tag{4.2}$$

where T_{cca} is the time to perform the Clear Channel Assessment (CCA), $T_{backoff}$ is the random

delay time when CCA finds the channel is busy. Note that to transfer a packet, it's possible to perform multiple times of CCA and backoff. T_{system} is the time needed for the platform to process a packet, it is different from platform to platform and is usually related to the length of the packet. T_{air} is the time need to transfer one packet in the rate of 250 kbps.

Now we need to allocate time slots to each node to transfer data to the aggregator one after another while meeting the delay requirement. This process is comparable to the task scheduling in real-time systems: the time slots to access the radio channel corresponds to the CPU slots to be allocated to each process, each node needs time to transfer data just like a process needing time to perform computational tasks, nodes need to transfer data periodically as the periodical tasks in real-time systems, and the delay requirement d is the same as period and deadline in real-time system scheduling.

Therefore, in our solution, we use *Rate Monotonic Scheduling* (RMS) algorithm [122] to assign time slots to each node to transfer data. To use RMS, we assign fixed priorities to nodes based on their delay requirement d, the smaller the d, the higher priority the node has, and thus will be allocated before nodes with larger d.

There are many benefits to use the RMS algorithm. It is proven that

Theorem 1 *Rate monotonic (RM) is an optimal priority assignment method.*

which means, if a schedule that meets all the delay requirements exists with fixed priorities, then RMS will produce a feasible schedule.

Moreover, we can easily figure out whether a feasible schedule exists when allocating slots to nodes by calculating the *channel utilization*, denoted as U:

$$U = \sum_{i=1}^{n} (c_i/d_i) \tag{4.3}$$

$U \leq 1$ is a necessary condition for feasibility regardless of scheduling policy. The more important result is that scheduling using static priorities is feasible if

$$U \leq n(2^{1/n} - 1) \tag{4.4}$$

With the number of nodes increasing, the limit of the bound is

$$\lim_{n \to \infty} n(2^{1/n} - 1) = \ln 2 \approx 0.69 \tag{4.5}$$

Property	Type
LastUpdated	DateTime
Channel	Int
Master	Boolean
Frame	Int
NodeSchedules	List of `NodeSchedule`

Table 4.10: Transmission schedule for a BSN.

Property	Type
NodeId	UUID
StartTime	Int
EndTime	Int

Table 4.11: Transmission schedule for a node.

So the feasibility of scheduling is guaranteed if the channel utilization is less than about 0.7. Note that this is a sufficient condition, but not necessary. It is possible that scheduling is feasible but the U is larger than the bound.

The transmission schedule is calculated at the base station every time when there is a grouping update, a BSN profile update, or an application profile update. Aggregators periodically poll the base station for the transmission schedule updates through the REST API defined in the next section. To make sure BSNs update their transmission schedules in time, we use m, the beacon broadcast interval in Algorithm 1 in Section 4.3, as the polling interval.

4.6.4 REST API to Get Transmission Schedule

Table 4.10 and Table 4.11 define the format of the response payload to describe the transmission schedule for a BSN. `LastUpdated` indicates when the schedule was calculated. `Channel` specifies which channel the BSN is assigned to. `Master` depicts if the BSN is the master of the group. The master BSN is randomly chosen by the base station from a group of BSNs. `Frame` is the length of a scheduling cycle. `NodeSchedules` contains transmission schedules for all nodes in the BSN. The schedule of each node contains `NodeId` to identify the node, `StartTime` to specify the time offset to start transmission for the node, and `EndTime` to specify the time offset to stop transmission for the node from the beginning of the frame. Both `StartTime` and `EndTime` are in milliseconds.

Listing 4.11 shows the Uri to get the transmission schedule for a BSN, and Listing 4.12 shows an example of BSN transmission schedule.

Listing 4.11: Uri to get BSN transmission schedule.

```
1 Request Uri:
```

```
GET http://<server>/bodyqos/bsns/<bsn-id>/schedule
```

Listing 4.12: Example of BSN transmission schedule

```
{
    "LastUpdated" : "2017-03-05 11:01:30",
    "Channel" : 26,
    "Master" : true,
    "Frame" : 1000,
    "NodeSchedules" : [
        {
            "NodeId" : "f107245a-4539-4afa-8249-69386a87146f",
            "StartTime" : 0,
            "EndTime" : 50
        },
        {
            "NodeId" : "00bebd6c-1691-488b-8e3e-e338fe222db4",
            "StartTime" : 100,
            "EndTime" : 120
        }
    ]
}
```

The master BSN coordinates data traffic by broadcasting messages to other aggregators in the same BSN group through the control channel (Channel 15 in our implementation):

1. The aggregator of the master BSN broadcasts schedule update time `LastUpdated`.
2. If the transmission schedule of a BSN is not up to date, it polls for the new schedule through API.
3. The aggregator of the master BSN broadcasts transmission initialization message to start the transmission according to the new schedule.

Aggregators relay control messages from the master BSN to ensure they reach every BSN in the group. The broadcast messages also serve for time synchronization.

4.7 Dynamic Power Adaptation

As discussed in Chapter 3, BSNs are subject to the body shadowing effect. Higher transmission power is needed to overcome the body shadowing effect. However, the higher the transmission power,

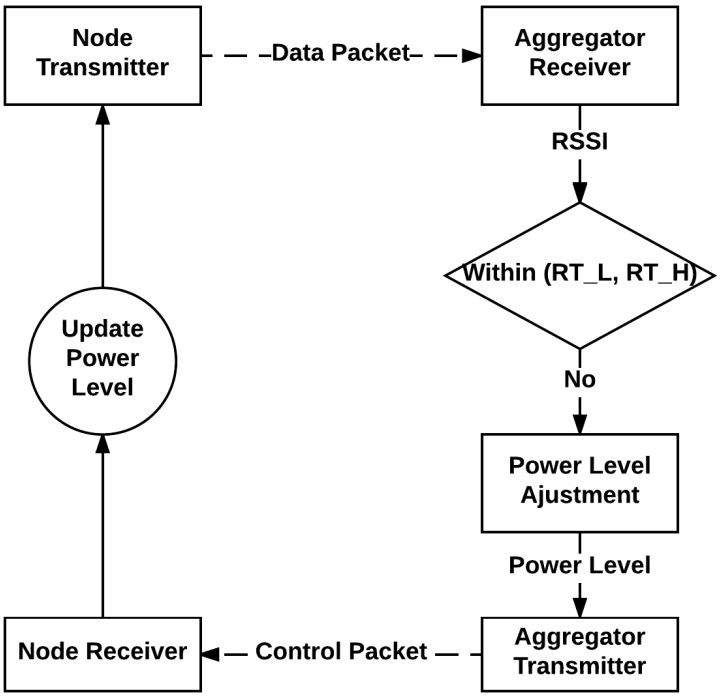


Figure 4.11: Power adaptation process.

the larger the interference range of a BSN. In addition, higher power level also means more energy consumption which is a major concern for BSNs. Therefore, ideally we should use the minimum amount of power needed to overcome body shadowing effect to minimize interference and energy cost.

The obvious approach is to start from a low power level, then gradually increase transmission power until we reach acceptable PRR. However, this process of transmitting, waiting for acknowledgments, and adjusting transmission power consumes precious time slots. Moreover, the frequent locomotion and activity/posture changes of a BSN wearer make BSN links highly variable, thus the transmission power needs adjustment frequently.

In our solution, we try to optimize the power adjustment process by using a reasonable starting power level, and then adjusting transmission power according to the RSSI of received packets.

According to the experiments done in Chapter 3, in indoor environment power level 11 is enough to achieve acceptable PRR, so we use power level 11 as our start power level.

Figure 4.11 shows the power adaptation process. When the aggregrator receives a data packet from a node, if the RSSI value is not in the range of RT_L and RT_H, the aggregator sends power adjustment message to the node to increase or decrease the transmission power in a linear way. There are some work [123] trying to improve the power adaptation process by using binary search to faster

find the next appropriate power level when people are moving around. But this assumes the presence of accelerometers on the nodes, which will make our system less generic without providing any major benefits. Therefore, in our system we choose to use a simpler linear adjustment process.

4.8 Discussion

In this chapter, we propose a QoS solution for multi-body, multi-function BSNs. In our solution, nearby BSNs are grouped together and their traffic is coordinated using different channels and time slots to avoid interference both from nodes of the same BSN and nodes of nearby BSNs.

The use of profiling not only enables our solution to compute possible BSN settings that can satisfy the fidelity requirements of multiple BSNs, but also separates BSN hardware details from application software requirements. Moreover, the use of REST APIs provides an easy and standardized way for BSNs and applications to register and mange themselves without worrying about adjusting QoS level or hardware level details.

During channel selection and bandwidth allocation, we implicitly assume that one channel is enough to accommodate the traffic of a BSN. This is reasonable since most existing BSN systems only require the aggregrator operating on one single channel.

Our solution doesn't guarantee there is always a BSN setting that can satisfy all the requirements, e.g. if one application requires too many packets such as one thousand to be sent per second, our delay-aware real-time scheduling will not work, since the minimum delay is too small and the channel utilization is too high. But in reality our solution works for most applications with reasonable configurations.

To guarantee link quality and reduce interference at the same time, we use power adaptation to automatically increase or decrease transmission power. This does not only reduce energy consumption, but also guarantees the reception signal strength at all aggregators is within the same range, which saves grouping from capture effects.

Our solution can also be easily adjusted to suit specific needs. For example, the RSSI vector can be sent in different rates to adjust the responsiveness of the grouping process, the preference of a BSN setting calculated in Algorithm 5 can be substituted to address different priorities, the delay-aware scheduling can be changed to used dynamic priorities such as Earliest Deadline First (EDF) to make sure scheduling is feasible as long as the channel utilization $U \leq 1$.

Chapter 5

Grammar-Based Fall Detection

5.1 Introduction

Chapter 4 discussed how to achieve reliable data transmission in BSNs. Besides reliable communication, how to develop BSN applications to reliably process these data is also of great importance. In this chapter we develop a grammar-based fall detection application. Using grammar to define events makes our algorithm extensible, which is very useful for detecting events without an common definition such as falls.

Falls are one of the most detrimental events for the aged population. About one out of three senior people 65 years or older have at least one fall per year, and falls are the leading cause of injury-related hospitalization for elderly people [50]. Detecting falls accurately can reduce the severe consequences, and thus is of great importance.

However, falls are difficult to accurately detect due to three reasons. First of all, certain fall-like activities are hard to distinguish from real falls. Since existing solutions mainly use accelerometers to detect falls, activities with larger accelerations such as sitting down quickly, cause many false positives when detecting falls. Second, falls have different characteristics according to their causes. For example, falls caused by environmental hazards and gait problems usually happen faster than those caused by dizziness and drop attacks. Third, unlike activities of daily living (ADL), falls are accidents, so it is extremely difficult to collect data of real falls of the elderly. Therefore, it is important to have a generic fall detection framework which can be adjusted and extended as more real falls are reported.

To overcome the problems of existing fall detection methods discussed above, in this chapter, we

present a grammar-based, posture- and context-cognitive fall detection framework that can detect falls with different activity levels. Our framework can be easily adjusted to detect falls more accurately as people gradually collect more data of real falls. Our fall detection framework makes the following contributions:

- *Our method can be easily tuned and extended.* Falls are difficult to detect accurately because there are so many different types of them. In our framework, we propose a context-free grammar to define various falls as sequences of sensor events, so that our system can be easily extended to handle new types of falls. The formal definition and generic detection framework for falls have not been studied before.

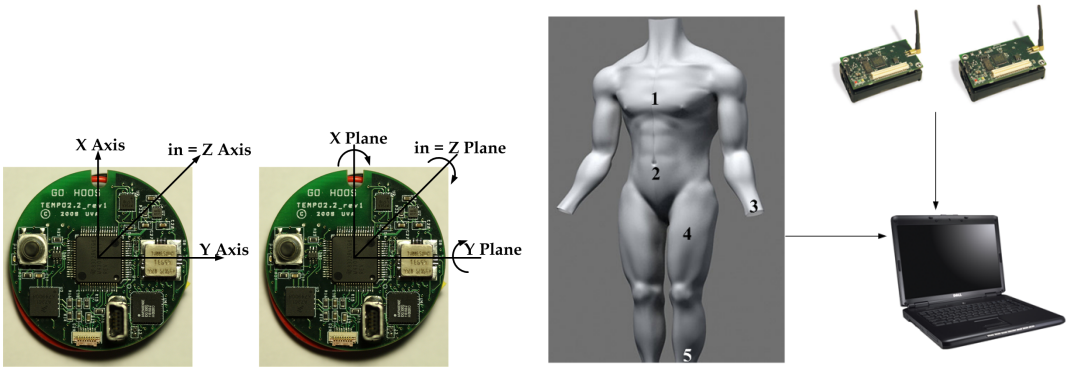
- *Our method is posture-cognitive and person-specific.* The posture and movement levels are determined in a novel way by using clustering and learning techniques, which makes our method person-specific. Training on posture and movement levels (not on falls) only requires several minutes.

- *Our method reacts to consequences of falls.* When evaluating the fall process, our method evaluates the immediate consequence of all falls — resting on the ground, combined with a period of low-level activities. If we detect the condition, we use the previous 5 seconds of data to determine if a fall actually occurred.

- *Our method is context-cognitive.* Environmental sensors attached on a bed or couch are used to eliminate false positives caused by fall-like activities also ending with a lying posture such as lying on the bed quickly.

- *Our method provides a mechanism to detect falls with different activity levels.* According to Rubenstein et al.[56], falls in elderly are caused by several different reasons such as environment-related accidents, gait and balance disorder, and dizziness and vertigo. Falls caused by environmental factors usually happen faster than falls caused by dizziness. By using a context-free grammar to define different kinds of falls, our method detects both fast and slow falls. The detection of slow falls has not been studied by previous work.

- *Our method achieves high accuracy.* In our evaluation, our method detects all 32 fast falls and 22 out of 24 slow falls from normal or fall-like activities.



(a) (b)

Figure 5.1: (a) The TEMPO 3.1 sensor node; (b) The data acquisition system setup.

5.2 Data Acquisition

Two kinds of sensors are used in our fall detection system: the TEMPO 3.1 nodes [3] and the widely used MICAz motes with MTS310 sensor boards. The TEMPO nodes are used to monitor the acceleration and rotational rates of different parts of the body, and the MICAz motes are used to monitor the vibration of specific furniture such as a bed to retrieve the subject's location context information.

The TEMPO 3.1 node includes a tri-axial accelerometer and a tri-axial gyroscope as shown in Figure 5.1(a). The MMA7261QT tri-axial accelerometer, made by Freescale Semiconductor, can monitor acceleration within a range of $\pm 10g$. The tri-axial gyroscope consists of an InvenSensce IDG-300 dual-axis gyroscope and an Analog Devices ADXRS300 Z-axis gyroscope. The IDG-300 can monitor angular velocity between $\pm 500°/s$. The ADXRS300 can monitor angular velocity between $\pm 300°/s$. The sensors are controlled by an TI MSP430F1611 microcontroller. The sampling rate is set to 120Hz, a bandwidth exceeding the characteristic response of human activities, to guarantee body movement details can be captured.

The MTS310 sensor board features a bi-axial accelerometer, which can monitor acceleration between $\pm 2g$. The sampling rate is set to 2Hz, which is sufficient for monitoring furniture vibrations.

Figure 5.1(b) shows the setup of our data acquisition system. In our experiments, we attach 5 TEMPO nodes to human body as shown in Figure 5.1(b). Using 5 nodes enables us to collect significant amounts of data to obtain very high accuracy and determine which nodes and locations are most critical. Due to space limitations we do not show these performance results, but we see that at least 3 nodes are required (chest, ankle, and wrist). The extra 2 nodes do improve performance and in the future when such nodes can be unobtrusively embedded in clothes using 5 nodes (adding

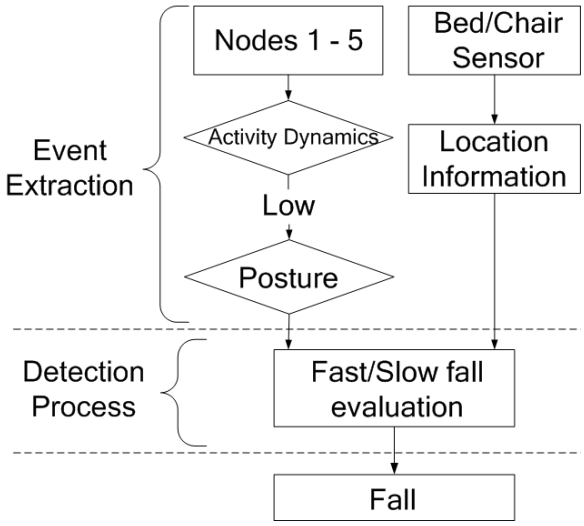


Figure 5.2: The main process to detect falls: posture recognition, localization, and fall process evaluation for both fast and slow falls.

on the thigh and waist) would be preferred.

The MICAz nodes are fixed to furniture including a bed and a chair. The subject is located by monitoring the vibration of the furniture.

To make sure our system works for a wide variety of situations, during the following experiments, three graduate students engaged in a battery of tests designed to simulate falls (fast fall forward/backward/leftward/rightward/ag-ainst wall and slow falls), fall-like activities (sit down fast upright, sit down fast reclined, jump into bed, stumble, jump), and normal activities (stand, sit, lie, walk, run). All fall data was taken on hard surfaces.

5.3 Fall Detection Framework

5.3.1 The Fall Detection Process

In our fall detection framework, we use a context-free grammar to define various falls as sequences of sensor events. These events can be the raw readings from on-body sensors exceeding thresholds, or more high-level information, such as the posture and location of the subject, inferred from the low-level sensor readings. As shown in Figure 5.2, the detection process can be divided into two main steps: *event extraction* and *fall process evaluation*.

During event extraction we buffer a short period of recent sensor readings and calculate high-level features. These raw readings and derived features are used later during fall process evaluation to determine if a fall has happened. In our current system, we calculate two kinds of high-level features:

Table 5.1: The production rules of the context-free grammar for defining fall processes.

$$F \rightarrow S \tag{5.1}$$
$$S \rightarrow E|ETS \tag{5.2}$$
$$E \rightarrow (E,E)|(SENSOR,FEATURE)|P|LOC \tag{5.3}$$
$$T \rightarrow (time,C)|(time,time)|\varepsilon \tag{5.4}$$
$$SENSOR \rightarrow chest|waist|wrist|thigh|ankle \tag{5.5}$$
$$FEATURE \rightarrow (threshold,C)|(threshold,threshold) \tag{5.6}$$
$$C \rightarrow < | > \tag{5.7}$$
$$P \rightarrow standing|sitting|lying \tag{5.8}$$
$$LOC \rightarrow bed|couch|other \tag{5.9}$$

Table 5.2: The meanings of non-terminals in the context-free grammar.

Non-terminal	Meaning
F	a fall process
S	a sequence of sensor events
E	a sensor event
T	the time interval between two sensor events
$SENSOR$	which sensor the readings are collected from
$FEATURE$	the feature of the collected readings
C	a comparison with time interval or threshold
P	the subject's posture
LOC	the subject's location

the posture and location of the subject. Unlike previous work using preset inclination thresholds [71] to determine postures, no preset thresholds are used in our system when calculating the posture of the subject by using clustering and learning techniques. Therefore, our posture recognition algorithm adapts to different users automatically. The location of the subject is detected by the sensors attached on bed and couch.

The key part of the fall process evaluation is a set of rules. Each rule defines a type of fall using the context-free grammar so that it can be parsed automatically. Then we compare the rules against the readings and features collected in the previous step to determine if a fall has happened. In this step we evaluate both fast falls and slow falls according to their respective rules.

In the following of this section, we first present the context-free grammar used to define falls, then discuss the extraction of posture information from on-body sensor readings, and discuss how to retrieve context information from environmental sensors. Last we present the fall process evaluation.

Table 5.3: The meanings of expressions in the context-free grammar.

Expression	Meaning
(E, E)	two events happen at the same time
$(time, C)$	time interval between two events is smaller ($<$) or larger ($>$) than $time$
$(time, time)$	time interval between two events is between two values of $time$
$(threshold, C)$	sensor readings are smaller ($<$) or larger ($>$) than $threshold$
$(threshold, threshold)$	sensor readings are between two values of $threshold$

5.3.2 Context-Free Grammar for Defining Falls

In our framework, a fall is defined as a sequence of sensor events. This sequence can be generated using our context-free grammar shown in Table 5.1. The meanings of the non-terminals of the grammar are shown in Table 5.2. Some of the expressions in Table 5.1 also have special meanings as shown in Table 5.3.

Rule 5.3 of the grammar shows that the sensor events used to define falls can be divided into two categories: sensor readings exceeding *threshold*s (or within two *threshold*s) and high-level features derived from sensor readings including the posture (P) and location (LOC) of the subject. The initial *threshold*s used in specific rules are determined based on simulated falls as shown later in Section 5.3.5, and Section 5.3.5. Our framework easily permits adjusting the *threshold*s if needed.

To demonstrate the usage of the proposed grammar, we use it to define several falls and fall-like activities:

Normal fast fall:

$(chest, (threshold, >))(lying, other)$

Forward slow fall:

$(thigh, (threshold, >))(2s, <)$

$(wrist, (threshold, >))(lying, other)$

Sitting down fast to couch:

$(waist, (threshold, >))(sitting, couch)$

Lying onto bed quickly:

$(chest, (threshold, >))(lying, bed)$

From these examples, we can see it is straightforward to define falls and fall-like activities using the grammar.

Our framework can be extended easily in two ways:

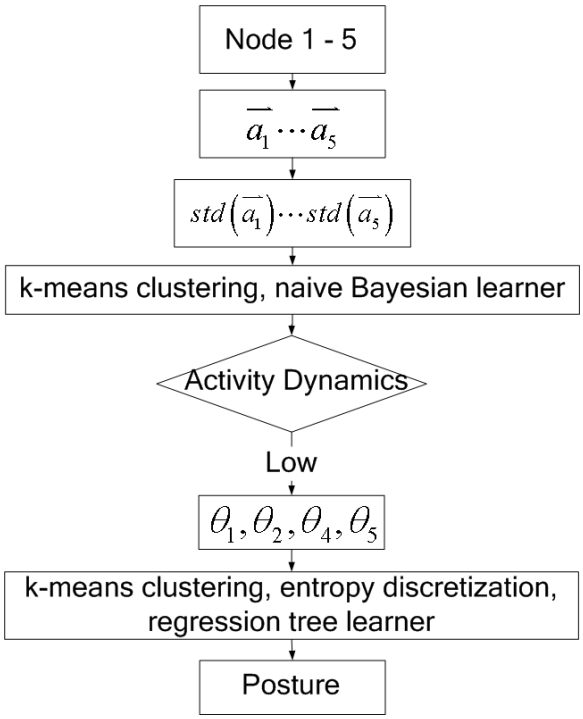


Figure 5.3: The process of posture recognition.

- *Extending the grammar.* In actual deployments, more sensors can be used to get additional information. For example, we can attach the MICAz node on a shower head to detect if the subject is showering. In this case, *showering* can be added as a new terminal in Rule 5.3.

- *Adding new rules.* Under a given grammar such as in Table 5.1, to detect a new kind of fall or fall-like activity, we only need to write a new rule using this grammar, and then the rule can be parsed automatically so that the system will be able to recognize this specific type of fall or fall-like activity.

5.3.3 Posture Recognition

There are many different types of falls, but most of them have the same immediate consequence — lying on the ground, combined with a period of low-level activities. Therefore, posture is an important feature to detect falls. Figure 5.3 shows the process of posture recognition in our framework: we first divide activities into three categories according to their dynamic levels, then we extract posture information when the subject is in low-level dynamic activities. During the process, by using the clustering and learning techniques, we do not need to use preset thresholds to detect postures.

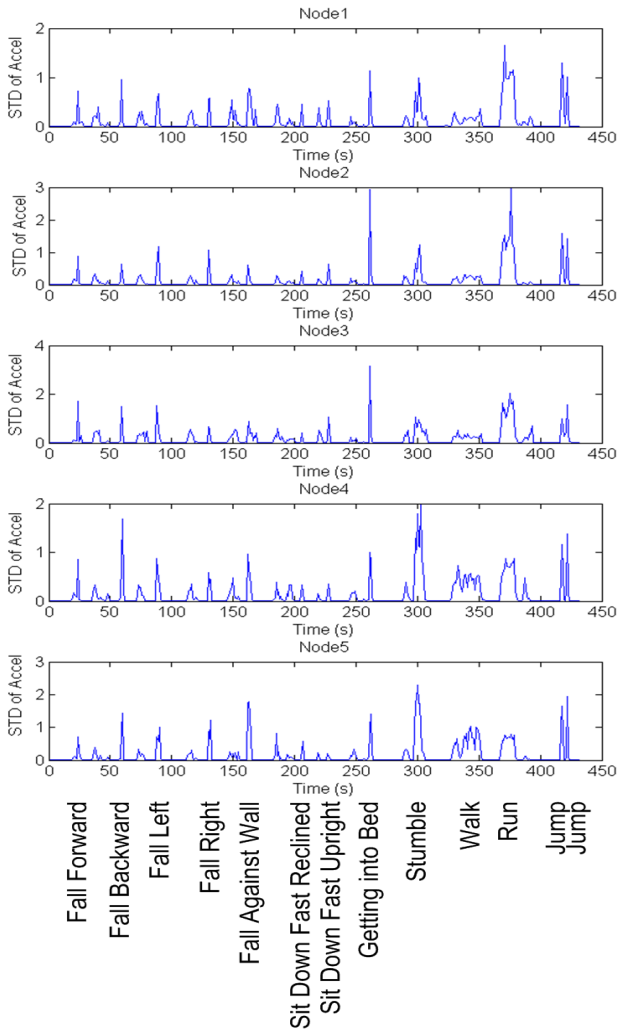


Figure 5.4: The standard deviation of the linear accelerations at the chest (Node 1), waist (Node 2), wrist (Node 3), thigh (Node 4), and ankle (Node 5) for various activities.

Activity Dynamic Level

We use the standard deviation of the acceleration readings from TEMPO nodes to determine the person's activity level. The monitoring cycle is one second, which means we calculate the standard deviation of the collected data every 120 samples.

The acceleration of each sensor can be calculated using Equation (5.10), where a_i is the vector magnitude linear acceleration of Node i, and a_{i_x}, a_{i_y}, a_{i_z} are the acceleration readings along the x-, y-, and z-axis.

$$a_i = \sqrt{a_{i_x}^2 + a_{i_y}^2 + a_{i_z}^2} \quad (i = 1, 2, \cdots, 5) \tag{5.10}$$

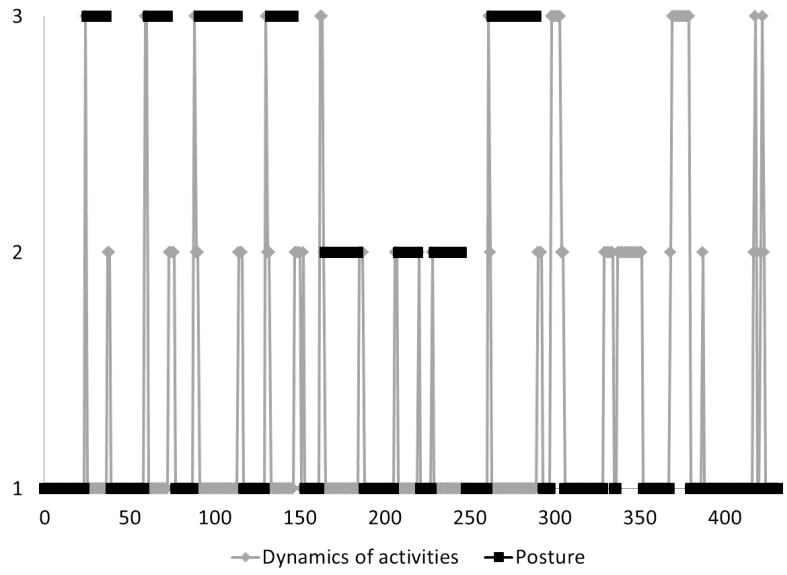


Figure 5.5: Gray line: clustering results of dynamic levels of different activities (1: Low; 2: Medium; 3: High); Black line: posture clustering results when activity dynamic levels are classified as low (1: standing; 2: sitting; 3: lying).

Figure 5.4 shows the standard deviation of the accelerations at different parts of the body when performing various activities. From Figure 5.4, we can see some activities like running are more dynamic than others like walking or standing still. Therefore, according to different standard deviation values, we divide human activities into three categories: high dynamic activities like running or jumping, medium dynamic activities like transactions between postures or walking, and low dynamic activities like standing, sitting, or lying.

Instead of using simple thresholds to determine activity categories, we use k-means clustering to partition activity dynamics into three categories. The gray line in Figure 5.5 shows the results of cluster analysis. The input data of the cluster analysis is the standard deviations of the acceleration as shown in Figure 5.4. From Figure 5.5 we can see that the results of clustering activity levels of the activities shown in Figure 5.4 are all correct: falling forward/backward/left/rig-ht/against wall, getting into bed, stumbling, running, and jumps are clustered as high dynamic activities (Class 3); activities including sitting down fast, walking, and the transactions between activities are clustered as medium dynamic activities (Class 2); when the person is standing, sitting, and lying, the activity dynamics are clustered as low (Class 1). Based on the clustering results, we build a naive Bayesian classifier to differentiate activity dynamic levels.

Extracting Posture Information

When activity dynamic levels are classified as low during one second, we recognize the postures (standing, sitting, or lying) of this one second time period to determine if there is possibly a fall.

In our experiment, we first perform an initial calibration for the stationary standing posture. In this step, we record the accelerometer readings from Nodes 1, 2, 4 and 5 as shown in Figure 5.1(b). Node 3 is not used because it is attached on the wrist and does not strongly relate to postures. Then we calculate the angle changes of each node based on the accelerometer readings along each axis using Equation (5.11).

$$\theta_i = \frac{180}{\pi} \arccos\left(\frac{\hat{a}_{i_x} a_{i_x} + \hat{a}_{i_y} a_{i_y} + \hat{a}_{i_z} a_{i_z}}{\sqrt{\hat{a}_{i_x}^2 + \hat{a}_{i_y}^2 + \hat{a}_{i_z}^2} \cdot \sqrt{a_{i_x}^2 + a_{i_y}^2 + a_{i_z}^2}}\right) \tag{5.11}$$

In Equation (5.11), $i = 1, 2, 4, 5$, θ_i is the orientation change of Node i, $\hat{a}_{i_x}$, $\hat{a}_{i_y}$, $\hat{a}_{i_z}$ are the acceleration readings of Node i during the calibration phase along the x-, y-, and z-axis, and a_{i_x}, a_{i_y}, a_{i_z} are the mean values of the accelerometer readings along each axis during low dynamic activities.

The black line in Figure 5.5 shows the results of posture clustering. Note that posture clustering results are only available when activity dynamic levels are low. In this step, we use k-means clustering to partition postures into three categories: standing (Class 1), sitting (Class 2) and lying (Class 3). The input data for posture clustering is the vector $(\theta_1, \theta_2, \theta_4, \theta_5)$. In Figure 5.5, all postures are clustered correctly. Based on the results we use a regression tree learner to build the posture classifier, which is used to recognize postures.

In Section 5.3.3 and Section 5.3.3, we first determine the activity level of the subject, then extract posture information if the subject is in low activity level. The learning process for posture recognition only requires a few minutes, and during this period the subject performs normal daily activities and three postures (standing, sitting, and lying). Unlike most previous work, by using clustering and learning techniques in our method, the whole posture recognition process can be automatically adjusted across different subjects, and no predefined thresholds are used. This makes our method person-specific.

5.3.4 Context Information Collection

Some activities like lying onto the bed quickly are extremely difficult to distinguish from real falls only using body sensors. However, by combining context information such as the location of the subject, these fall-like activities can be distinguished from real falls easily: when the on-body sensor

Figure 5.6: Collect readings from MICAz motes attached on bed and couch

readings indicate a fall has happened, if the location of the subject is on the bed, then it is obviously a false alarm. Sensors in environment are becoming common in assisted living systems, e.g. GE's QuietCare[80] uses environmental sensors to learn residents' routine. In our system, we collect context information using ambient environmental MICAz motes (as shown in Figure 5.6).

Figure 5.6 shows a typical deployment of environmental sensors to retrieve location information. In this setting, one MICAz mote (with MTS310 sensor board) is attached onto the bed, the other is attached onto the couch. Every half second each MICAz mote sends x- and y-axis acceleration readings back to the sever. The right of Figure 5.6 shows the data collected from these sensors. If the deviation of the collected data is larger than a threshold value (we use 100 in our experiments), we know the subject is just sitting down to the couch or lying onto the bed.

In fact, many other kinds of context information can be collected from environmental sensors. For example, the assisted living system AlarmNet [26] uses X10 sensors to track which room the subject is in. By utilizing the data collected by these assisted living systems, we can have more knowledge about the subject, which can help improve fall detection accuracy.

5.3.5 Fall Process Evaluation

After knowing the subject is resting on the ground with low-level movements (Section 5.3.3), and not on a bed or in the couch (Section 5.3.4), our method checks the data for 5 seconds earlier to evaluate whether the process that achieved this posture is a fall. We do not use learning techniques during fall process evaluation because falls are rare accidental events and it is not feasible to train the system

by letting elderly people fall. The *threshold*s used for the fall process evaluation are determined by simulated falls.

Falls have different characteristics according to their causes. Rubenstein et al. [56] summarized major causes of falls in elderly people and their relative frequencies. The first four causes for falls in elderly are: falls stemming from environmental hazards (31%), gait/balance disorder (17%), dizziness and vertigo (13%), and drop attacks (9%). In these categories, falls caused by environmental hazards and gait problems usually happen faster than those caused by dizziness and drop attacks. However, previous work on fall detections discussed in Section 2.3 only focuses on fast falls.

In this section, we show that both fast and slow fall processes can be defined and detected using the production rules in our framework.

Fast Falls

In the examples in Section 5.3.2, we define fast falls as

$$(chest, (threshold, >))(lying, other).$$

Though this definition is enough as a demonstration of the usage of the proposed grammar, in a real fall detection system, we need to figure out which *LOC* is more useful for detecting a fall, and what the specific value of the *threshold* should be. To solve this problem, three graduate students performed five kinds of fast falls (fall forward/backward/left-ward/rightward, fall after stumbling) and typical normal daily activities (walk, sit down, lie down) two to three times.

Figure 5.7 shows the maximum acceleration of each node during different activities. From Figure 5.7, we can see that in spite of some overlap the max acceleration during a fall is most of the time much larger than during normal activities. Therefore, as shown here and in many previous works we can use preset thresholds to detect fast falls. Nodes attached on chest (Node 1), waist (Node 2), and thigh (Node 4) are more useful because their max accelerations have almost no overlap with those of normal activities. However, accelerations of nodes attached on wrist (Node 3) and ankle (Node 5) often have overlap between different activities because of impulsive movements, thus they are not suitable for detecting fast falls. In our method, a fast fall process is detected if $a_1 > 3.0g$ and $a_4 > 3.0g$. Therefore, the rule used to detect fast falls can be written as Equation (5.12).

$$((chest, (3.0g, >)), (thigh, (3.0g, >)))(lying, other) \tag{5.12}$$

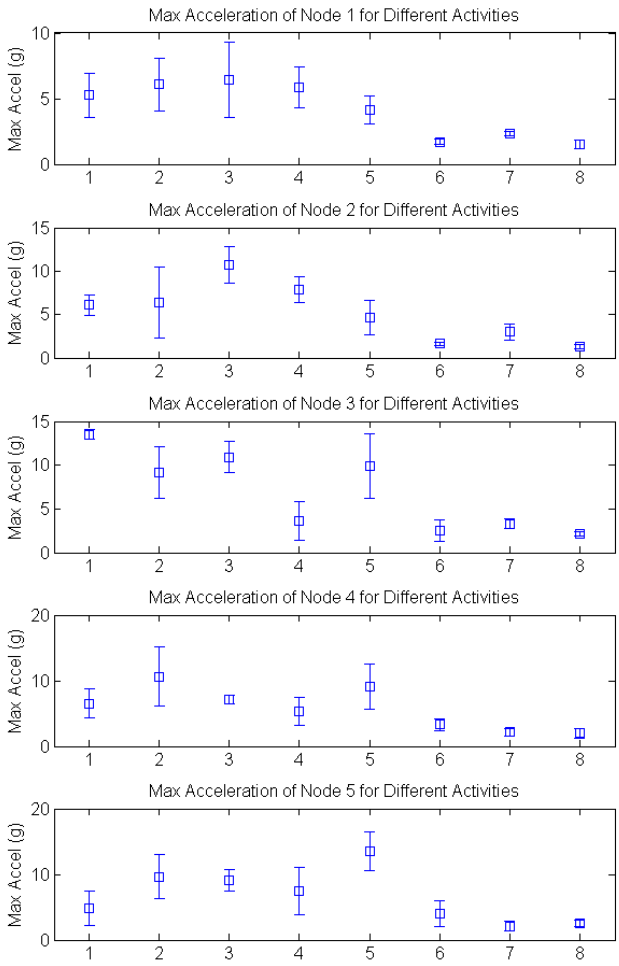


Figure 5.7: The max acceleration of nodes for different activities: 1) fall forward; 2) fall backward; 3) fall leftward; 4) fall rightward; 5) stumble and fall; 6) walk; 7) sit down; 8) lie down.

Slow Falls

Besides fast falls, falls caused by dizziness and drop attacks are also common in elderly people. Drop attacks are falls associated by sudden leg weakness, but without dizziness. For elderly people, lacking exercise results in poor muscle condition, decreased strength, and loss of flexibility. Therefore, when bending, reaching, or rising from a chair or bed, elderly people are prone to falls.

These falls are not as sudden as fast falls, and the accelerations of these falls are not as large. Therefore, it is hard to distinguish them only using acceleration thresholds. Figure 5.8 shows two simulated slow falls: the person rose from a chair, then fell forward to the ground before he could stand steadily. During the fall process, the accelerations of both Node 1 and Node 4 are under the black line ($3.0g$) and thus not large enough to be detected as a fall according to Equation (5.12).

However, in this kind of fall, people usually push their hands out to cushion the body, so there are

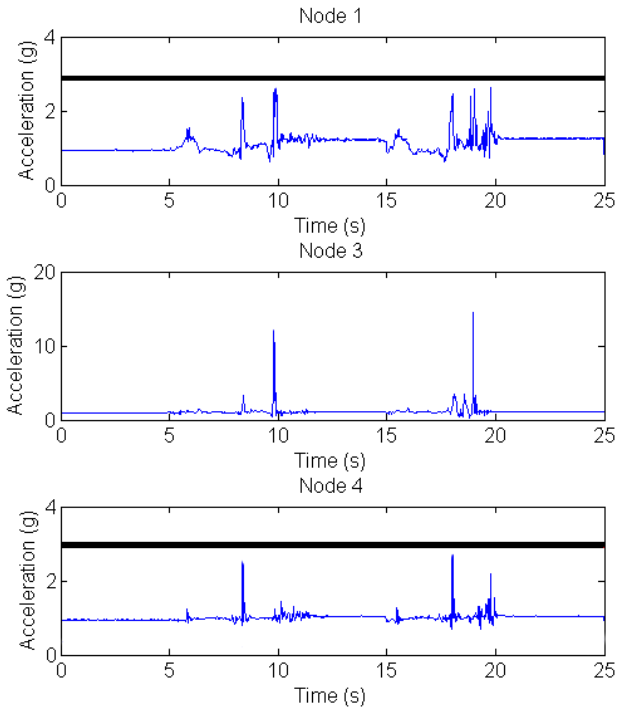


Figure 5.8: Accelerations of Node 1 (chest), Node 3 (wrist), and Node 4 (thigh) for a slow fall caused by dizziness.

large acceleration readings from wrist (Node 3) as shown in Figure 5.8. Thus the fall process can be detected as a sequence of sensor events: sitting posture, large accelerations from wrist, and resting on the ground with low-level movements. Usually the movement of wrist is not suitable for fall detection because its impulsiveness, however, as an event of a sequence, it can be useful. Therefore, the rule used to detect this kind of slow falls can be written as Equation (5.13).

$$sitting(wrist, (8.0g, >))(lying, other) \tag{5.13}$$

5.4 Evaluation

In this section, we evaluate our fall framework by studying two special cases and performing system integration tests. By discussing the first special case, we show how to use context information to distinguish fall-like activities from real falls. In the second case study, we show how our system deals with slow falls. The system integration test then shows the overall performance of our fall detection system.

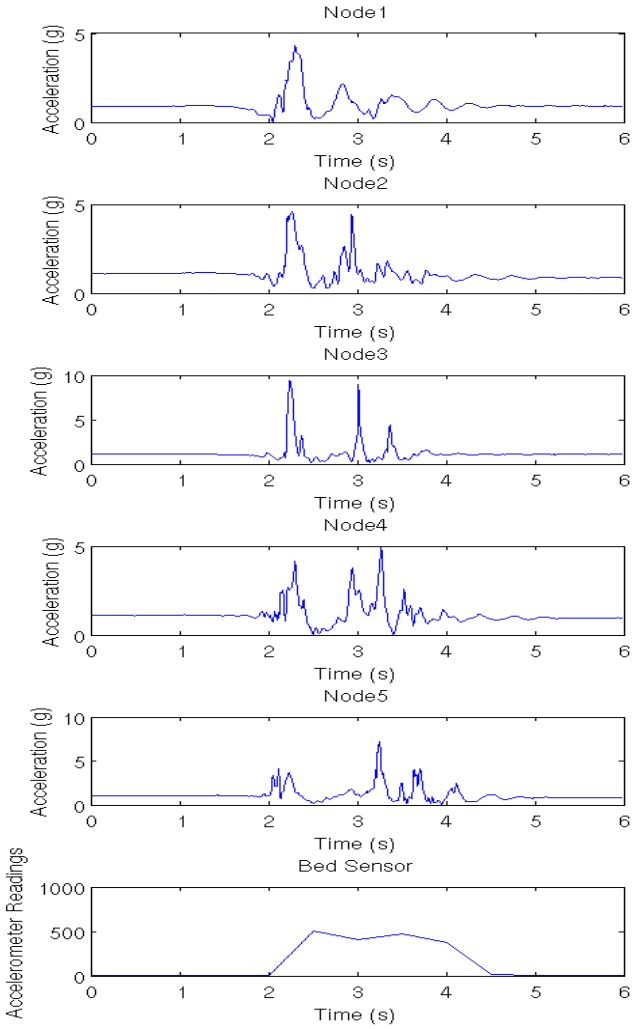


Figure 5.9: Accelerations of Node 1 (chest), Node 2 (waist), Node 3 (wrist), Node 4 (thigh), Node 5 (ankle), and bed sensor for lying onto bed quickly.

5.4.1 Special Case Study

Lying onto Bed Quickly

Lying onto the bed quickly is extremely difficult to distinguish from real falls, because they all feature large accelerations and fast body orientation changes. The first five plots in Figure 5.9 show six seconds of acceleration from Node 1 to Node 5 when the subject lies onto the bed quickly. In Section 5.3.5, we use $a_1 > 3.0g$ and $a_4 > 3.0g$ as thresholds to detect fast falls. However, in Figure 5.9 we can see every node has max acceleration larger than $3.0g$, so most previous work only based on monitoring acceleration cannot handle this fall-like activity correctly.

However, distinguishing lying onto bed from real falls becomes straightforward if we know the location of the subject. The last plot in Figure 5.9 shows the accelerometer readings of the MICAz

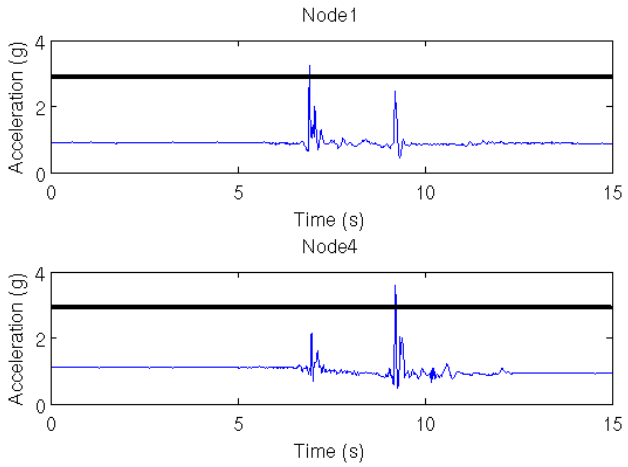


Figure 5.10: Accelerations of Node 1 (chest) and Node 4 (thigh) for falling against wall

sensor attached to the bed. When the subject gets into bed, the bed sensor shows large readings (from 2s to 4s), and the activity will not be detected as a fall in our system.

Fall against Wall

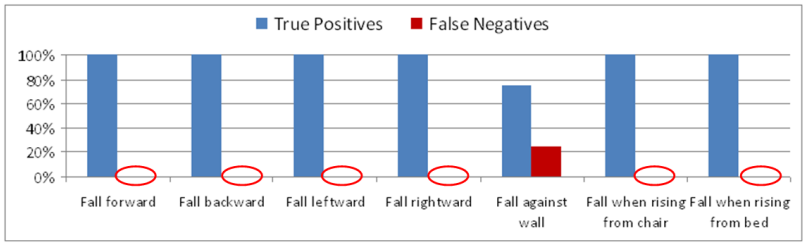
In this section, we use an example to illustrate how to detect a new type of fall — fall against a wall — using our system. In this kind of fall, the subject first touches the wall for support, then slips down to the ground, and ends with a sitting position. Figure 5.10 shows typical accelerometer readings from Node 1 (chest) and Node 4 (thigh) when falling against a wall. When the subject touches the wall, the acceleration of Node 1 (chest) is lager than that of Node 4 (thigh). However, when the subject slips down to the ground, the acceleration of the thigh is larger. Based on this observation, we can define a fall against a wall as

$$((chest, (threshold, >)), (thigh, (threshold, <)))$$
$$((chest, (threshold, <)), (thigh, (threshold, >)))$$
$$(sitting, other)$$

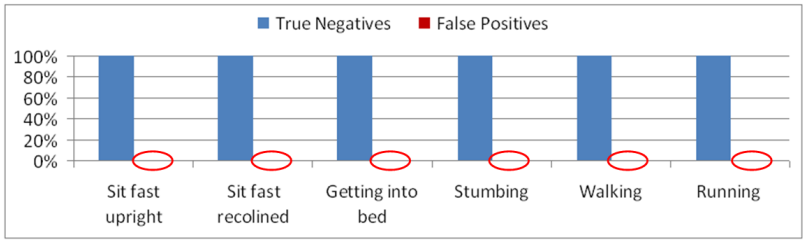
From our experiments, using $3.0g$ as the value of $threshold$ can detect falls against a wall.

5.4.2 System Integration Tests

To evaluate our algorithm as a complete system, three students performed a series of activities: simulated each kind of falls 8 times (falling forward/backward/left/right/against wall, falling when rising from chairs/bed), 56 falls were simulated in all; also the students performed other fall-



(a)



(b)

Figure 5.11: (a) True positive performance (sensitivity); (b) True negative performance (specificity).

like activities and normal activities including sitting down fast upright/reclined, getting into bed, stumbling, walking, running, and jumping. Each fall-like and normal activity was performed 2 times by each person.

Figure 5.11 shows the sensitivity (true positive performance, the number of detected falls divided by the total number of falls) and specificity (true negative performance, the number of detected fall-like or normal activities divided by the total number of these activities) of our fall detection framework.

Figure 5.11(a) shows the detection results of both fast falls and slow falls: fast falls include falling forward, backward, leftward and rightward; slow falls include falling against a wall, falling when rising from a chair or bed. All fast falls are detected correctly. Only two slow falls (against wall) out of all 24 slow falls are not detected using our method. This is because we recognized the sitting posture after the fall against a wall as a lying posture. By contrast, 20 out of all 24 slow falls (8 falls against wall, 5 falls when rising from chair, 7 falls when rising from bed) are not detected only using thresholds ($3.0g$) for the accelerations of Node 1 (chest) and Node 4 (thigh).

Figure 5.11(b) shows the detection results of fall-like activities (including sitting down fast ending with an upright or recolined position, getting into the bed quickly, and stumbling) and some normal daily activities (including walking and running). Our method distinguishes all these activities from falls. For fall-like activities such as sitting down and getting into the bed, the environmental sensors play a key role to eliminate the false alarms. The ending posture (either lying or sitting, definitely

not standing) in the definition of falls distinguishes stumbling from real falls.

5.5 Discussion

In this chapter, we proposed a grammar-based fall detection framework which uses both posture and context information, and can detect both fast and slow falls. By using a context-free grammar to define various falls as sequences of sensor events, our framework can be easily tuned and extended to detect new types of falls.

The use of clustering and learning techniques to recognize activity levels and postures makes our method person-specific and increases the recognition accuracy. The training only requires subjects performing several minutes of normal daily activities.

In our solution high-level features such as posture and location information are utilized to enable our method to react to the consequence of falls — subjects resting on the ground, combined with low-level activities. The ability to combine features extracted from both on-body and environmental sensors improves our detection accuracy.

Slow falls, which are common in the elderly yet have not been studied by previous work, can also be effectively detected using our method. Evaluation shows our method can distinguish most falls from normal activities correctly.

However, currently only common slow falls are discussed, other slow falls need to be handled by adding new rules in the future. In addition, irregular physiological data like low blood pressure is very useful to help detect falls, as more types of sensor are integrated to BSNs, our framework can be extended to achieve better fall detection accuracy.

Chapter 6

In-Person Interaction Monitoring

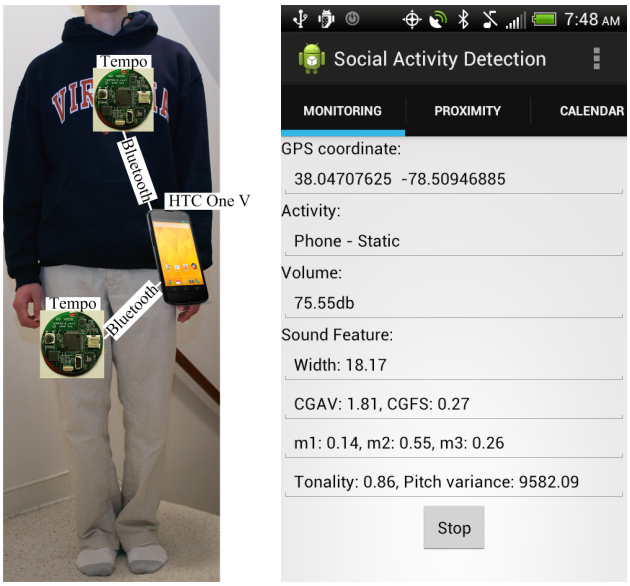
6.1 Introduction

According to the U.S. Department of Health and Human Services [1], the population of 65 and older reached 46.2 million in 2014. By 2040, older population is projected to be 82.3 million. However, 15%-20% of senior people have significant depressive symptoms [81]. Many studies show that lacking in-person interactions plays an important role in the initialization and development of depression [82] and is one of the most important indicators of physical and mental health in aging patients [83]. Therefore, in-person interaction monitoring is of great importance, because it enables psychiatric clinicians and geriatric professionals to perform more accurate diagnosis and treatment of psychological problems for elderly people.

Plenty of existing work targets people's daily activities, but they are not well poised to detect in-person interactions, because: 1) most activity detection methods such as [101] use very limited types of sensors, usually only accelerometers, and lack the multitude of sensing modalities needed to detect rich interactions between people; 2) existing solutions such as [91, 92, 93, 94] detect human interactions solely based on data collected from one person, and ignore the involvement of multiple participants, which is the intrinsic nature of in-person interactions.

These problems limit existing systems from extracting in-person interaction details such as the involvement of a person in a conversation. Yet this detailed information is critical for psychiatrists or caregivers to monitor and assess people's mental health, as indicated in gerontological studies [82, 95].

To overcome these limitations, in this chapter, we propose a multi-modal in-person interaction monitoring system using smartphones and on-body sensors. Our system first detects seven kinds



(a) (b)

Figure 6.1: System platform: (a) hardware; (b) mobile app user interface.

of primitives from *all participants* interacting with each other: activities of subjects, proximity information, presence of speech, speech volume analysis, GPS coordinates, phone call records, and calendar events. By analyzing these primitives, our system detects details about interactions. Our system suits best for facilities such as nursing homes where it can be deployed to all people and primitives from all participants of an interaction can be collected. It can also be used when people interacting with the subject do not have our system, because primitives collected from the subject such as activities, speech, and GPS are already sufficient to detect many interactions such as dining out though with less details.

Moreover, unlike some existing work [83, 109] requiring deployment of environmental sensors, our system only uses smartphones and optionally on-body sensors to detect in-person interactions. This significantly improves the mobility and thus practicality of our system, since many interactions of interest such as dining with others happen where environmental sensors are unavailable.

6.2 Data Acquisition

Inertial sensors (TEMPO nodes [3]) and a smartphone (HTC One V) are used in our social interaction detection system. The TEMPO nodes are attached to the chest and thigh to monitor the motion and posture of the body, collecting acceleration data at a sampling rate of 120Hz — sufficient to capture the characteristic response of human activities. The HTC One V phone records audio signals around

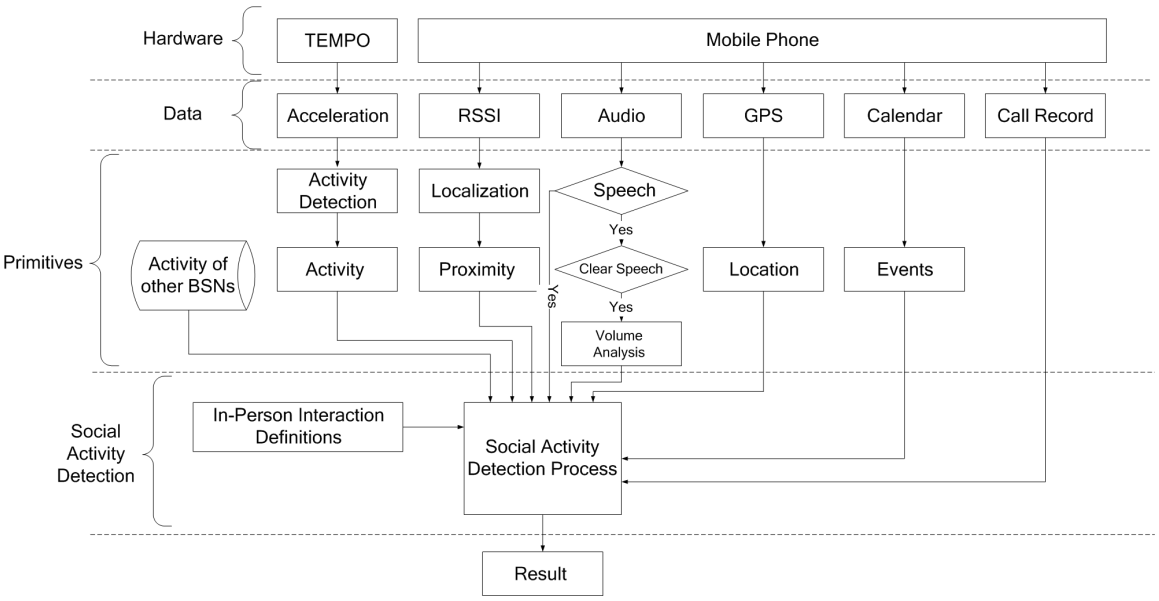


Figure 6.2: The framework to detect social interactions.

the subject, detects the proximity of other people, and collects GPS coordinates, calendar events, and phone call records. The phone also functions as a data aggregator for TEMPO nodes. The hardware platform is shown in Figure 6.1(a). The mobile app user interface is shown in Figure 6.1(b).

6.3 Multi-Modal In-Person Interaction Monitoring System

To overcome the shortcomings of existing solutions, we employ a multi-modal system to detect in-person interactions with detailed involvement information. The framework to detect in-person interactions is shown in Figure 6.2. The techniques employed in the framework is detailed below.

6.3.1 Primitives

From data collected our system calculates seven kinds of primitives to detect in-person interactions. These primitives are: 1) activities of subjects, including static postures (standing, sitting, and lying) and dynamic movements; 2) the proximity information of nearby BSNs; 3) the presence of speech; 4) speech volume analysis during clear speech; 5) location information from GPS coordinates; 6) phone call records; 7) calendar events.

Activity

In activity detection, first we determine whether a person is dynamically moving or statically posturing (dynamic vs. static) according to the variation of accelerations. If the person is staticcaly posturing,

Table 6.1: The average value and standard deviation of θ and $\hat{a}_y$ for different postures.

Posture	$mean(\theta)$	$std(\theta)$	$mean(\hat{a}_y)$	$std(\hat{a}_y)$
Standing	7.28°	2.81	$1.07g$	0.01
Sitting	71.77°	3.64	$1.01g$	0.01
Lying	13.58°	1.78	$-0.07g$	0.10

we use accelerations from TEMPO nodes to determine specific static postures: standing, sitting, or lying.

To distinguish specific static postures, we calculate the angle between the trunk and thigh, θ, using Equation (6.1), where $\hat{a}$ and a are the acceleration readings from chest and thigh, respectively.

$$\theta = \frac{180}{\pi} \arccos\left(\frac{\hat{a}_x a_x + \hat{a}_y a_y + \hat{a}_z a_z}{\sqrt{\hat{a}_x^2 + \hat{a}_y^2 + \hat{a}_z^2} \cdot \sqrt{a_x^2 + a_y^2 + a_z^2}}\right) \tag{6.1}$$

Table 6.1, which is obtained from three subjects performing static postures multiple times, shows the value of θ and $\hat{a}_y$ (gravitational vector at chest) for different postures. Then we can distinguish standing ($\theta < 45°$, $\hat{a}_y > 0.5g$), sitting ($\theta > 45°$, $\hat{a}_y > 0.5g$), and lying ($\theta < 45°$, $\hat{a}_y < 0.5g$).

To make our system more flexible, when TEMPO nodes are not present, our system automatically uses the accelerometer on the smartphone to distinguish dynamic vs. static. The system is easier to wear and deploy by excluding TEMPO nodes, but it also makes activity detection results unreliable since people do not always keep their phones at the same position. In addition, without TEMPO nodes our system does not distinguish specific static postures (standing, sitting, and lying), which are critical to detect specific kinds of interactions, e.g. it is an important feature of meetings that multiple people sit near to each other.

Proximity

Most radio devices provide Received Signal Strength Indicator (RSSI), which can be used to adequately estimate the distance between nodes when they are within several meters of each other , hence we use RSSI to detect proximity. In our system, we use the Bluetooth radio on the smartphone to detect the presence and estimate the distance of other BSNs in the vicinity.

The relation between distance and RSSI is described in Equation (6.2)[124], in which d is the transmitter-receiver distance, n is the attenuation constant, X_σ is a zero-mean Gaussian with standard deviation σ (multipath effects), and $RSSI_{ref}$ is the RSSI value at reference distance d_{ref}.

$$RSSI = RSSI_{ref} - 10nlog_{10}\left(\frac{d}{d_{ref}}\right) + X_\sigma \tag{6.2}$$

Table 6.2: Pre-measured $RSSI_{ref}$ and n for typical indoor and outdoor environments.

Environment	$mean(RSSI_{ref})$	$std(RSSI_{ref})$	n
Indoor	−61.56	3.08	2.51
Outdoor	−83.89	3.46	1.35

Usually n and σ are obtained through curve fitting of empirical data, but commonly we can assume X_σ to be 0 and distance can be obtained via Equation (6.3), in which $RSSI_{ref}$ is measured at $d_{ref} = 1$m.

$$d = 10^{\frac{RSSI_{ref} - RSSI}{10n}} \tag{6.3}$$

According to Equation (6.3), we need to do measurements to determine $RSSI_{ref}$ and n to extract exact distance from RSSI. However, since we only need rough estimation of proximity when detecting in-person interactions, using pre-measured $RSSI_{ref}$ and n are sufficient. Table 6.2 shows $RSSI_{ref}$ and n we measured for typical indoor and outdoor environments for up to 3 meters.

By extracting proximity information from Bluetooth radio, our system not only discovers nearby BSNs, but also roughly estimates their distance to the subject. Distance estimation is important because to monitor an interaction, we need to determine whether a BSN is a participant or not. In our system, we consider BSNs within 3 meters potential participants of an interaction.

Speech Detection

In our system, we classify audio signals recorded by a smartphone microphone into three categories: clear speech, speech in noise, and other sounds. The classification is based on audio features mentioned in [125].

We extract eight features divided to 3 groups from recorded audio signals. They are:

- Amplitude modulation features: *width*, $m1$, $m2$, $m3$. *width* characterizes the modulation depth in the signal. $m1$, $m2$, $m3$ are the modulation depth for the modulation spectra of the signal envelope in three modulation frequency ranges: 0-4Hz, 4-16Hz, and 16-64Hz.

- Spectral profile features: $CGAV$ (spectral center of gravity), $CGFS$ (fluctuations of the spectral center of gravity). $CGAV$ and $CGFS$ describe static and dynamic spectral profile, respectively.

- Harmonicity features: *tonality* and *pitchvar*. *tonality* is the ratio of harmonic (pitch is present) to inharmonic (pitch is absent) parts in the sound. *pitchvar* is the variance of pitch.

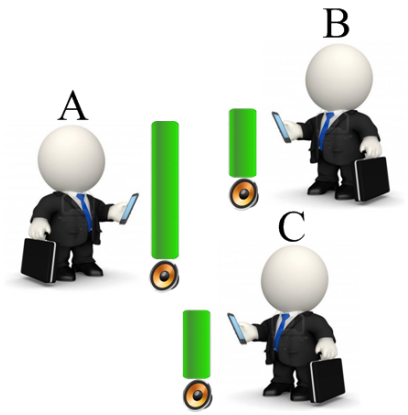


Figure 6.3: Conversation volume analysis when three people are talking with each other.

In our system, all features above are extracted on the smartphone in real-time every second. Using these features and bagged tree classifier, our system can detect 93.23% of clear speech and 85.85% of speech in noise. By accurately detecting both clear speech and speech in noise, our system provides a much more comprehensive picture of verbal communication comparing to existing work such as CenceMe[93, 94], which can only detect clear speech.

Speech Volume Analysis

For clear speech, we perform volume analysis to determine who is speaking. When one person is speaking, all mobile phones on the subject and nearby persons record the speech. By comparing their volume, we can figure out who is speaking, because the volume from the person speaking is larger than that on all other persons. However, this may not be true when there are noises around, therefore sound classification mentioned in the previous section is necessary to make sure that volume analysis is only performed for clear speech.

Figure 6.3 illustrates the scenario when three people are talking with each other. When A speaks, phones of A, B, and C all record the voice signal. Since the volume recorded by A is larger than that recorded by B and C, we can detect that A is speaking.

GPS, Call Records, Calendar

The GPS data collected from mobile phones is very effective to detect some kinds of social interactions such as dining out, going to sport events or movies. Phone call records contain information such as the time, duration, caller (for incoming calls) or callee (for outgoing calls), and thus are used in our system to monitor the subject's interactions with others over their smartphones. A calendar entry consists of the time, location, and name of an event.

6.3.2 In-Person Interaction Detection

Each primitive discussed in the previous section provides basic components to analyze in-person interactions. However, one single primitive can hardly give enough details. For example, through GPS the subject can be located in an office building, but it is hard to know if the person is involved in any kind of in-person interactions such as talking with others.

Therefore, we adopt a multi-modal approach to detail in-person interactions: in our system, every in-person interaction is semantically defined using the vocabulary of primitives. Three in-person interactions are listed below to demonstrate how they are defined using a combination of primitives:

- Meeting: speech detection (clear speech) + proximity (multiple people are nearby) + activity (sitting posture) + volume analysis (how many times and how long the subject speaks).
- Dining out: GPS (in a restaurant) + proximity (there are people nearby) + speech detection (speech in noise)
- Exercise: activity (dynamic) + GPS (in a gym) + proximity (there are people nearby)

These examples show how multi-modal sensing helps extract details of in-person interactions. For example, instead of just detecting the "meeting" event, our system also evaluates the involvement of the subject into the meeting by using volume analysis to detect times and length the subject spoke. Similarly, by using the primitive of proximity, our system does not only detect "dining out" or "exercise", but also detects if the subject is doing these activities with others.

6.4 Evaluation

We perform two types of tests to evaluate our system: component tests and integration tests. Component tests are used to verify primitives, while integration tests are used to evaluate the effectiveness of the whole system to detect in-person interactions.

6.4.1 Component Test

Activity Detection

Our system detects activities in two levels of granularity: use the accelerometer on the smartphone to distinguish dynamic and static activities, or when TEMPO nodes are used, we detect specific postures including standing, sitting, and lying. To evaluate the performance of our activity detection

	Dynamic	Static
Dynamic	91.19%	8.81%
Static	2.53%	97.47%

Table 6.3: Confusion matrix for detecting static postures and dynamic movements using phone accelerometers.

	Recall	Precision
Lying	95.92%	100%
Standing	96.81%	98.89%
Sitting	96.71%	99.69%
Dynamic	100%	87.19%

Table 6.4: Results of detecting dynamic movements and specific static postures including standing, sitting, and lying using TEMPO nodes.

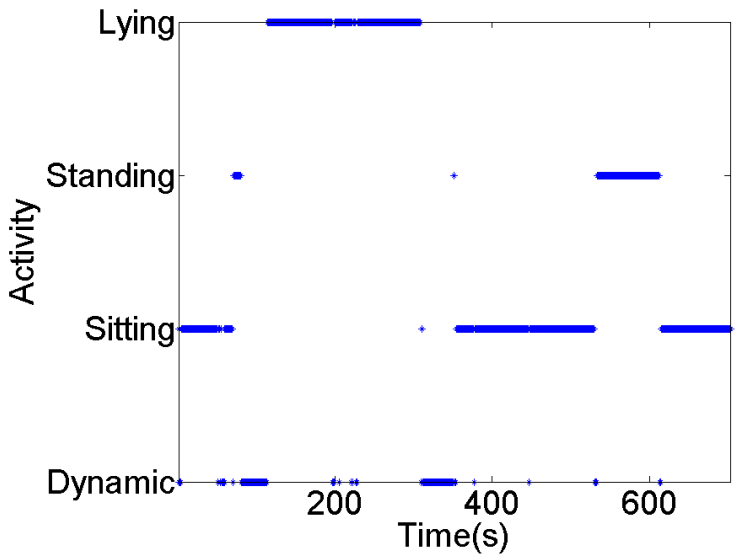


Figure 6.4: Activity detection results using TEMPO nodes.

	d (m)	$mean(d_m)$ (m)	$std(d_m)$ (m)
Indoor	1	1.06	0.14
	2	1.93	0.19
	3	2.51	0.29
Outdoor	1	1.05	0.22
	2	2.15	0.37
	3	2.73	0.55

Table 6.5: Distance estimated using RSSI: d is real distance, d_m is measured distance.

method, each of four subjects wear our system and performs various daily activities for 10 to 15 minutes. We use a camera to capture video of the subjects as the ground truth.

Table 6.3 shows the confusion matrix of using the accelerometer on a smartphone to distinguish static postures from dynamic movements. During these experiments, the smartphone was put into a trousers pocket, where people usually keep their phones, and the accuracy is higher than 90%. However, when people make calls or hold their phones in the hand, the phone accelerometer can no longer be used to detect activities.

Therefore, our system also utilizes on-body sensors. Using TEMPO nodes as described in Section 6.2, our system correctly detects more than 95% of various static postures and dynamic activities as shown in Table 6.4. Most of errors happen when subjects move their legs while standing, sitting, or lying. In this case, our system detects dynamic activities instead of static postures, and hence the precision for the dynamic case in Table 6.4 is relatively lower than others. Figure 6.4 shows the activity detection results using two TEMPO nodes as the subject sits, walks, lies down, walks, sits, stands, and then sits again. It also shows that the majority of misclassification is caused by regarding static postures as dynamic activities.

Proximity Information

Proximity information of other BSNs is retrieved through Bluetooth discovery service. Each smartphone tries to discover all nearby neighbors every 5 seconds. The process provides two levels of proximity information: 1) it detects the presence of a neighbor; 2) using the RSSI values collected, our system can approximate the distance between a neighbor and the subject.

To test the accuracy of estimating distance using RSSI, two subjects stand 1m, 2m, and 3m away from each other in both indoor (office and house) and outdoor environments. Table 6.5 shows the results of estimating distance according to Equation (6.3). Though the channel condition changes constantly, the average of measured distance d_m veritably corresponds to real distance d. The variance of d_m caused by channel condition changes makes it hard to estimate d accurately when two BSNs

Classifier	Accuracy (%)	Speech		Speech in Noise		Noise	
		Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)
Naive Bayes	88.08	92.05	95.13	73.08	67.71	89.78	88.73
Discriminant	88.98	93.13	91.73	73.96	72.27	91.14	94.93
Boosted Tree	87.27	87.55	94.85	78.84	58.18	90.75	92.64
Bagged Tree	92.37	93.23	95.74	85.85	78.08	94.62	95.62

Table 6.6: Classification resutls for the four classifiers. Simple classifiers such Naive Bayes and Discriminant achieve about 88% accuracy, which can be improved to more than 92% using the more complicated Bagged Tree classifier.

are only in the vicinity for a short period of time. However, since effective in-person interactions like meetings usually last more than one minute, our system is still able to extract proximity information reliably.

Speech Detection

To evaluate the our speech detection method, 6 subjects recorded 100 minutes of audio data to train the classifiers: 50 minutes of clear speech, 20 minutes of speech in noise (the ambient noise is from street and supermarket), and 30 minutes of ambient noise from a walking street and a supermarket.

Then seven of the eight features described in Section 6.3.1 are automatically selected to achieve the best performance for each classifier. The features selected are: $width$, $m1$, $m2$, $m3$, $CGAV$, $CGFS$, and $tonality$. The classifiers used include Naive Bayes, Discriminant, Boosted Tree, and Bagged Tree.

Table 6.6 shows the classification results. Among all classifiers, Discriminant is the easiest to implement and can achieve around 88% overall accuracy. Bagged Tree achieves best results in terms of sensitivity (recall), precision, and overall accuracy of 92.37%. Speech in noise is most difficult to correctly identify, because it is usually misclassified as clear speech when noise level is too low, or as noise when speech volume is too low.

Volume Analysis

How often and how long one speaks indicate the involvement during interactions with others. However, existing work fails to provide this information because it is difficult to identify the speaker only using the microphone on one single BSN. By sending volume recorded by each BSN to the back end server and detecting clear speech, our system is able to detect who is speaking.

Figure 6.5 shows the directional turn-takes of conversations between 3 subjects A, B, C, and the speech time of each subject. During the experiment, three subjects talk with each for 13 minutes. The conversation is recorded and manually labeled as ground truth. Our results show that A, B, and

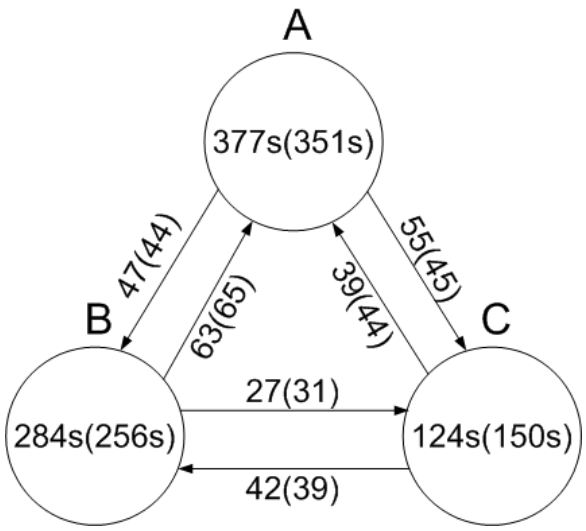


Figure 6.5: Directional turn-takes and length of speech during a meeting of 3 subjects A, B, and C. Numbers in parentheses are the ground truth obtained from manually labeled data.

C each speak for 377s, 284s, and 124s, respectively, which are very close to the ground truth of 351s, 256s, and 150s. Also there are 55 times (against the ground truth of 45 times) that C starts talking after A. These information provides details about the involvement during a conversations, and thus indicates the quality of the interaction.

Most errors are caused by two reasons: 1) there are rare times two or more subjects speak at the same time, and in this case, relative volume cannot correctly identify the speaker; 2) ambient noise can also change relative volume levels, and thus invalidates the results. Therefore, as described in Section 6.3.1, we only analyze the volume data of clear speech.

6.4.2 Integration Test

After evaluating each primitive, this section demonstrates how the multi-modal system provides more details of in-person interactions. Three typical interactions are studied: meeting, dining out, and exercising at a gym. More details about these activities can be detected by using different primitives:

- Meeting with colleagues: GPS (office building) + proximity (BSNs discovered) + activity (sitting) + speech detection (clear speech)
- Meeting with and speaking to colleagues: GPS (office building) + proximity (BSNs discovered) + activity (sitting) + speech detection (clear speech) + volume analysis (the subject is speaking)
- Dining out: GPS (restaurant)
- Dining out with friends: GPS (restaurant) + proximity (friends' BSNs discovered)

Activity	Recall (%)	Precision (%)
Meeting with colleagues	92.32	94.31
Meeting with and talking to colleagues	85.64	88.14
Dining out	100	100
Dining out with friends	100	100
Dinning out and talking with friends	82.91	75.41
Exercising in a gym	100	100
Exercising in a gym with friends	88.62	100

Table 6.7: Results of using multi-modal sensing to detect more details about interactions.

- Dining out and talking with friends: GPS (restaurant) + proximity (friends' BSNs discovered) + speech detection (speech in noise)
- Exercising in a gym: GPS (gym)
- Exercising in a gym with friends: GPS (gym) + proximity (friends' BSNs discovered)

Table 6.7 shows the experiment results of integration tests. We collect data on two people meeting and three people meeting for about half an hour. Four people dine out 3 times to collect dining data. Exercising data is collected when two people played table tennis. Though the accuracy of interaction detection becomes lower with requesting more details according to Table 6.7, our system still achieves accuracy of more than 80%.

6.5 Discussion

In this chapter, we proposed a multi-modal way to detect in-person interactions using smartphone and on-body sensors. Our system detects seven kinds of primitives including activities of subjects, proximity information, presence of speech, speech volume analysis, GPS coordinates, phone call records, and calendar events. This multi-modal way enables our system to extract more details about in-person interactions.

Unlike most existing work only relying on data collected from one person, our system collects primitives from all participants interacting with each other. In-person interactions intrinsically involve multiple participants, therefore, by comparing data collected from multiple persons, our method extracts interesting details such as who speaks more during an interaction, which is an important indicator of the quality of an interaction.

The component tests and integration tests demonstrate the practicality and accuracy of detecting individual primitives and complex interactions such as meeting, dining, and exercising using our system.

In the current system, in-person interactions are defined semantically by the combination of primitives. In the future, by combing the calendar events and other primitives, our system can be improved to automatically discover patterns of new kinds of in-person interactions in an unsupervised way.

Chapter 7

Unification of BSN QoS and Applications

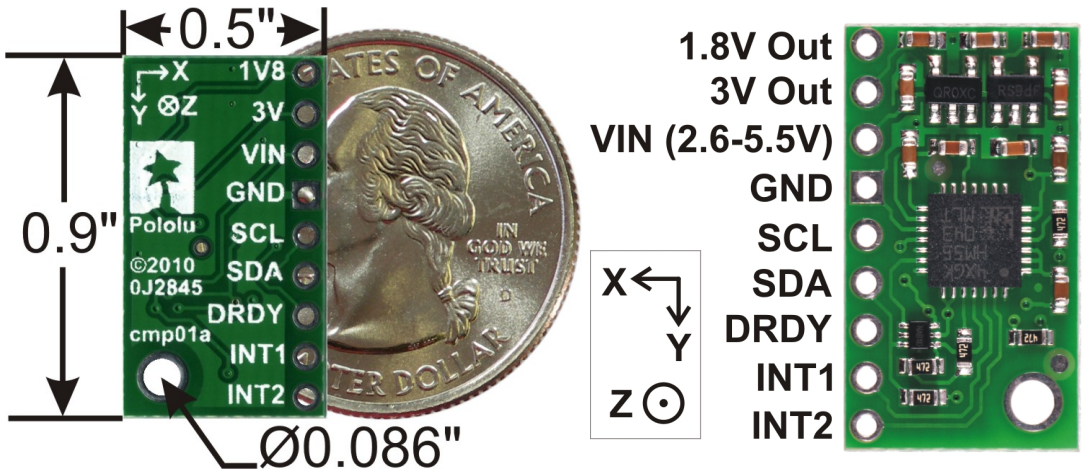
7.1 Introduction

So far we have proposed a QoS framework for multi-body, multi-function BSNs in Chapter 4, and implemented and evaluated the fall detection application in Chapter 5 and the in-person interaction monitoring application in Chapter 6. In this Chapter, the QoS framework and BSN applications discussed in previous chapters are unified into one complete BSN system.

Because TEMPO nodes use the Bluetooth radio, we need to use a Zigbee platform instead of TEMPO nodes to retrieve acceleration data to integrate the two applications into our BSN QoS framework. In our system, we choose the TelosB mote as the sensing platform, because it features an embedded antenna and is thus less intrusive than other motes such as MICAz.

Though TelosB motes default to have various sensors such as the humidity sensor and temperature sensor, there is no accelerometer on TelosB motes, so we connect the LSM303DLM 3D compass and accelerometer sensor board from Pololu Robotics & Electronics to TelosB motes via I^2C interface to collect acceleration data.

As smartphones have become a necessity and evermore powerful over the last ten years, it is natural to use them as aggregators because they can not only perform complex data processing, but they themselves are also a feature-rich sensing platform. In addition, as 4G cellular network provides cheaper and faster internet experience, smartphones can continue communicating with the base station even when there is no WiFi around. Though there are no smartphones with built-in Zigbee



(a) (b)

Figure 7.1: LSM303DLM 3D compass and accelerometer carrier with voltage regulators: (a) bottom view with dimensions; (b) labeled top view.

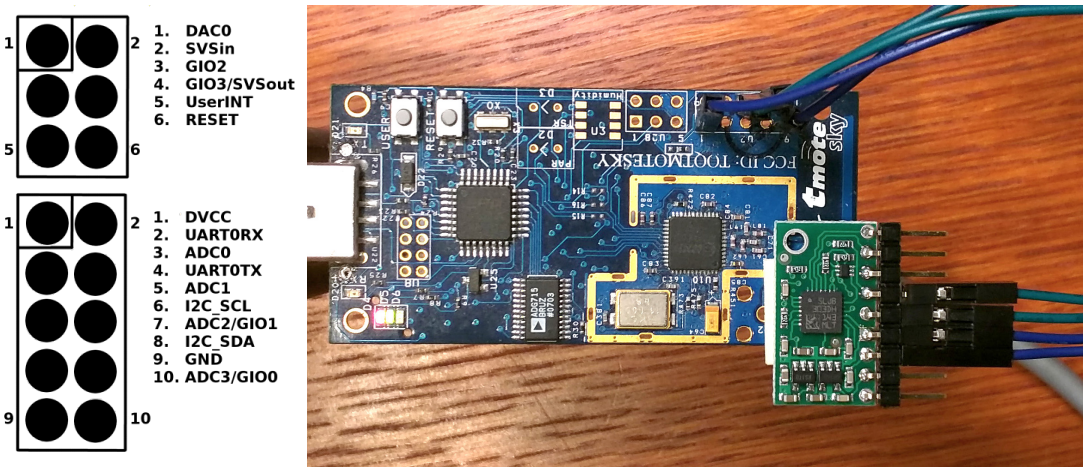
support, it is not difficult to connect a Zigbee motes to smartphones with a USB host interface. In our system, to simplify the development process, we use a laptop connected with two TelosB motes as the aggregator. The two TelosB motes operate on different channels and perform different functions: one is to communicate with other aggregators over the control channel 15, the other is to collect data from nodes of the same BSN.

Later in this chapter, we first introduce the details of the hardware platform, then profile both the BSN platform and the two applications we discussed in Chapter 5 and Chapter 6. After this we demonstrate the effectiveness of profiling and scheduling, power adaptation, and grouping in our system. At last, we evaluate the whole system against multiple QoS metrics including packet loss, bandwidth utilization, and application fidelity satisfaction.

7.2 BSN Hardware Platform

To implement a complete BSN system with the fall detection application and in-person interaction monitoring application running on top of our QoS solution, we use TelosB motes with LSM303DLM sensor board as the on-body sensor and a laptop connected with two TelosB motes as the aggregator.

The LSM303DLM is a system-in-package featuring a 3D digital linear acceleration sensor and a 3D digital magnetic sensor. The 3D accelerometer has a linear acceleration full-scale of $\pm 2/\pm 4/\pm 8g$ along each axis selectable by the user. The LSM303DLM includes an I^2C serial bus interface that supports standard mode (100KHz) and fast mode (400KHz). The system can be configured to generate an



(a) (b)

Figure 7.2: Connecting LSM303DLM sensor boards to TelosB motes: (a) extension pin of TelosB motes; (b) TelosB mote with LSM303DLM connected.

interrupt signal by inertial wakeup or free-fall events according to a programmed acceleration event along the enabled axes. Both wakeup and free-fall can be used simultaneously on two different accelerometer interrupts. Thresholds and timing of interrupt generators are programmable by the end user. The LSM303DLM features two data-ready signals which indicate when a new set of measured acceleration data and magnetic data are available, therefore simplifying data synchronization in the digital system that uses the device. Figure 7.1 shows the dimension and interface of LSM303DLM.

Figure 7.2(a) shows the extension connector of TelosB mote. By connecting VIN, GND, SCL, and SDA shown in Figure 7.1(b) with DVCC, GND, I2C_SCL, and I2C_SDA in Figure 7.2(a), we are able to use TelosB motes to collect acceleration data from LSM303DLM sensor board through I^2C interface. Figure 7.2(b) shows a TelosB node used in our system with the LSM303DLM sensor board connected.

To simplify the development process, we use laptops instead of smartphones connected with two TelosB motes as the aggregators. One TelosB mote is used to transmit/receive control messages including the RSSI measurement messages and the BSN transmission schedule updating messages. The other TelosB mote is used to collect acceleration data from other TelosB motes with LSM303DLM sensor boards connected: one is on the chest, the other is attached to the left thigh. The BSN hardware platform is as shown in Figure 7.3.

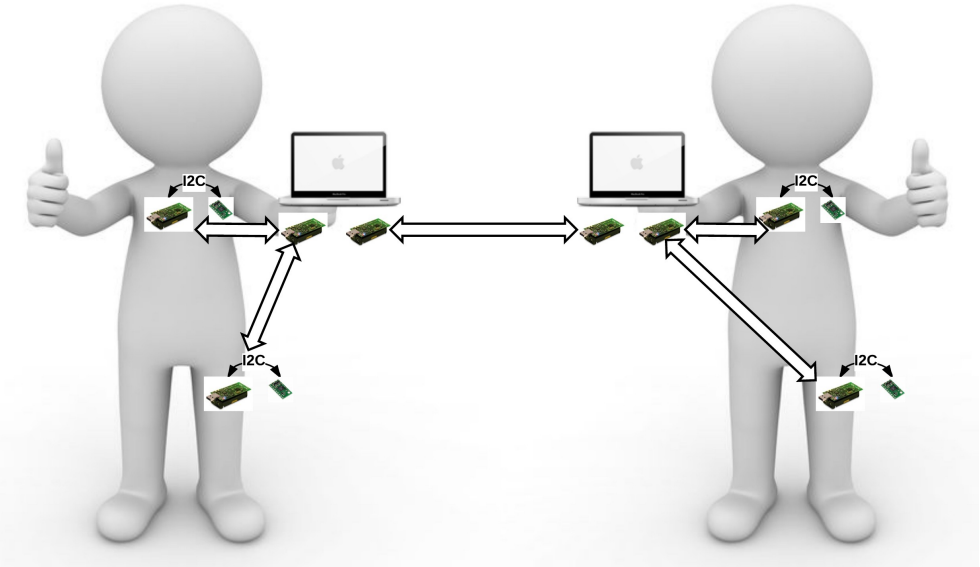


Figure 7.3: BSN hardware platform.

7.3 Profiling BSN and Applications

Based on the BSN hardware platform discussed in the previous section and the two BSN applications discussed in Chapter 5 and Chapter 6, we profile the BSN and applications for system integration and evaluation in later sections of this chapter.

7.3.1 BSN Profile

As shown in Figure 7.3, each BSN platform used in our system uses four TelosB motes: two are connected to the laptop for communication, the other two are attached to the chest and left thigh to collect acceleration data, respectively.

Listing 7.1 shows the profile of one BSN hardware platform used in your system. To keep the profile concise, we only include the accelerometer in the sensor list of each node. Sensors not used in our system such as the humidity sensor and the temperature sensor on TelosB motes are not included in the profile.

The LSM303DLM sensor board can be configured to work under low power mode or normal power mode. The output data rate under low power mode can be configured to 0.5, 1, 2, 5, or 10Hz. The output data rate under normal power mode can be 50, 100, 400, or 1000Hz. In our experiments, we use the normal power mode, and set the output data rate to 400Hz. The linear accelerometer on LSM303DLM can be set to three sensing range: $\pm 2g$, $\pm 4g$, or $\pm 8g$. Since LSM303DLM uses a 12-bit

ADC, the resolution (sensitivity) in each range can be calculated by dividing the range with the 2^{12}. The accuracy of the accelerometer is only provided for the sensing range of $\pm 2g$ in the datasheet.

Listing 7.1: Profile of the BSN hardware platform.


```
{
    "Id" : "24ab24c6-5c22-49ff-9c68-0fc88e3542e1",
    "Name" : "Profile of BSN with Two TelosB Connected with LSM303DLM",
    "Priority" : 1,
    "Nodes" : [
        {
            "Id" : "32748d31-a03c-4de3-9a76-9c2028bab15f",
            "Name" : "Chest Node",
            "Type" : "TelosB",
            "Location" : "Chest",
            "Sensors" : [
                {
                    "Id" : "d3cd99de-f351-4dc1-90ee-437789c280df",
                    "Name" : "Acceleration Sensor",
                    "Type" : "3D Accelerometer",
                    "Rate" : [0, 400],
                    "Modes" : [
                        {
                            "Id" : "9b7b3914-63ac-45a1-8413-785a5faf8ec4",
                            "Range" : [-8.0, 8.0],
                            "Resolution" : 0.0039,
                            "Unit" : "g",
                            "Bit" : 48
                        },
                        {
                            "Id" : "601b7ad8-5441-4b8d-98e9-b01dc0efd8f3",
                            "Range" : [-4.0, 4.0],
                            "Resolution" : 0.002,
                            "Unit" : "g",
                            "Bit" : 48
                        },
                        {
                            "Id" : "07edca74-6d4b-42aa-b5ce-2764483881fa",
                            "Range" : [-2.0, 2.0],
                            "Resolution" : 0.001,
                            "Accuracy" : 0.06,
                            "Unit" : "g",
```

```
                            "Bit" : 48
                        }
                    ]
                }
            ]
        },
        {
            "Id" : "616ec3eb-8d4c-46e2-8add-b66868de646a",
            "Name" : "Left Thigh Node",
            "Type" : "TelosB",
            "Location" : "Left Thigh",
            "Sensors" : [
                {
                    "Id" : "096ee1c0-6691-4f78-a277-d1c16d7a06f2",
                    "Name" : "Acceleration Sensor",
                    "Type" : "3D Accelerometer",
                    "Rate" : [0, 400],
                    "Modes" : [
                        {
                            "Id" : "95bcc67c-bbc3-43a9-b2e4-6430d65f8329",
                            "Range" : [-8.0, 8.0],
                            "Resolution" : 0.0039,
                            "Unit" : "g",
                            "Bit" : 48
                        },
                        {
                            "Id" : "be3fee0d-79f4-4759-a094-af6ea0fb5fac",
                            "Range" : [-4.0, 4.0],
                            "Resolution" : 0.002,
                            "Unit" : "g",
                            "Bit" : 48
                        },
                        {
                            "Id" : "0a6ebc62-46bf-482d-ad11-06399d734570",
                            "Range" : [-2.0, 2.0],
                            "Resolution" : 0.001,
                            "Accuracy" : 0.06,
                            "Unit" : "g",
                            "Bit" : 48
                        }
                    ]
```

```
                }
            ]
        }
    ]
}
```

7.3.2 Profile of the Fall Detection Application

Listing 7.2 shows the profile for fall detection used in our system. The profile contains two possible configurations. The first one is preferred with the preference level of 1. It requires 120Hz sample rate. The second has a lower preference level and requires 50Hz sample rate.

Listing 7.2: Profile of the fall detection application.

```
{
    "Id" : "dce07cfb-8b37-4529-95c8-4da9e558ccc4",
    "Name" : "Fall Detection",
    "Priority" : 1,
    "Configurations" : [
        {
            "Id" : "7206d248-a079-44cb-bafd-7768f9449c23",
            "Preference" : 1,
            "SensorRequirements" : [
                {
                    "Location" : "Chest",
                    "Type" : "3D Accelerometer",
                    "Rate" : 120,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                },
                {
                    "Location" : "Left Thigh",
                    "Type" : "3D Accelerometer",
                    "Rate" : 120,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                }
            ]
        },
```

```
        {
            "Id" : "9d3dfbef-1821-4eb6-9662-802e7c387eca",
            "Preference" : 2,
            "SensorRequirements" : [
                {
                    "Location" : "Chest",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                },
                {
                    "Location" : "Left Thigh",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                }
            ]
        }
    ]
}
```

7.3.3 Profile of the In-Person Interaction Monitoring Application

Listing 7.3 shows the profile for the in-person interaction monitoring application. It also requires two accelerometers as the fall detection application, but it only needs to sample acceleration at 50Hz. The priority is set to 2, which is lower than the fall detection, since social interactions are not so critical as fall events.

Listing 7.3: Profile of the in-person interaction monitoring application.

```
{
    "Id" : "fbde8aa6-a9e9-402d-946c-c2cd85ec515d",
    "Name" : "Social Detection",
    "Priority" : 2,
    "Configurations" : [
        {
            "Id" : "0a061cf6-7423-482c-ab0a-539fd33b03a6",
```

```
            "Preference" : 1,
            "SensorRequirements" : [
                {
                    "Location" : "Chest",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                },
                {
                    "Location" : "Left Thigh",
                    "Type" : "3D Accelerometer",
                    "Rate" : 50,
                    "Delay" : 1000,
                    "Range" : [-8.0, 8.0],
                    "Unit" : "g"
                }
            ]
        }
    ]
}
```

After building up the hardware platform to host both the fall detection and social activity monitoring in Section 7.2 and profiling the BSN platform and applications in Section 7.3, we can evaluate our QoS framework proposed in Chapter 4.

In the following sections we will discuss how our profiling and scheduling algorithms work in real systems in Section 7.4, demonstrate the correctness and effectiveness of power adaptation and grouping in Section 7.5 and Section 7.6, respectively, and perform integration tests to evaluate the performance of the whole system in Section 7.7.

7.4 Profiling and Scheduling Evaluation

There are three major pillars to support our QoS solution: profiling and scheduling, power adaptation, and grouping. In this section, based on profiles of the BSN hardware platform and two BSN applications defined in Section 7.3, we discuss how to apply our profiling and scheduling algorithms in real systems.

7.4.1 Compute BSN Settings

In our system, profiles are used as an innovative way to match BSN hardware specifications and software requirements. We have Algorithm 3 to validate application profiles against BSN profiles to produce BSN settings, and Algorithm 4 to combine two BSN settings so that it can satisfy the requirements of multiple applications. By applying these algorithms to profiles defined in Sectionn 7.3, we can get the combined BSN settings that can satisfy the requirements of both fall detection and in-person interaction monitoring. The combined settings are shown in Listing 7.4.

Listing 7.4: Combined BSN settings for fall detection and in-person interaction monitoring.


```
{
    "BSNId" : "24ab24c6-5c22-49ff-9c68-0fc88e3542e1",
    "Settings" : [
        {
            Id : "1390571d-f8db-4d19-a2bf-6a4397cabd37",
            "Applications" : [
                {
                    "ApplicationId" : "dce07cfb-8b37-4529-95c8-4da9e558ccc4",
                    "ConfigurationId" : "7206d248-a079-44cb-bafd-7768f9449c23"
                },
                {
                    "ApplicationId" : "fbde8aa6-a9e9-402d-946c-c2cd85ec515d",
                    "ConfigurationId" : "0a061cf6-7423-482c-ab0a-539fd33b03a6"
                }
            ],
            "Sensors" : [
                {
                    "NodeId" : "32748d31-a03c-4de3-9a76-9c2028bab15f",
                    "SensorId" : "d3cd99de-f351-4dc1-90ee-437789c280df",
                    "ModeId" : "9b7b3914-63ac-45a1-8413-785a5faf8ec4",
                    "Rate" : 120,
                    "Bit" : 48,
                    "Delay" : 1000
                },
                {
                    "NodeId" : "616ec3eb-8d4c-46e2-8add-b66868de646a",
                    "SensorId" : "096ee1c0-6691-4f78-a277-d1c16d7a06f2",
                    "ModeId" : "95bcc67c-bbc3-43a9-b2e4-6430d65f8329",
                    "Rate" : 120,
                    "Bit" : 48,
```

```
                    "Delay" : 1000
                }
            ]
        },
        {
            "Id" : "bc36defe-7c00-4e4e-851c-77dc7d00705c",
            "Applications" : [
                {
                    "ApplicationId" : "dce07cfb-8b37-4529-95c8-4da9e558ccc4",
                    "ConfigurationId" : "9d3dfbef-1821-4eb6-9662-802e7c387eca"
                },
                {
                    "ApplicationId" : "fbde8aa6-a9e9-402d-946c-c2cd85ec515d",
                    "ConfigurationId" : "0a061cf6-7423-482c-ab0a-539fd33b03a6"
                }
            ],
            "Sensors" : [
                {
                    "NodeId" : "32748d31-a03c-4de3-9a76-9c2028bab15f",
                    "SensorId" : "d3cd99de-f351-4dc1-90ee-437789c280df",
                    "ModeId" : "9b7b3914-63ac-45a1-8413-785a5faf8ec4",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 1000
                },
                {
                    "NodeId" : "616ec3eb-8d4c-46e2-8add-b66868de646a",
                    "SensorId" : "096ee1c0-6691-4f78-a277-d1c16d7a06f2",
                    "ModeId" : "95bcc67c-bbc3-43a9-b2e4-6430d65f8329",
                    "Rate" : 50,
                    "Bit" : 48,
                    "Delay" : 1000
                }
            ]
        }
    ]
}
```

7.4.2 Preference and Bandwidth of BSN Settings

In our QoS solution, when one channel cannot accommodate all BSNs assigned to it, the BSN with lowest priority is adjusted first to reduce bandwidth requirement. Algorithm 5 is used to calculate the preference of BSN settings. According to the algorithm, in Listing 7.4 the BSN setting with the Id of `1390571d-f8db-4d19-a2bf-6a4397cabd37` is more preferable than BSN setting `bc36defe-7c00-4e4e-851c-77dc7d00705c`, because the fall detection application with the Id of `dce07cfb-8b37-4529-95c8-4da9e558ccc4` has higher priority (lower `Priority` value) than the in-person interaction monitoring application, and the configuration with the Id of `7206d248-a079-44cb-bafd-7768f9449c23` has a lower `Preference` value of 1.

Algorithm 6 is used to determine the bandwidth requirement of each BSN setting. The bandwidth requirement of the BSN setting `1390571d-f8db-4d19-a2bf-6a4397cabd37` in Listing 7.4 is:

$$
\begin{aligned}
payloadBits &= 2 \times 120 \times 48 = 11520 \\
minPacket &= max(\left\lceil \frac{1000}{1000} \right\rceil, \left\lceil \frac{11520}{28 \times 8} \right\rceil) = 52 \\
Bandwidth &= 11520 + 52 \times 8 \times 18 = 19008bps
\end{aligned}
$$

Similarly, the bandwidth requirement of the BSN setting `bc36defe-7c00-4e4e-851c-77dc7d00705c` in Listing 7.4 is:

$$
\begin{aligned}
payloadBits &= 2 \times 50 \times 48 = 4800 \\
minPacket &= max(\left\lceil \frac{1000}{1000} \right\rceil, \left\lceil \frac{4800}{28 \times 8} \right\rceil) = 22 \\
Bandwidth &= 4800 + 22 \times 8 \times 18 = 7968bps
\end{aligned}
$$

7.4.3 Transmissions Scheduling

To schedule transmission, in Section 4.6.2 we first allocate BSNs in the same group to different channels based on their bandwidth requirement and the channel bandwidth. The bandwidth requirement of a BSN setting can be calculated using Algorithm 6 as demonstrated in the previous section. However, we still need to figure out the effective bandwidth of a channel.

IEEE 802.15.4 protocol claims 250kbps data rate, but this is just a theoretical limit and usually cannot be achieved in real systems. For example, in our system, according to Equation (4.2), there are T_{cca}, $T_{backoff}$, and T_{system} besides the air time T_{air} to transfer a packet.

Payload (byte)	Packet (byte)	T_{air} (ms)	T_{tran} (ms)	T_{air}/T_{trans}(%)
1	19	0.6080	2.0367	29.85
4	22	0.7040	2.2472	31.33
8	26	0.8320	2.5381	32.78
12	30	0.9600	2.8409	33.79
16	34	1.0880	3.1447	34.60
20	38	1.2160	3.4483	35.26
24	42	1.3440	3.7453	35.88
28	46	1.4720	4.0486	36.36

Table 7.1: T_{air} and T_{tran} of packets of different sizes for TelosB.

Since the interference between motes are avoided by grouping and time sharing in our system, we can disable CCA, which removes both T_{cca} and $T_{backoff}$. To figure out the real time needed to transfer a packet using TelosB motes, we use one TelosB mote to continuously send packets of different sizes to another receiver TelosB mote. The receiver calculates the number of packets received per second, which can be used to calculate the average time needed to transfer one packet. In the experiment, the two motes use Channel 26 to avoid WiFi interference, and acknowledgment and CCA are both disabled. Table 7.1 shows the values of the theoretical air time T_{air}, the measured transmission time T_{tran}, and the ratio T_{air}/T_{tran} under different packet sizes for TelosB motes.

The values of T_{air} and T_{tran} are plotted in Figure 7.4. From Figure 7.4 we see a major difference between T_{air} and T_{tran}, which is caused by the processing time of the TelosB platform. The figure also shows the linear change of T_{tran} with the packet size, which shows the system overhead increases with payload size linearly. Figure 7.5 shows the linear fit between the packet size and T_{tran}. Therefore, we can calculate the transmission time of a packet in milliseconds with n bytes of payload on a TelosB platform:

$$T_{tran} = 0.075 \times (n + 18) + 0.6 \tag{7.1}$$

Figure 7.6 shows the ratio between T_{air} and T_{tran} for TelosB motes. As expected, with the payload size increasing, T_{air}/T_{tran} becomes larger. So to improve bandwidth utilization, larger payload size is preferred. In our system, we always try to send 28 bytes of payload when possible, then the time to transfer a packet becomes:

$$T_{tran} = 4.05ms \tag{7.2}$$

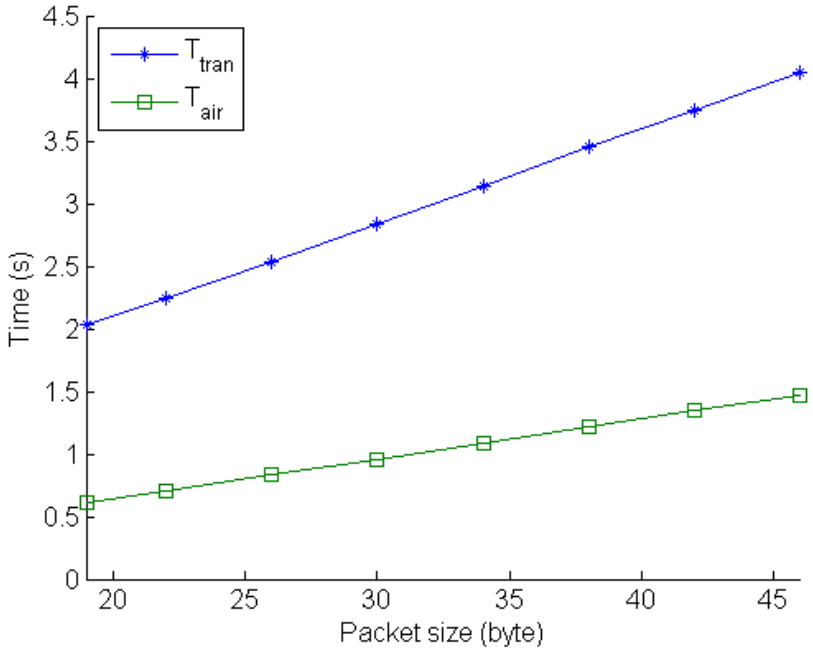


Figure 7.4: The values of T_{tran} and T_{air} of packets of different size for TelosB.

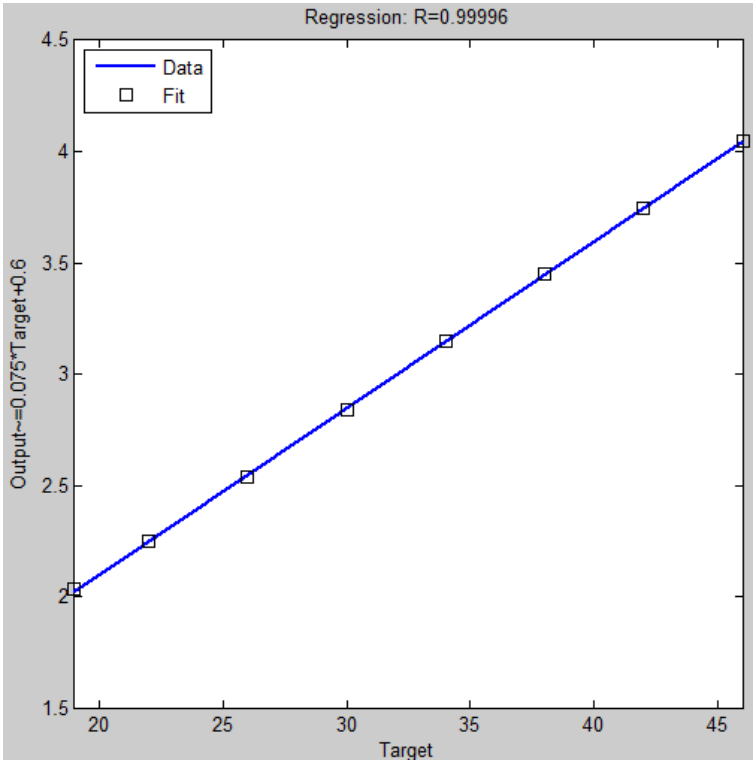


Figure 7.5: Linear fit between the packet size and T_{tran} for TelosB.

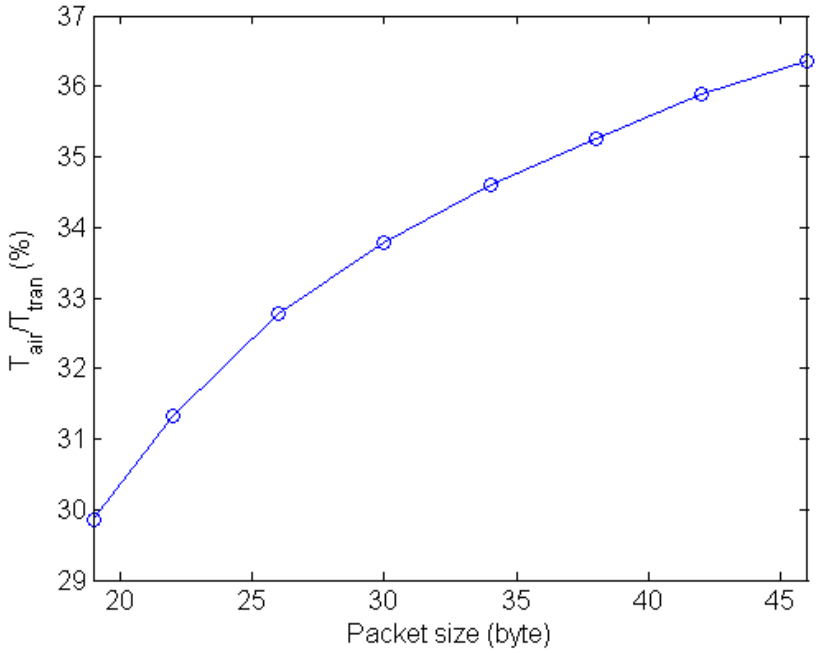


Figure 7.6: The values of T_{air}/T_{tran} of packets of different size for TelosB.

Based on this, we can also calculate the effective bandwidth per channel for TelosB motes when payload size is 28 bytes:

$$EffectiveBandwidth = (1000/4.05) \times 46 \times 8 = 90864.20bps \tag{7.3}$$

which is much smaller than the theoretical 250 kbps. Therefore, after calculating the bandwidth requirement of a BSN setting as in the previous section, when allocating BSNs in the same group to different channels in our system, we should use Equation (7.3) as the channel bandwidth, not 250 kbps.

Based on Equation (4.1) and Equation (7.1), we can calculate the transmission time needed for a TelosB mote during time d:

$$c = n \times T_{tran} = \left\lceil \frac{payloadBits \times minDelay}{1000 \times maxPayloadSize} \right\rceil \times (0.075 \times (maxPayloadSize + 18) + 0.6) \tag{7.4}$$

Based on the effective bandwidth shown in Equation (7.3) and c shown in Equation (7.4), we can perform both channel selection and delay-aware real-time scheduling proposed in Section 4.6.2 and Section 4.6.3.

7.5 Power Adaptation Evaluation

Power adaptation is a very important part in our system, it serves three objectives: to overcome body shadowing, one major obstacle of guaranteeing PRR, adequate transmission power is needed; then to make grouping more effective, we need to reduce the interference range of BSNs as much as possible, so we need to use as less power for transmission as possible; thirdly, to prolong the lifetime of BSNs, we need to lower transmission power and reduce energy consumption. These objectives are obviously incompatible with each other. The reliable communication is of the most importance in our system, because if data can not be transferred reliably (PRR) and in time (delay), the fidelity of the applications cannot be guaranteed and all other dimensions of metrics do not matter any more. However, we obviously cannot just use the maximum transmission power. The best compromise is to use the "just enough" power to transfer data. And the power adaptation is the way to find the "just enough" power.

In this section, we discuss how we select the key parameters RT_L and RT_H to guarantee the correctness of our power adaptation algorithm, and evaluate the effectiveness of power adaptation.

7.5.1 Correctness, Selection of RT_L and RT_H

RT_L and RT_H are the key parameters to guarantee the correctness of the power adaptation algorithm, because they determine the link quality for node-to-aggregator traffic. If RT_H is too large, the system ends up wasting energy since the link quality is better than needed, and this will increase the interference range of the BSN and make the grouping algorithm less efficient. If RH_L is too low, the link quality and PRR cannot be guaranteed, which will fail the main goal of our QoS framework.

It's worth noting that these thresholds vary across platforms, because different mote platforms may use different kinds of radio chips which have different sensitivities. For example, our system is implemented using TelosB motes which use CC2420 operating at 2.4GHz, while in [123] Mica2Dot motes which use CC1000 operating at 900MHz are used, therefore the RSSI thresholds proposed in in [123], $RT_L = -88dBm$ and $RT_H = -82dBm$, cannot be used directly in our system.

To figure out the relation between RSSI and PRR, we attach 5 TelosB motes to the chest, wrist, thigh, ankle, and back as shown in Figure 3.2. Each mote sends packets to the aggregator located at the waist at the rate of 50 packets per second for two seconds (100 packets sent per mote per turn) one after another. No two motes send packets at the same time. Each turn transmission power circulates in the list of 3, 7, 11, 15, 19, and 23, which correspond to -25dBm, -15dBm, -10dBm, -7dBm, and -5dBm according to Table 3.1. Packet acknowledgment and CCA are disabled. Channel

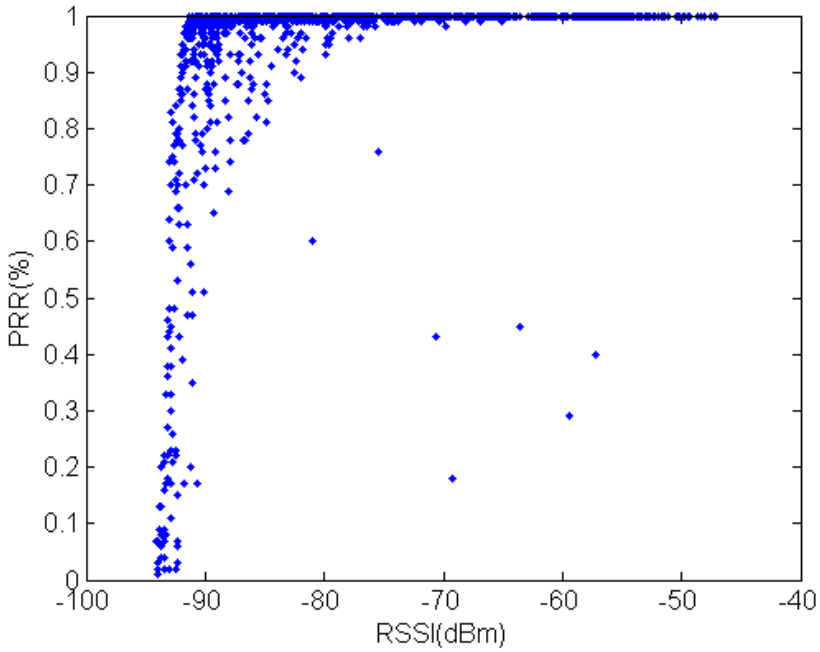


Figure 7.7: The relation between RSSI and PRR.

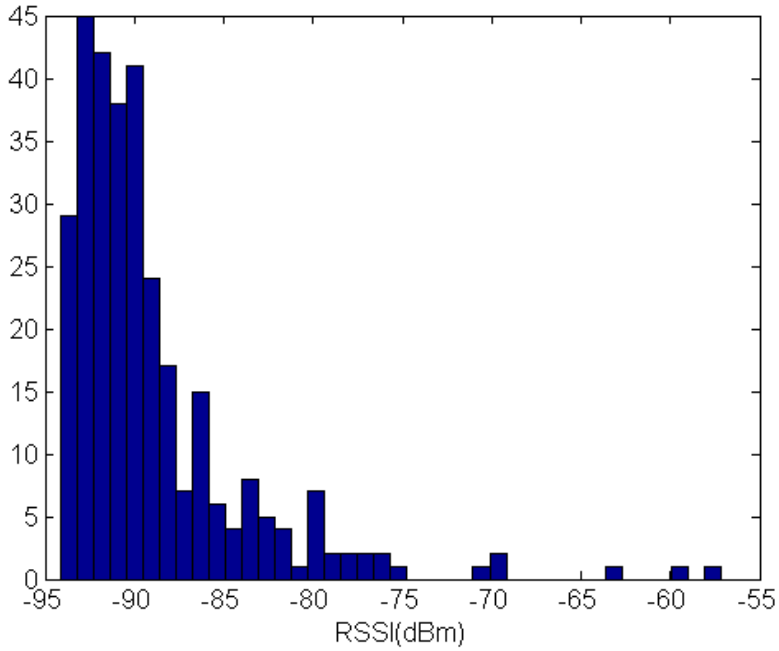


Figure 7.8: The distribution of RSSI when PRR is lower then 99%.

26 is used to avoid WiFi interference. PRR is calculated every 2 seconds according to the number of packets received from each mote. During the experiment, the subject performs normal daily activities such as sitting, standing, and walking for about an hour.

Figure 7.7 shows the relation between RSSI and PRR. Excluding a couple of outliers, we can see that when RSSI is large enough, the PRR is always around 100%. Figure 7.8 shows the histogram of RSSI values when PRR is lower than 99%. Ruling out the outliers that can be easily correlated to Figure 7.7, we can see when RSSI is between $-75dBm$ and $-65dBm$, PRR never drops below 99%. Therefore, we choose $RT_L = -75dBm$ and $RT_H = -65dBm$.

7.5.2 Effectiveness

With RT_L and RT_H selected, we can evaluate the effectiveness of power adaptation. In our experiment, TelosB motes are attached to the chest, thigh, wrist, and ankle of the subject, while the aggregator is located at the waist. Nodes start to send packets using power level 11. Within every second, each node sends one packet to the aggregator one after another. No two nodes transfer data at the same time to avoid interference. As soon as the aggregator receives a packet, it sends a power adjustment message to the sender if the RSSI value of the received packet is out of the range of RT_L and RT_H. During the whole process, the subject freely performs normal daily activities such as sitting, standing, walking, running, etc.

Figure 7.9 shows how the transmission power level is adjusted along time. Among nodes of the four locations, the chest node keeps using the lowest power level, lower than 5 most of the time, since it has the best line of sight with the aggregator. Similarly, for the node attached to the thigh, using power level less than 10 is enough to achieve the target RSSI at the aggregator. However, the ankle node needs to use power level as high as 25 to maintain satisfying RSSI. The wrist node uses lower power than the ankle node, because it is closer the aggregator. But the power level of the wrist node varies much more than the chest node and the thigh node due to its frequent movement. Figure 7.10 shows the RSSI values of these nodes at the aggregator during the same period. RSSI values of the ankle node are most sensitive to posture changes and vary most.

Figure 7.9 and Figure 7.10 demonstrate the effectiveness of the power adaptation algorithm. Figure 7.9 shows that to maintain the quality of a link in terms of RSSI, the power needed over time can be of great difference. Without power adaption, it would be very difficult to balance between saving energy when the link quality costs less power to maintain and maintaining good link quality by increasing transmission power when needed. Figure 7.10 shows the variance of RSSI is centered

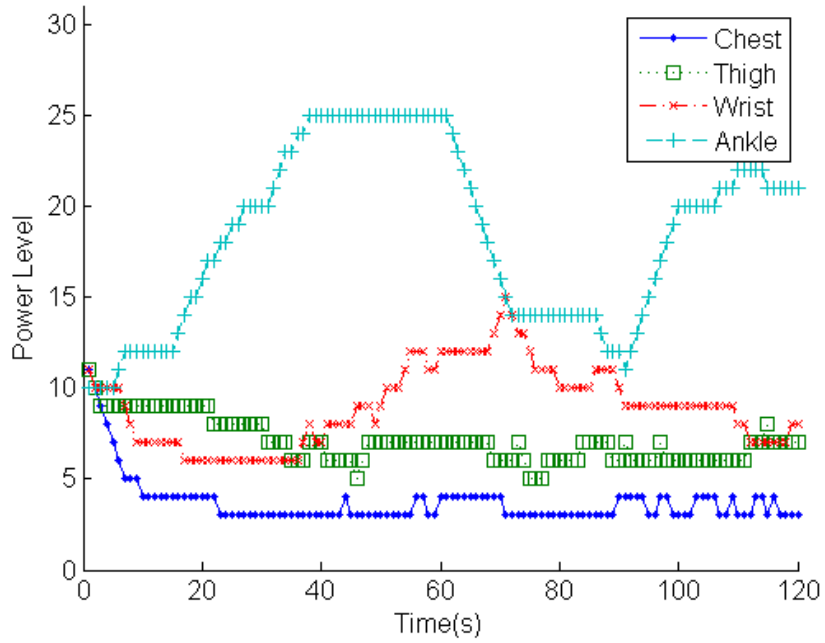


Figure 7.9: Transmission power level changes during normal daily activities with power adaptation. Sensor nodes are attached to the chest, thigh, wrist, and ankle.

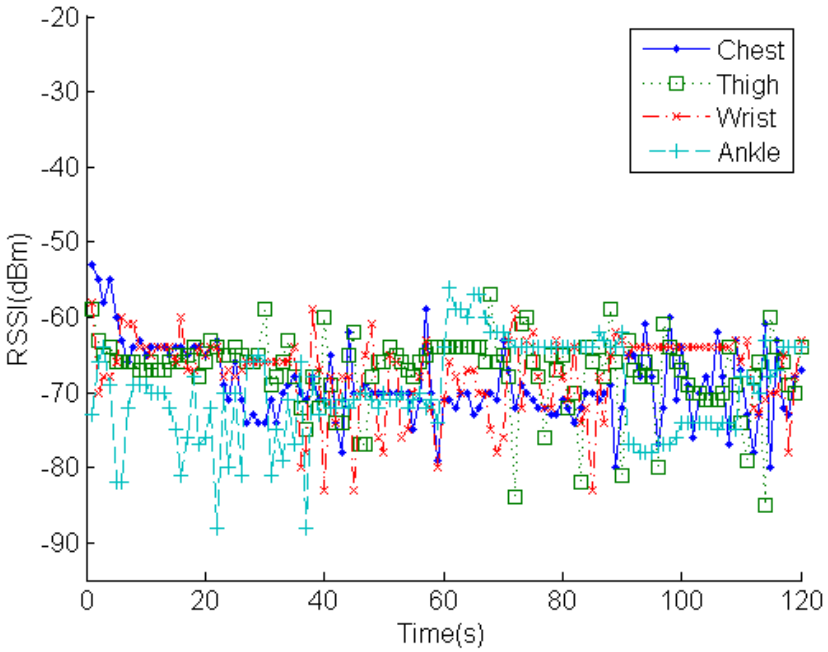


Figure 7.10: RSSI changes during normal daily activities with power adaptation. Sensor nodes are attached to the chest, thigh, wrist, and ankle.

around the target range between $-75dBm$ and $-65dBm$ we choose in the previous section, which means the link quality and PRR can be guaranteed with power adaptation.

7.6 Grouping Evaluation

Grouping is the first algorithm proposed in Chapter 4, and it is also the foundation of the whole QoS solution. We coordinate the transmission of nodes within the same group of BSNs based on the assumption that there are no or very little interference between each group of BSNs. To support this assumption, the performance of grouping in terms of responsiveness and correctness needs to be evaluated. We also show the effectiveness of the grouping algorithm in common BSN locomotion scenarios after discussing various parameters that affect the performance of the grouping algorithm.

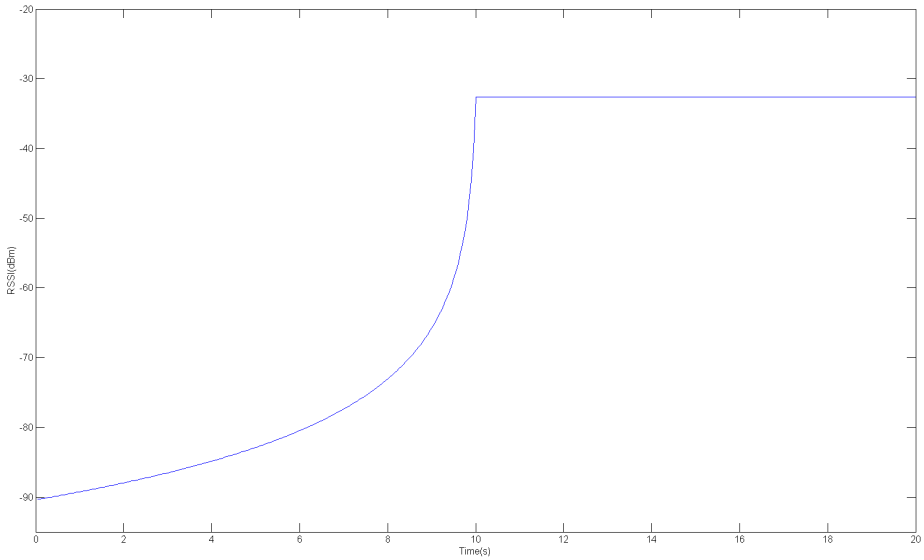
7.6.1 Responsiveness, Selection of w and m

The responsiveness of grouping is about how fast we can detect the locomotion of BSNs and change our grouping results accordingly. The responsiveness of grouping is crucial in the following scenarios:

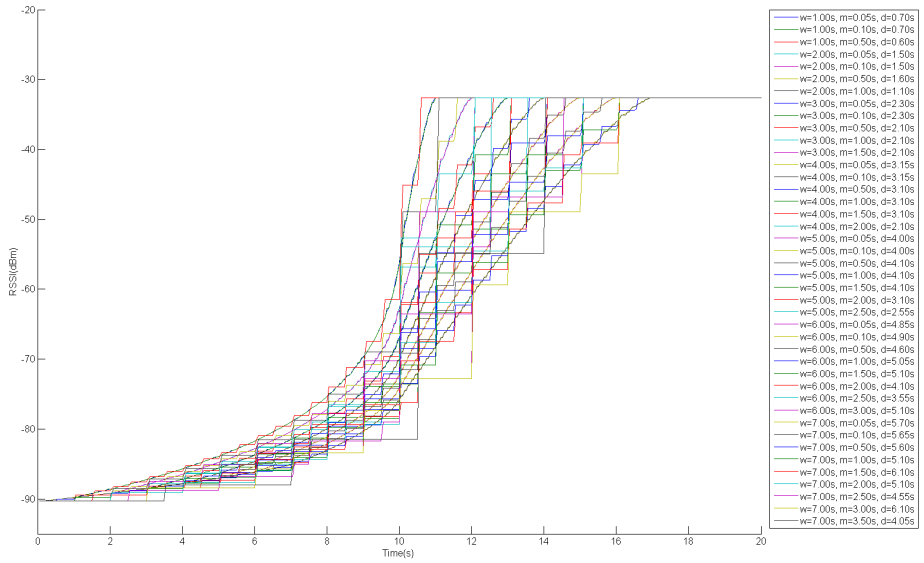
- when one BSN comes to the vicinity of other BSNs or when BSNs come together, the faster the grouping speed, the less chance of transmission collisions, and thus better BSN link qualities;
- when one BSN leaves a group of BSNs or when BSNs scatter away from each other, the faster the grouping speed, the faster we can start to reschedule transmission to improve bandwidth utilization as well as the fidelity of applications, since they can switch to higher fidelity configurations.

The responsiveness of grouping in Algorithm 1 is controlled by the moving average window size w and the beacon broadcast interval m. To evaluate various combinations of w and m, we performed both simulation and real measurements of the two scenarios mentioned above. The reason to perform both simulation and real measurements is that RSSI values are sensitive to the environment and noisy in nature, so simulation can help reveal patterns buried by noisy measurements, while real measurements are to make sure the simulation is not carried too far away from the reality.

To study BSNs getting towards each other, we consider one BSN approaches another BSN which is 14 meters away in the normal walking speed of about $1.4m/s$, so it takes about 10 seconds for the two BSNs to meet. After they meet, the two BSNs stay together for another 10 seconds for the moving average of RSSI values to converge.

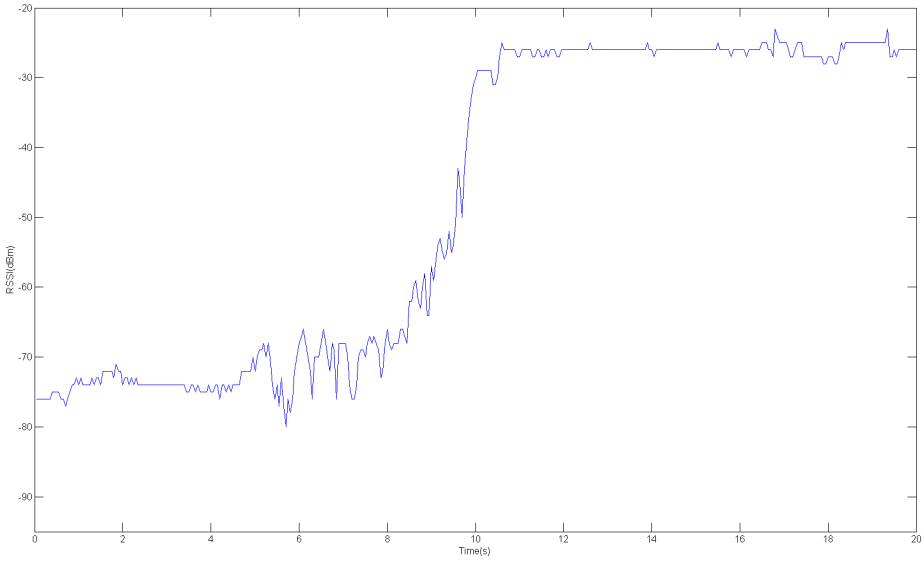


(a)

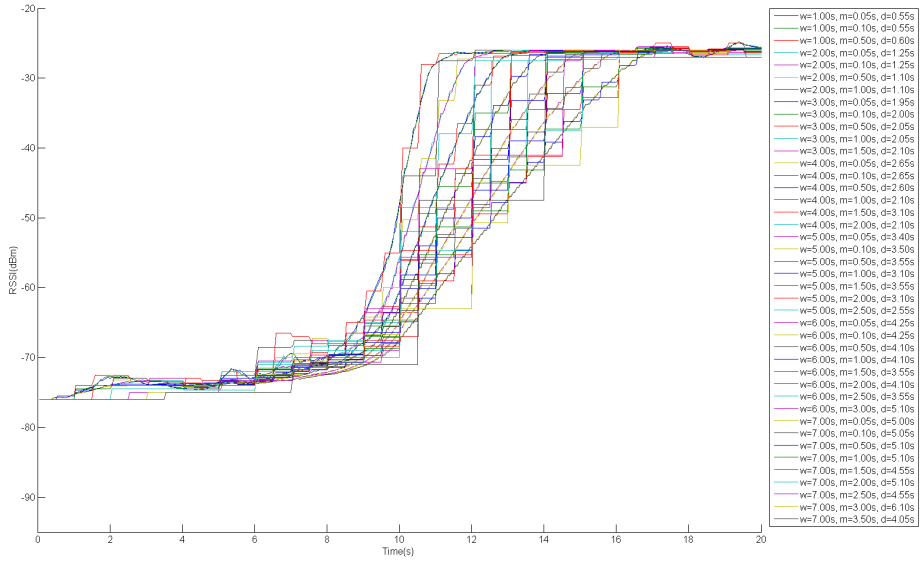


(b)

Figure 7.11: Simulation results when two BSNs approach each other: (a) RSSI values; (b) Moving average changes with different w and m.

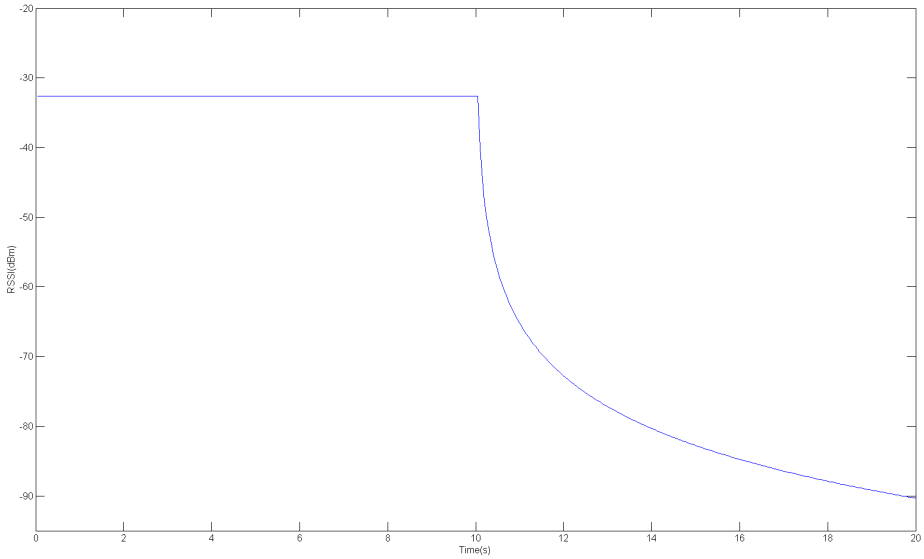


(a)

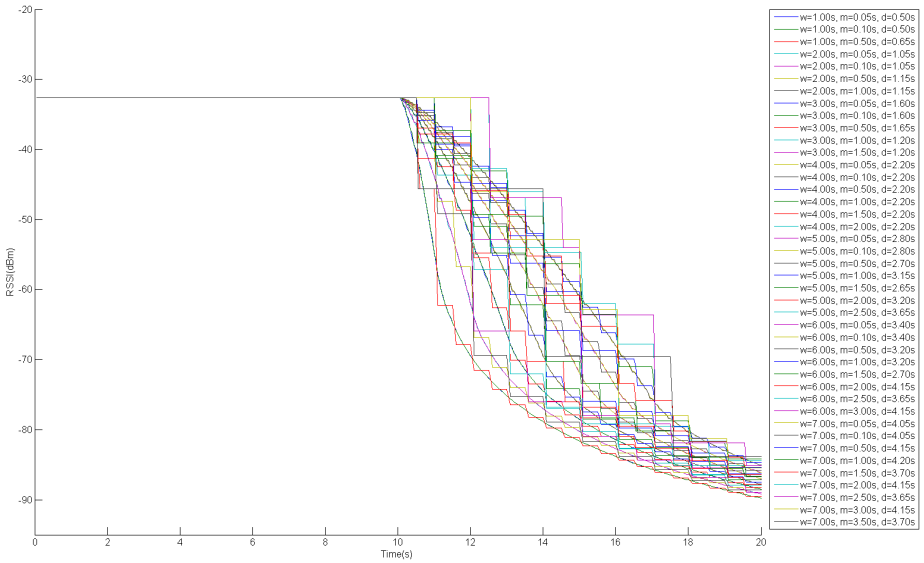


(b)

Figure 7.12: Real measurements when two BSNs approach each other: (a) RSSI values; (b) Moving average changes with different w and m.

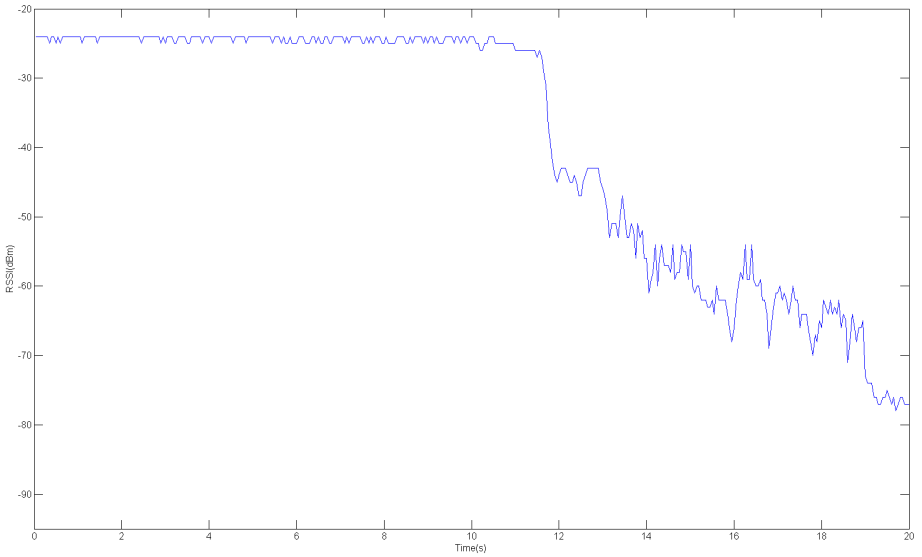


(a)

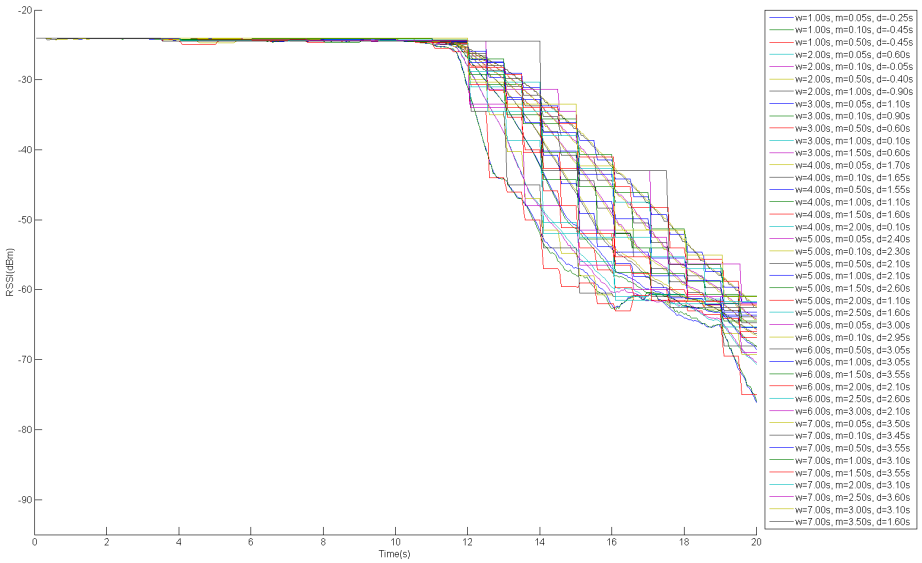


(b)

Figure 7.13: Simulation results when two BSNs leave from each other: (a) RSSI values; (b) Moving average changes with different w and m.



(a)



(b)

Figure 7.14: Real measurements when two BSNs leave from each other: (a) RSSI values; (b) Moving average changes with different w and m.

Figure 7.11 shows the simulation results of this process. Figure 7.11(a) shows the RSSI values which are calculated based on the distance between the two BSNs using Equation (6.2). Figure 7.11(b) shows how the different values of the moving average window size w and the beacon broadcast interval m affect the time needed for the moving average of RSSI values to converge to a stable value. The value d in the legends of Figure 7.11(b) is the additional time used to converge RSSI moving averages after two BSNs meet. The value of d changes with w and m.

In general, we observe d increases with w, which makes sense since the larger the moving window, the slower the changing of the moving average. However, it is interesting that with different values of beacon interval m, the delay d can be smaller for a larger moving window w. For example, when $w = 7.00s$, $m = 2.50s$, $d = 4.55s$, but when $w = 6.00s$, $m = 1.50s$, $d = 5.10s$.

Figure 7.12 shows the real measurements of the approaching process described above. Figure 7.12(a) shows the RSSI values, while Figure 7.12(b) shows the relation between w, m, and d. The real measurements shown in Figure 7.12 also shows the same pattern. For example, when $w = 5.00s$, $m = 2.50s$, $d = 2.55s$, which is smaller then some values of d when $w = 4$.

Figure 7.13 and Figure 7.14 show the process when two BSNs are leaving from each other, which is the opposite of the approaching process. In the experiment, two BSNs stay together for about 10 seconds, and then one BSN walks away in the normal walking speed of about $1.4m/s$, and after about 10 seconds, the two BSNs are 14 meters apart. Figure 7.13 shows the simulation results of this process, while Figure 7.14 show the real measurements. Similar to the approaching process, the value d is the time taken for RSSI moving averages to converge after two BSNs are apart. We also observe the pattern that d can be smaller for larger w with various m.

Considering the normal walking speed of $1.4m/s$, 4 seconds are the time needed to walk about 5.6 meters, and according to our study in Chapter 3, 6 meters are the distance when BSNs operating at the normal power level of 11 don't interference with each other. So to guarantee the responsiveness of the grouping algorithm, we want to make sure RSSI moving averages can converge in around 3 seconds (less then 4 seconds to leave some safe space) after RSSI values become stable. Based on our simulation and experiments, the moving window size w is the most important factor to affect the convergence time, so w cannot be too large. However, to avoid the effect of radio uncertainty, we need a relatively larger w/m to make the "average" meaningful. Meanwhile, to save our system from flood of grouping messages, m should not be too small. Therefore, in our system, we choose $w = 4.00s$ and $m = 1.00s$. The simulation and measurements in this section also provide a guideline if a more responsive or more stable grouping performance is ever needed.

7.6.2 Correctness, Selection of RSSI Threshold and Distance Metrics

The correctness of the grouping is about whether our algorithm can actually guarantee there is no or very little interference between BSN groups. This is related to the RSSI threshold we use to cluster BSNs. If the RSSI threshold (which is actually for $RSSI * (-1)$, since RSSI values are negative) is higher, more BSNs are considered to interfere with each other and thus are grouped together. This means less chance of interference, but the bandwidth utilization becomes lower at the same time, since nodes within one group cannot transmit packets at the same time. If the RSSI threshold is lower, BSNs will be more prone to be assigned to different groups, and thus can transmit packets all together. This can result in packet collisions if the interference is not in fact negligible between BSNs of different groups.

In Section 3.4, we studied the interference pattern between two BSNs. When nodes of the two BSNs both use a static transmission power level of 11, which have been shown earlier in Section 3.2 as the power level enough to overcome body shadowing effects most of the time, inter-BSN interference is negligible when these two BSNs are at least six meters away. In this scenario (transmission power level equals to 11, two BSNs are 6 meters away), the mean value of RSSI from interference packets at the aggregator of the subject BSN is $-70.49dBm$, the standard deviation is 1.78. Therefore, we can use 70.49 as the distance threshold when performing hierarchical clustering when all BSN nodes are using the static transmission power of 11.

However, the transmission power level of 11 is not always necessary to achieve high PRR. For example, the chest node can maintain PRR above 99% using lower power levels when communicating with the aggregator on the waist. On the other hand, for nodes far away from or out of the sight of the aggregator, power level 11 can be insufficient for reliable communication. So we proposed dynamic power adaptation in Section 4.7. This presents new challenges to our grouping algorithm: BSNs with nodes using larger power levels have much larger interference range, while nodes using lower power levels can be more subjective to interference at the receiver due to the change of relative signal strength.

Fortunately, power adaptation presents both challenges and the key to the solution. According to Section 7.5.1, the power adaption mechanism keeps RSSI values at the aggregators between $RT_L = -75dBm$ and $RT_H = -65dBm$. This means we don't need to consider the transmission power used by each node, because the signal strength at the aggregator always falls around the range of RT_L and RT_H. Therefore, we only need to find the interference signal strength that impacts the receptions of signals within RT_L and RT_H. This RSSI value will be our clustering threshold to group

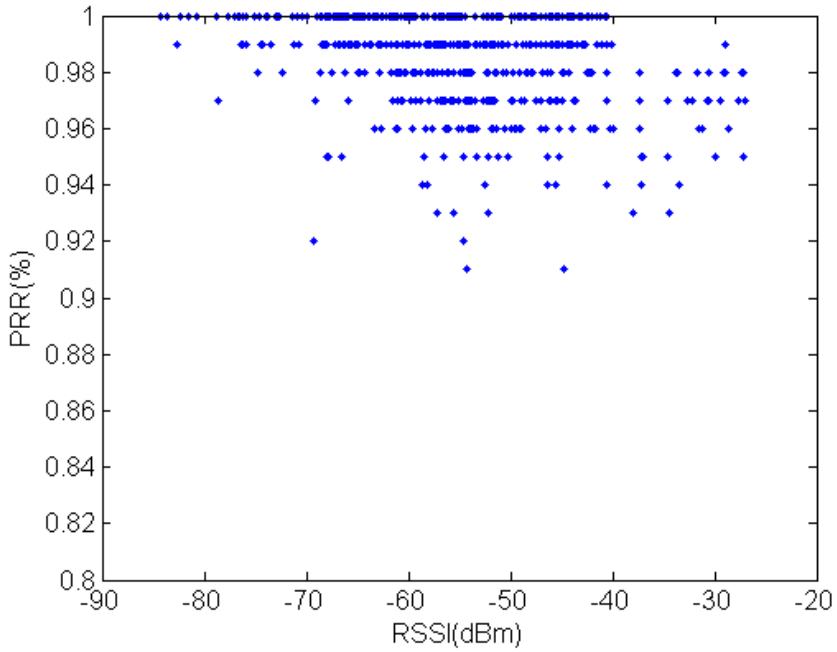


Figure 7.15: The relation between interference RSSI and the PRR of the subject BSN.

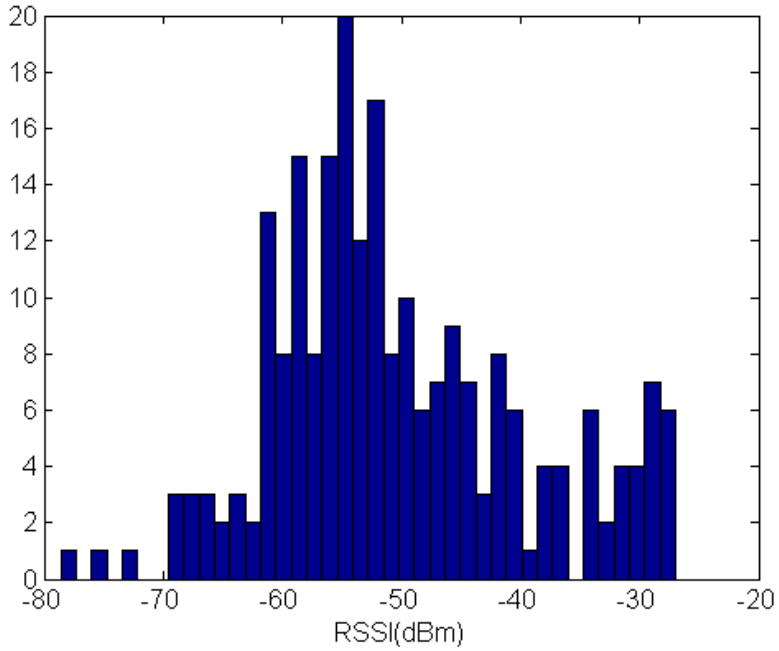


Figure 7.16: The distribution of RSSI when PRR of the subject BSN is lower than 99%.

nearby BSNs.

To figure out the threshold, we attach 5 TelosB motes to the chest, thigh, wrist, ankle and back to the subject BSN, and let these motes send packets to the aggregator attached to the waist at the rate of 50 packets per second for two seconds (100 packets sent per mote per turn) one after another. Power adaptation is enabled to make sure the RSSI at the aggregator is within RT_L and RT_H. No two motes within the subject BSN send packets at the same to avoid intra-BSN interference. Packet acknowledgment and CCA are disabled. Channel 26 is used to avoid WiFi interference. Then we set up the interference BSN in the exact same way except that in the interference BSN instead of using power adaptation each turn the power level used to send packets circulates among 3, 7, 11, 15, 19, 23, 27, and 31. Every two seconds we calculate the PRR of the subject BSN and the average of interference RSSI values at the aggregator of the subject BSN.

The relation between the interference RSSI values and the PRR of the subject BSN is shown in Figure 7.15. There are most dots with lower PRR values when the interference RSSI is around $-60dBm$ and $-50dBm$. Actually, PRR of the subject BSN is affected more when the interference RSSI is larger, but it is not common for the interference RSSI to reach around $-30dBm$, so there are fewer dots in that range.

Figure 7.16 shows the distribution of RSSI values when the PRR of the subject BSN is lower than 99%. Excluding some outliers, we can see that when the interference RSSI is smaller than $-70dBm$, it doesn't affect the PRR of the subject BSN much any more. So we choose $-70dBm$ as the cluster threshold for our grouping algorithm.

Besides the RSSI threshold, choosing different distance metrics to calculate the distance between clusters in the hierarchical clustering process used in Algorithm 2 can produce different grouping results. For example, when we arrange 5 BSNs in line and each one is 5 meters away from its neighbors as shown in Figure 7.17(a). Using the same $-70dBm$ as the clustering threshold but different distance metrics, we can get following grouping results:

- $\{[1,2],[3,4,5]\}$ when using the average distance between elements as the distance between clusters ("average linkage") as shown in Figure 7.17(b).
- $\{[1,2,3,4,5]\}$ when using the nearest distance between elements as the distance between clusters ("single linkage") as shown in Figure 7.17(c).
- $\{[1,2],[3,4],[5])\}$ when using the average distance between elements as the distance between clusters ("complete linkage") as shown in Figure 7.17(d).

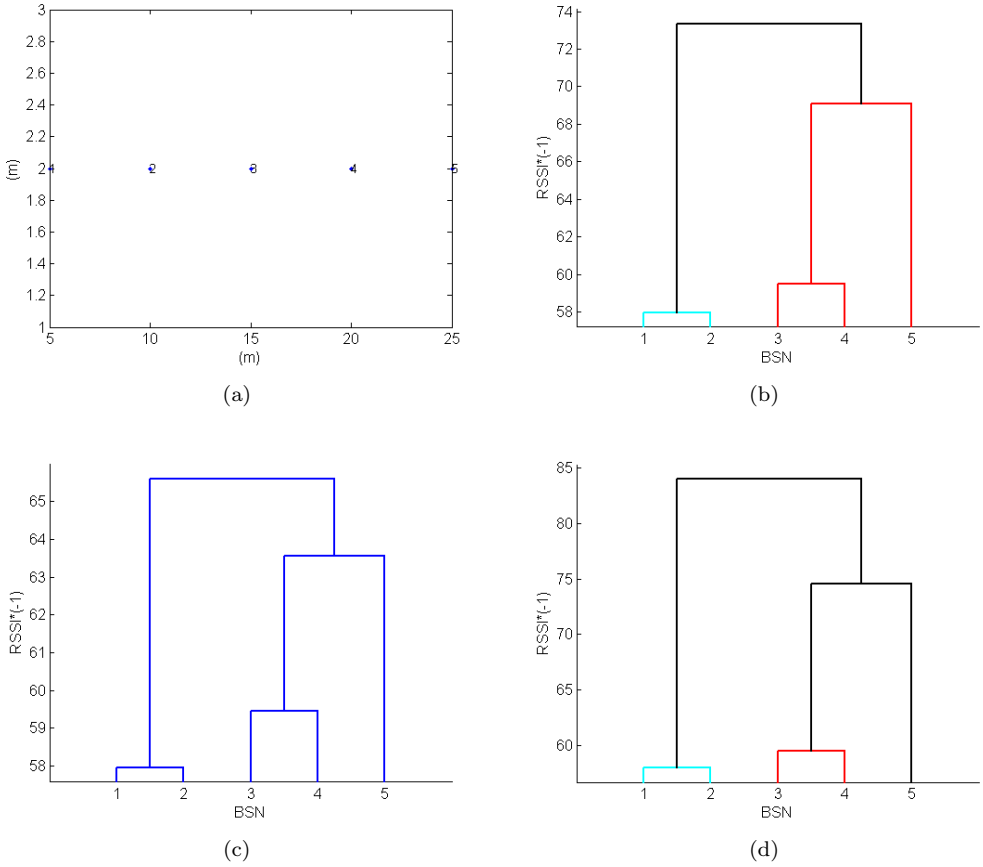


Figure 7.17: Grouping results using different distance metrics to calculate the distance between clusters: (a) 5 BSNs are aligned in a line, each is 5 meters away from its neighbors; (b) using average linkage criteria; (c) using single linkage criteria; (d) using complete linkage criteria.

Obviously, every BSN interferes with its neighbors according to our RSSI threshold selected earlier this section, so it seems we should use the single linkage so that all 5 BSNs can be grouped together and transfer data at different time. However, every BSN actually *only* interfere with its neighbors. If we use the single linkage criteria, 5 BSNs share one Zigbee channel, which dramatically lowers the bandwidth utilization. On the other side, if we choose to use the complete linkage, 5 BSNs are allocated to 3 groups, which can increase the bandwidth utilization but also the change of packet collisions. Therefore, we choose to use the average linkage criteria which is a compromise between the two extremes.

In Figure 7.17(b), BSN 2 and 3 are in different groups, but they are close enough to interfere with each other if they transfer data at the same time. We call this scenario "boarder interference" problem caused by using the average linkage criteria. A second level adjustment of the transmission schedule can be used to solve the problem of boarder interference: after calculating the transmission

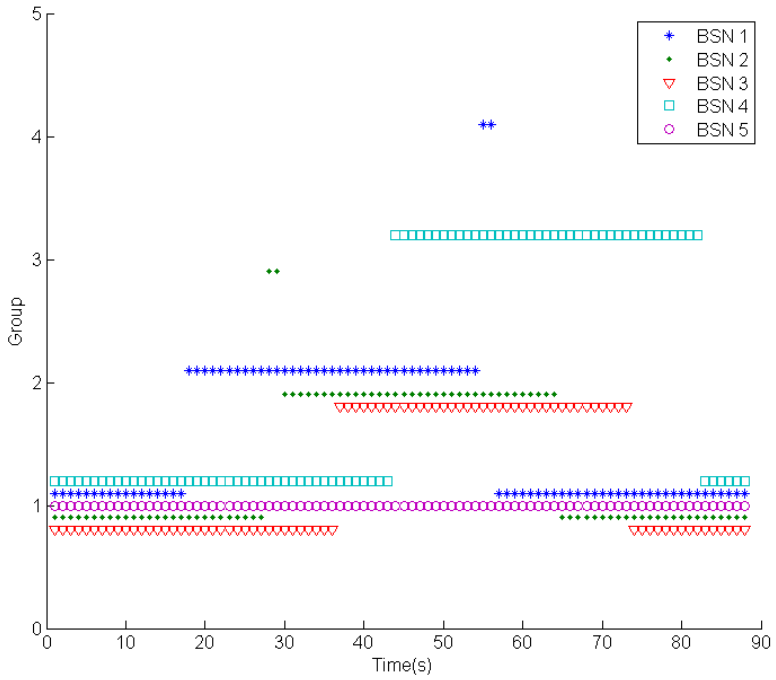


Figure 7.18: Grouping results with 5 BSNs

schedule for each group, if the nearest distance between two groups are below the threshold, we can adjust the transmission schedule of both groups to avoid BSNs with close distance transferring data at the same time. However, considering staying in a line is not a common scenario, since most BSNs or people tend to naturally gather in groups, our system chooses to use the average linkage criteria without performing second level optimization.

7.6.3 Effectiveness

With the parameters of the grouping algorithm discussed in previous sections, we show the effectiveness of the grouping algorithm in this section. In the experiment, 5 BSNs stay together in the beginning, and then perform following movements:

1. BSN 1 walks away
2. BSN 2 walks away, and joins BSN 1
3. BSN 3 walks away, and joins BSN 1 and 2
4. BSN 4 walks away, but in a different direction, not joining BSN 1, 2, and 3
5. BSN 1 walks back to BSN 5

6. BSN 2 walks back to BSN 1 and 5
7. BSN 3 walks back to BSN 1, 2, and 5
8. BSN 4 walks back to BSN 1, 2, 3, and 5

The time interval between each movement is about 10 seconds. Figure 7.18 shows the grouping results during this period. The parameters used in the grouping algorithm is the same as we discussed in previous sections: moving average window $w = 4s$, beacon interval $m = 1s$, RSSI clustering threshold is $-70dBm$, and the linkage criteria used in hierarchical clustering is average. In the experiment we observe grouping results correctly correlated to the locomotion of BSNs, and the grouping results update mostly within two seconds after the movement starts. During the time one BSN walks away from one group to another, it can be temporarily assigned to a new group, for example, when BSN 2 left Group 1 to join Group 2, it was temporarily grouped into Group 3. This also happened when BSN 1 moved from Group 2 back to Group 1, it was first grouped into a separate Group 4. This shows there is a dramatic change of link conditions when people turn around and start walking because these movements change the reflection and shadowing effects.

7.7 Integration Evaluation

In the previous sections, we have evaluated all three pillars of our QoS solution: profiling and scheduling, power adaptation, and grouping. Based on profiles of the BSN hardware platform and two BSN applications defined in Section 7.3 and key parameters determined for scheduling in Section 7.4, power adaptation in Section 7.5, and grouping in Section 7.6, we focus on the integral performance of the system. We show how our system performs comparing to the base line solution of using CSMA/CA. The comparison metrics are packet loss, bandwidth utilization, and fidelity of applications.

7.7.1 Experiment Setup

In this section, we discuss the setup of integration tests. We made a few decisions to make the comparison between our solution and the baseline CSMA more reasonable.

First, we only compare the operation conditions on one channel, because BSNs on each channel are scheduled using the same set of algorithms, the performance of our QoS solution is similar across channels. The performance of CSMA network is also channel agnostic. Therefore, in our solution, all BSNs running our QoS solutions are allocated to Channel 26.

In the tests, we use up to 5 BSNs as described in Section 7.2. Based on the BSN platform and application profiles defined in Section 7.3, we can get the BSN settings computed as in Section 7.4.1. The reason to use no more than 5 BSNs is that according to the bandwidth requirement we calculate in Section 7.4.2 and the effective channel bandwidth we measured in Section 7.4.3, when 5 BSNs are in the same group, one channel can accommodate at most $\lfloor 90864.20/19008 \rfloor = 4$ BSNs operating at the high fidelity setting with 120Hz sampling rate. The fifth BSN can only operate under the low fidelity setting with 50Hz sample rate. By using 5 BSNs, we should be able to see the change of BSN settings as 5 BSNs gather and scatter.

Increasing the number of BSNs after 5 is not that interesting since the only difference it makes is to make more BSNs operate at the low fidelity setting. For CSMA, more BSNs on the same channel will make results worse when BSNs are nearby.

The transmission for our system is scheduled so that each node caches the data collected from LSM303DLM sensor boards and sends these data continuously to the aggregator within the delay requirement one after another. However, nodes using the default CSMA don't have a transmission schedule. In our comparison, to minimize the packet delay, nodes using CSMA sends packets with 24 bytes of payload which contains 4 samples of acceleration data form LSM303DLM as soon as the data is ready, so the time interval to send a packet is about is $1/120 \times 4 \approx 34$ milliseconds.

We performed both static and dynamic experiments to study the integral performance of the system. In static experiments, BSNs don't move around, but the number of BSNs in the vicinity varies from 1 to 5. The static experiments are to show the how number of BSNs affects the performance of our solution and CSMA.

In the dynamic results, 5 BSNs are in a living area of 10 square meters, BSNs can move around freely. In the dynamic results we show how the profile switching works, and how the PRR and data throughput changes over time for both our solution and CSMA. To synchronize the performance of our solution and CSMA over time, each BSNs have two set of motes: one set runs our BSN QoS solution on Channel 26 with control messages sent on Channel 15, the other set sends data using default CSMA MAC provided by TinyOS on Channel 25.

7.7.2 Static Results

In the static experiments we vary the number of BSNs within 2 square meters from 1 to 5. The close distance between BSNs are to guarantee they are clustered into the same group in our solution, which helps avoid the interference of rescheduling and transmission schedule updates. When in the

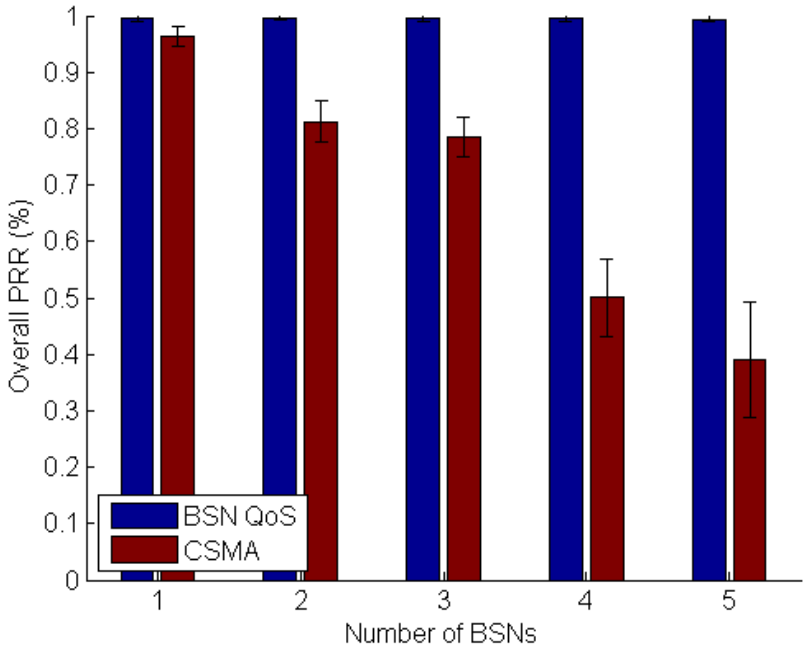


Figure 7.19: Overall PRR of BSN QoS and CSMA with different number of BSNs.

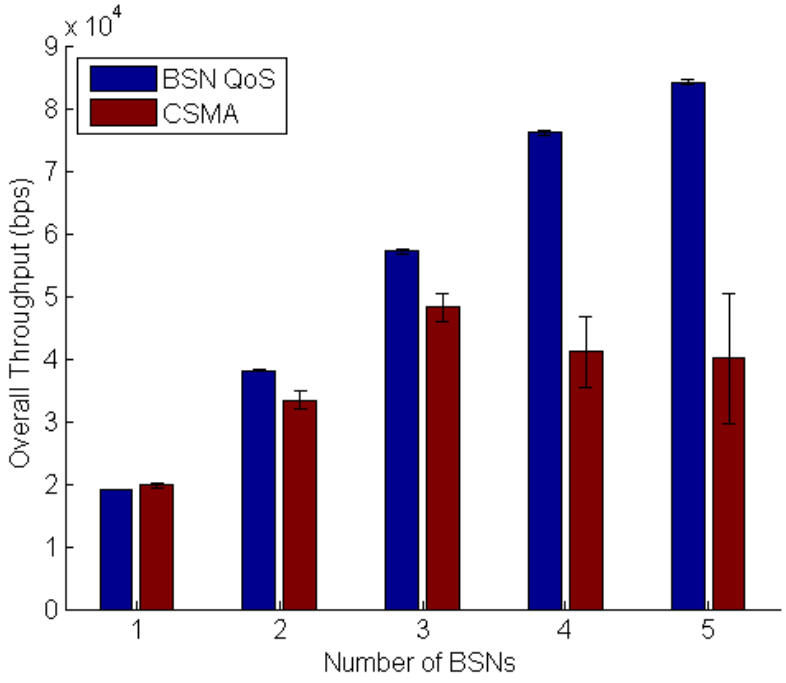


Figure 7.20: Overall throughput of BSN QoS and CSMA with different number of BSNs.

same group, BSNs are scheduled to send packets one by one within the delay requirement of 1 second. The data sending time for each node is

$$c = \left\lceil \frac{120 \times 6 \times 8 \times 1000}{1000 \times 28 \times 8} \right\rceil \times (0.075 \times (28 + 18) + 0.6) \approx 106ms$$

when the BSN uses the high fidelity setting with 120Hz sampling rate, or

$$c = \left\lceil \frac{50 \times 6 \times 8 \times 1000}{1000 \times 28 \times 8} \right\rceil \times (0.075 \times (28 + 18) + 0.6) \approx 45ms$$

when the BSN uses the low fidelity setting with 50Hz sampling rate according to Equation (7.4).

The traffic pattern for CSMA is that each node sends four samples as soon as data is ready, so the time interval to send a packet is $1/120 \times 4 \approx 34ms$

Figure 7.19 shows the overall PRR of our system and CSMA when the number of BSNs changes from 1 to 5. Our solution performs better than the CSMA across the board with near 100% PRR, while CSMA performs much worse due to the interference between nodes. When there are 5 BSNs, the overall PRR for CSMA nodes is about 0.39, and the lowest PPR can be as low as 0.22. Either way, considering $120 \times 0.39 = 46.80$ and $120 \times 0.22 = 26.40$, with PRR as lows as these the fidelity of the two applications cannot be guaranteed since even the lowest amount of data needed which is 50 samples of acceleration data per second cannot be transferred.

Figure 7.20 shows the overall throughput of our system and CSMA when different number of BSNs are close to each other. With near to 100% PRR, the data throughput per channel increases with the number of BSNs linearly using our solution. When BSN 5 is added to the group the data throughput increases less since it is forced to use the low fidelity setting with 50Hz sampling rate. However, for CSMA,when the number of BSNs increases after 3, the data throughput becomes even lower due to the interference between nodes. When 5 BSNs are around, the overall data throughput of our solution is more than twice that of CSMA.

7.7.3 Dynamic Results

Dynamic experiments are used to demonstrate how our system compares to the CSMA over time. To correlate the performance of our system and that of the CSMA, we use two set of nodes in each BSN: one set transmits data using our QoS solution, the other set uses CSMA. The two set of nodes transfer at different channels so that there is no interference between them.

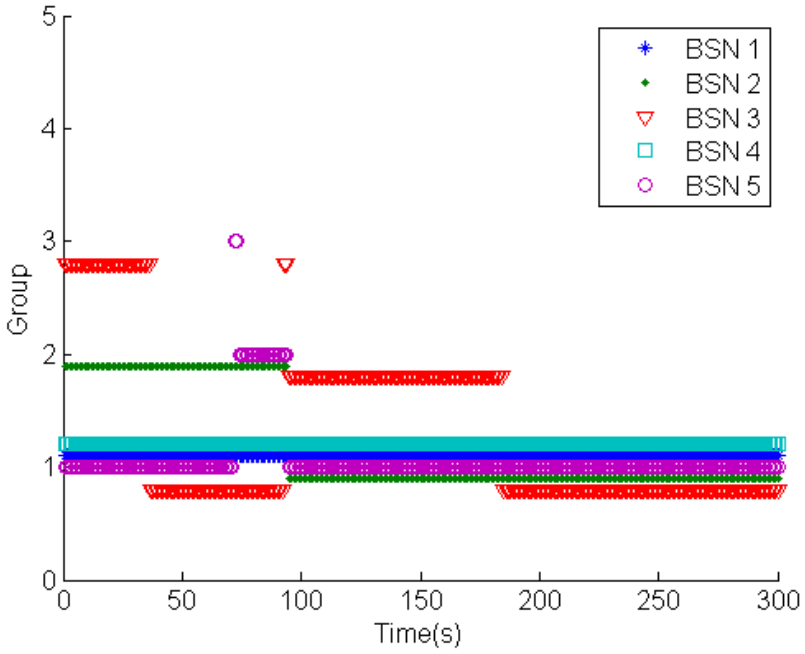


Figure 7.21: Changes of grouping results over time.

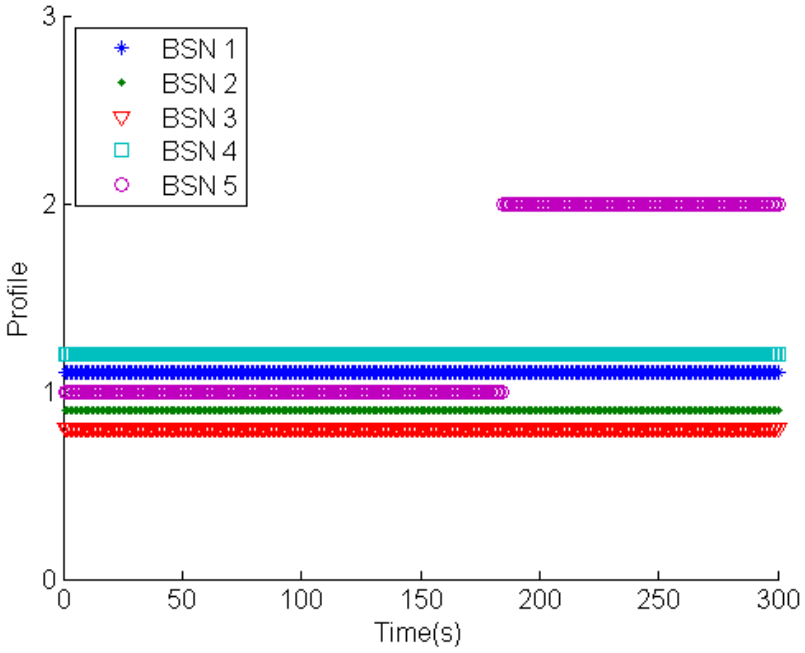


Figure 7.22: Changes of BSN settings over time.

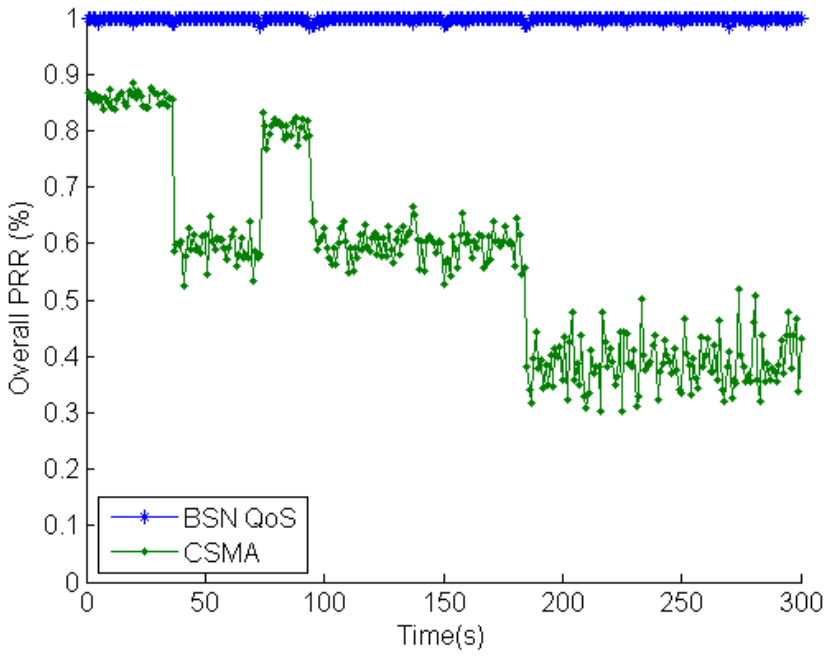


Figure 7.23: Changes of overall PRR of BSN QoS and CSMA over time.

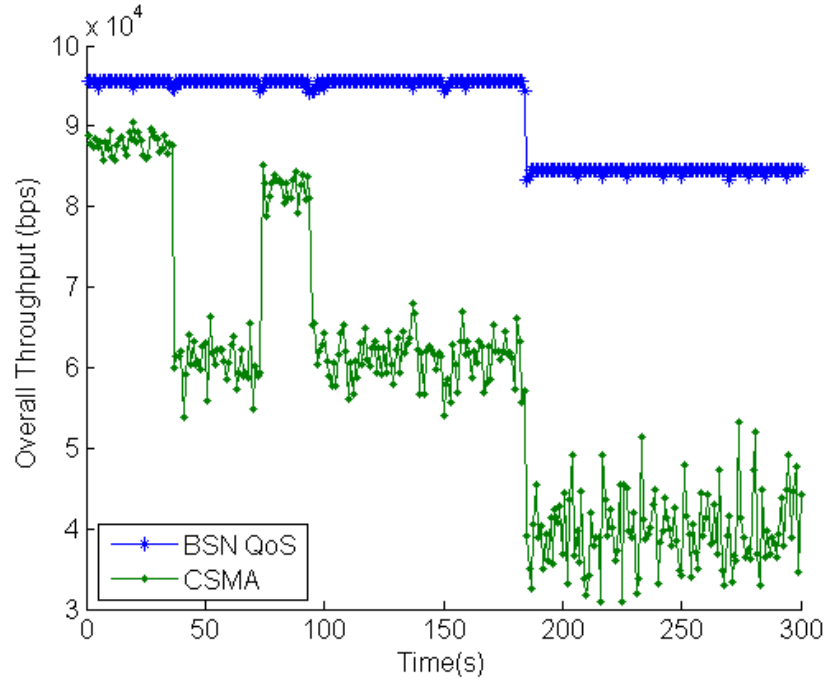


Figure 7.24: Changes of overall throughput of BSN QoS and CSMA over time.

Figure 7.21 shows the grouping results of our algorithm over a period of 5 minutes. Though not very accurate, the grouping results generally correlate to the relative locations of BSNs. Therefore, the moving sequences shown in Figure 7.21 is

1. BSN 3 moves to BSN 1, 4, and 5
2. BSN 5 leaves BSN 1, 3, and 4, then joins BSN 2
3. BSN 2 and 5 move to BSN 1 and 4, while BSN 3 moves away from BSN 1 and 4
4. BSN 3 joins BSN 1, 2, 4, and 5

Figure 7.22 shows the BSN setting selected for each BSN over time. Before all BSNs are in the same vicinity, all BSNs use Setting 1 which is the high fidelity setting. When all 5 BSNs are together, BSN 5 switched to Setting 2 which samples data at 50Hz instead of 120Hz.

Figure 7.23 shows the overall PRR of our QoS solution and CSMA over the moving sequence we concluded form Figure 7.21. We can see some impacts to the PRR of our system when BSNs regroup. This is because when BSNs move towards each other, there can be interference between them in a short period when they are not grouped together. However, our system reaches PRR above 97% all the time. Though there are lots of fluctuations for the PRR of CSMA, we can still see the general pattern that the more BSNs are around, the lower the PRR.

Figure 7.24 show the overall data throughput of our QoS solution and CSMA. As shown in the figure, our solution achieves higher throughput all the time, and the difference between our solution and CSMA becomes larger when more BSNs are close to each other.

7.8 Discussion

In this chapter we unified two BSN applications including fall detection and in-person interaction monitoring with our BSN hardware platform which uses TelosB motes with LSM303DLM sensor boards to collect acceleration data.

In Section 7.2 we described our hardware platform in detail, which helps us to profile the platform in Section 7.3.1. The profiling of the two BSN applications in Section 7.3.2 and Section 7.3.3 are based on our work in Chapter 5 and Chapter 6. This profiling work not only shows how we can separate the hardware details from the application requirements, but they are also the important input to our BSN system to schedule transmissions using the algorithms proposed in Section 4.5 and Section 4.6.

Then in Section 7.4, Section 7.5, and Section 7.6 we discussed how to choose key parameters for our QoS system. The effective bandwidth and the transmission time needed for TelosB motes to transfer data is calculated based on our measurement to make sure the feasibility of our scheduling algorithm. The power adaptation threshold RT_L and RT_H are selected based on the relation between RSSI and PRR. The moving average window size w and the beacon interval m are selected based on both simulation and real measurements to guarantee the responsiveness of the grouping algorithm. Based on RT_L and RT_H we discussed how to select the RSSI threshold in hierarchical clustering to group BSNs. The threshold is selected based on the relation between interference RSSI and the PRR of the subject BSN. We also discussed the boarder interference problem and the selection of distance metrics in clustering.

At last in Section 7.7, we compared our system to a baseline solution using CSMA/CA under both static and dynamic situations. The integration test results show that when multiple BSNs are near to each other, our QoS solution achieves higher PRR, more effective bandwidth utilization, and better application fidelity.

Chapter 8

Conclusions

8.1 Summary of Dissertation

Most of current work on BSNs focus on problems within one BSN. However, BSN systems have become more and more popular as they provide a cheaper way to perform both accident detection and health monitoring. How to coordinate BSN systems to achieve reliable communication and how to guarantee the fidelity of data used by BSN applications have become urgent problems. In this dissertation, we proposed a QoS framework to build reliable multi-body, multi-function BSN systems.

First in Chapter 3, we performed an empirical study to investigate the characteristics of on-body BSN links. We studied how the body shadowing effect, the interference within on BSN, and the interference between BSNs affect the link qualities of BSN systems. This study provides us with insights on both the problems and solutions. As shown in Section 3.2, body shadowing severely impacts BSN link qualities, but it can be overcome with higher transmission power. We also show how intra- and inter-BSN interference can cause packet loss. Coordinating the transmission schedules from nodes to the aggregator can help avoid interference. The study also demonstrates the most distinguishing character of BSNs: the highly dynamic body movement and locomotion, which constantly changes the body shadowing and environments where BSN operates. Therefore, a solution to guarantee the reliability of BSNs must deal with and utilize this constancy of variability.

We proposed our BSN QoS solution in Chapter 4. In the solution, we use power adaptation to dynamically adjust the transmission power of BSN nodes to use the minimum power needed to overcome the body shadowing effect and maintain the link quality of node-to-aggregator traffic. Power adaptation also saves energy and reduces the interference range of BSNs at the same time.

To deal with inter-BSN interference as well as utilize the frequent locomotion of BSNs, our solution dynamically groups BSNs into separate groups according to their impacts on other BSNs which are measured by RSSI. In this way, we narrow down the problem of transmission scheduling to one group of nearby BSNs, which make our solution scalable to coordinate large amount of BSNs.

Since the ultimate goal for reliable BSN communication is for developing reliable BSN applications, we propose a profiling approach to integrate the requirements of multiple BSN applications into the transmission scheduling of BSNs in the same group. Profiles are not only input to the scheduling algorithm, they also provide a way to separate BSN hardware platforms and application requirements, which is a good start for application independent BSN platforms and platform independent BSN applications.

We also proposed algorithms to validate application profiles against BSN profiles to produce possible BSN settings, to combine BSN settings to produce settings that can satisfy the requirements of multiple applications, to compute the preference of possible BSN settings, and to calculate the bandwidth requirement of a BSN setting. Based on the BSN settings of multiple BSNs, we use rate monotonic scheduling algorithm which has been widely used in real-time operating systems to schedule the transmission of multiple BSNs in the same group. Application requirements such the sampling rate and the delay limit are integrated into the process of transmission scheduling so that the fidelity of applications can be guaranteed.

Then we designed and implemented two BSN applications: fall detection in Chapter 5 and in-person interaction monitoring in Chapter 6. These two applications represent two common types of BSN applications: event detection and health monitoring, respectively. Our fall detection algorithm uses a context-free grammar to define various falls as sequences of sensor events, which enables the system to be easily tuned and extended to detect new types of falls. The use of clustering and learning techniques to recognize activity levels and postures makes our method specific to the subject and improves the recognition accuracy. Our method can also detect slow falls by using high level features such as postures and location information. Our in-person interaction monitoring application uses a multi-modal approach to detect social interactions from multiple primitives including activities identified by acceleration data, proximity of other people deduced from RSSI, and the presence of human speech from microphones. Speech volume analysis across multiple BSNs provides more details about the quality of interactions such as how much the subject speaks. The evaluation results of the fall detection and in-person interaction monitoring show both of them achieve high accuracy.

The approaches used to develop these two applications can be combined to simplify the process of BSN application development. By defining a list of primitives and then using context-free grammar to

define various activities with these primitives, the development of an event detection BSN application can be as easy as an if statement. (Or maybe a switch statement, if you want to detect multiple events and make your program look more elegant.)

In Chapter 7 we unified the proposed QoS solution and two BSN applications by building a BSN platform with TelosB motes and LSM303DLM sensor boards. During the process, the three pillars of our QoS solutions including profiling and scheduling, power adaptation, and grouping are all separately evaluated. Based on the profiles of the BSN platform and two applications, we carefully calculated the effective bandwidth of a Zigbee channel in our system and the time needed to transfer data for each node under different BSN settings. We also discussed the selection key parameters for power adaptation and grouping to guarantee their correctness and effectiveness. With profiles of BSN and applications defined and all parameters of the QoS solution determined, we performed integration testing to demonstrate our system achieves higher PRR, utilizes bandwidth utilization more effectively, and better satisfies application fidelity requirements in both static and dynamic situations.

All in all, in this dissertation, we studied the BSN system in a holistic way from lower level transmission scheduling, to profiles that separate the details of BSN platforms and the requirements of applications, to the development of two representative BSN applications, and finally to the unification of our QoS solution and BSN applications. Combining all these work, we can build reliable multi-body, multi-function BSN systems.

8.2 Discussion and Future Work

In this section we propose potential extensions in the future as below:

- Potential work to solve the boarder interference problem caused by using the average linkage criteria when grouping nearby BSNs. To balance the need to maximize bandwidth utilization and to reduce transmission collisions, we choose to use the average distance between elements of two clusters as the distance between clusters. In this case, Section 7.6.2 shows BSNs belong to different groups can interfere with each other when they are close. The potential solution is to adjust the transmission schedule based on the values in the similarity matrix. If there is strong interference between two BSNs of different groups, we can switch the slots so that BSNs of different groups but within interference range of each other don't transfer data at the same time.

- Potential work to extend the contents of profiling. Our current effort to profile BSN platforms mainly focuses on the profiling of the sensors. However, a more complete BSN profile including more information, such as different power levels of CPU, RAM, and radio chips, is needed for our system to switch settings to guarantee the lifetime of BSN systems.

- Potential work to port our QoS solution to smartphones. Currently our system uses laptops connected with TelosB motes as aggregators. The ideal platform would be smartphones. Porting the QoS solution to smartphones can make our system much more convenient to use. Using smartphones also adds a set of sensors, such as accelerometers, microphones, light sensors, and GPS, to the standard sensor list of our system. In this case, features requiring these sensors can be added such as using binary search to faster find the next appropriate power level when the subject is performing activities of high dynamical levels.

- Potential work to find clear channels using noise flooring sensing. In our system, we make all WiFi networks operate on Channel 1 to avoid interference from WiFi. Potential work can be done to use noise floor sensing to find which channels are free of interference, so that the channels to be allocated to BSNs can be dynamically selected. If the channel condition changes, we can migrate BSNs to channels with less noise.

- Potential work to improve the fall detection application. The use of context-free grammar in our fall detection algorithm enables our solution to be easily extended and tuned. Extending the rule set to define more types of slow and fast falls can dramatically improve the practicality of our method. In addition, integrating physiological data to the fall detection framework can also increase fall detection accuracy. For example, low blood pressure is a good indicator of dizziness and thus falls.

- Potential work to improve the in-person interaction monitoring application. Our solution uses a multi-modal approach to monitor in-person interactions. Increasing types of primitives such as using social networks can enable our system to detect more types of social activities and achieve better accuracy. Another possible improvement is to automatically define new types of social activities based on the calendar events and according values of primitives.

- Potential work to build a platform for developing BSN applications. By combing the use of context-free grammar and the definition of primitives used in our fall detection and in-person interaction monitoring, potentially we can build a framework to simplify BSN programming in which the end user only needs to specify an event using a sequence of primitive values.

Bibliography

[1] Administration on Aging. A profile of older americans: 2015. Technical report, U.S. Department of Health and Human Services, 2015.

[2] Thaddeu R. F. Fulford-Jones, Gu-Yeon Wie, and Matt Welsh. A portable, low-power, wireless two-lead ekg system. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 2141–2144, San Francisco, CA, USA, Sept. 2004.

[3] Adam T. Barth, Mark A. Hanson, Harry C. Powell Jr., and John Lach. Tempo 3.1: A body area sensor network platform for continuous movement assessment. In *Proceedings of the 2009 6th International Workshop on Wearable and Implantable Body Sensor Networks (BSN'09)*, pages 71–76, Berkeley, CA, USA, June 2009.

[4] Jay Chen, Karric Kwong, Dennis Chang, Jerry Luk, and Ruzena Bajcsy. Wearable sensors for reliable fall detection. In *Proceedings of the 27th Annual International Conference of the IEEE EMBS*, pages 3551–3554, Shanghai, China, Sept. 2005.

[5] Qiang Li, John A. Stankovic, Mark A. Hanson, Adam T. Barth, and John Lach. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Proceedings of the 2009 6th International Workshop on Wearable and Implantable Body Sensor Networks (BSN'09)*, pages 138–143, Berkeley, CA, USA, June 2009.

[6] Chulsung Park and Pai H. Chou. An ultra-wearable, wireless, low power ecg monitoring system. In *Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS)*, London, UK, 2006.

[7] Rahul C. Shah, Lama Nachman, and Chieh yih Wan. On the performance of bluetooth and IEEE 802.15.4 radios in a body area network. In *Proceedings of the ICST 3rd International Conference on Body Area Networks (BodyNets)*, Tempe, AZ, USA, Mar. 2008.

[8] Rahul C. Shah and Mark Yarvis. Characteristics of on-body 802.15.4 networks. In *Proceedings of the 2nd IEEE Workshop on Wireless Mesh Networks (WiMesh)*, pages 138–139, Reston, VA, USA, Sept. 2006.

[9] Emiliano Miluzzo, Xiao Zheng, Kristóf Fodor, and Andrew T. Campbell. Radio characterization of 802.15.4 and its impact on the design of mobile sensor networks. In *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN)*, Bologna, Italy, Feb. 2008.

[10] David Jea and Mani B. Srivastava. Packet delivery performance for on-body Mica2dot wireless sensor netowrks. In *Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Santa Clara, CA, USA, Sept. 2005. Poster.

[11] Anirudh Natarajan, Mehul Motani, Buddhika de Silva, Kok-Kiong Yap, and K. C. Chua. Inverstigating network architecturs for body sensor networks. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet)*, pages 19–24, San Juan, Puerto Rico, USA, June 2007.

[12] Anirudh Natarajan, Buddhika de Silva, Kok-Kiong Yap, and Mehul Motani. Link layer behavior of body sensor area networks at 2.4 ghz. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Beijing, China, Sept. 2009.

[13] Wan Du, David Navarro, and Fabien Mieyeville. Performance evaluation of ieee 802.15.4 sensor networks in industrial applications. *International Journal of Communication Systems*, 28(10):1657–1674, July 2015.

[14] Wafa Badreddine, Claude Chaudet, Federico Petruzzi, and Maria Potop-Butucaru. Broadcast strategies in wireless body area networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, number 8 in MSWiM'15, pages 83–90, Cancun, Mexico, 2015. ACM.

[15] Laura Marie Feeney and Viktoria Fodor. Reliability in co-located 802.15.4 personal area networks. In *Proceedings of the 6th ACM International Workshop on Pervasive Wireless Healthcare*, MobiHealth'16, pages 5–10, Paderborn, Germany, 2016.

[16] Attaphongse Taparugssanagorn, Alberto Rabbachin, Matti Hämäläinen, Jani Saloranta, and Jari Iintti. A review of channel modeling for wireless body area network in wireless medical communications. In *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications*, Saariselkä, Finland, 2008.

[17] Kamya Yekeh Yazdandoost and Kamran Sayrafian-Pour. Channel model for body area network (BAN). Technical report, IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs), Apr. 2009.

[18] Muhammad Mahtab Alam, Elyes Ben Hamida, Dhafer Ben Arbia, Mickael Maman, Francesco Mani, Benoit Denis, and Raffaele D'Errico. Realistic simulation for body area and body-to-body networks. *Sensors*, 16(4), 2016.

[19] Ankur Kamthe, Miguel A. Carreira-Perpinan, and Alberto E. Cerpa. M&M: Multi-level markov model for wireless link simulations. In *Proceedings of the 7nd International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 57–70, Berkeley, CA, USA, Nov. 2009.

[20] Jan-Hinrich Hauer, Vlado Handziski, and Adam Wolisz. Experimental study of the impact of WLAN interference on IEEE 802.15.4 body area networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, Feb. 2009.

[21] X. Gui H. Ghayvat, S.C. Mukhopadhyay. Issues and mitigation of interference, attenuation and direction of arrival in ieee 802.15.4/zigbee to wireless sensors and networks based smart building. *Measurement*, 86:209–226, May 2016.

[22] D. Ben Arbia, M. M. Alam, R. Attia, and E. Ben Hamida. Data dissemination strategies for emerging wireless body-to-body networks based internet of humans. In *Proceedings of the 11th International IEEE Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'15)*, pages 1–8, Oct. 2015.

[23] Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, John A. Stankovic, and Tian He. ATPC: Adaptive transmission power control for wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 223–236, Boulder, Colorado, USA, 2006.

[24] Muhannad Quwaider, Jayanthi Rao, and Subir Biswas. Transmission power assignment with postural position inference for on-body wireless communication links. *ACM Transactions on Embedded Computing System (TECS)*, 10:14:1–14:27, Aug. 2010.

[25] Buddhika de Silva, Anirudh Natarajan, and Mehul Motani. Inter-user interface in body sensor networks: Perliminary investigation and an infrastructure-based solution. In *Proceedings of the 6th International Workshop on Body Sensor Networks (BSN)*, Berkeley, CA, USA, June 2009.

[26] Anthony D. Wood, John A. Stankovic, Gilles Virone, Leo Selavo, Zhimin He, Qiuhua Cao, Thao Doan, Yafeng Wu, Lei Fang, and Radu Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 22(4):26–33, July–Aug. 2008.

[27] Huaming Li and Jindong Tan. Heartbeat driven medium access control for body sensor networks. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet)*, pages 25–30, 2007.

[28] Sana Ullah, Henry Higgins, S. M. Riazul Islam, Pervez Khan, and Kyung Sup Kwak. On PHY and MAC performance in body sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2009.

[29] Sana Ullah, Pervez Khan, Young-Woo Choi, Hyung-Soo Lee, and Kyung Sup Kwak. MAC hurdles in body sensor networks. In *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT)*, volume 2, pages 1151–1155, Phoenix Park, Korea, Feb. 2009.

[30] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-MAC: a hybrid MAC for wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, USA, Nov. 2005.

[31] Sabin Bhandari and Sangman Moh. Mac protocols for wireless body area sensor networks: A comparative review. *International Journal of Applied Engineering Research*, 11(16):8810–8819, 2016.

[32] Long Hu, Yin Zhang, Dakui Feng, Mohammad Mehedi Hassan, Abdulhameed Alelaiwi, and Atif Alamri. Design of qos-aware multi-level mac-layer for wireless body area network. *Journal of Medical Systems*, 39(12), Dec. 2015.

[33] Rae Hyeon Kim, Jeong Gon Kim, and Bang Won Seo. Chanel access with priority for urgent data in medical wireless body sensor networks. *International Journal of Applied Engineering Research*, 11(2):1162–1166, 2016.

[34] Sabin Bhandari and Sangman Moh. A priority-based adaptive mac protocol for wireless body area networks. *Sensors*, 16(3), 2016.

[35] Adnan Khan, Syed Irfan Ullah, Arshad Farhad, A Wajid Ullah Khan, Abdus Salam, Muazzam A. Khan, and M. Sikandar Hayat Khiyal. Load adaptive dynamic protocol for qos provision in wireless body area sensor networks. *International Journal of Computer Science and Information Security (IJCSIS)*, 14(5):325–330, May 2016.

[36] Jie Dong, Yu Ge, and David B. Smith. Two-hop relay-assisted cooperative communication in wireless body area networks: An empirical study. *ACM Transactions on Sensor Networks*, 12(4):32:1–32:13, Sept. 2016.

[37] Nadeem Javaid, Ashfaq Ahmad, Yahya Khan, Zahoor Ali Khan, and Turki Ali Alghamdi. A relay based routing protocol for wireless in-body sensor networks. *Wireless Personal Communications*, 80(3):1063–1078, Feb. 2015.

[38] Sidrah Yousaf, Nadeem Javaid, Umar Qasim, Nabil Alrajeh, Zahoor Ali Khan, and Mansoor Ahmed. Towards reliable and energy-efficient incremental cooperative communication for wireless body area networks. *Sensors*, 16(3), 2016.

[39] Javed Iqbal Bangash, Abdul Waheed Khan, and Abdul Hanan Abdullah. Data-centric routing for intra wireless body sensor networks. *Journal of Medical Systems*, 39(9), Sept. 2015.

[40] Tariq Umer, Muhammad Amjad, Muhammad Khalil Afzal, and Muhammad Aslam. Hybrid rapid response routing approach for delay-sensitive data in hospital body area sensor network. In *Proceedings of the 7th International Conference on Computing Communication and Networking Technologies (ICCCNT'16)*, pages 3:1–3:7, Dallas, TX, USA, 2016.

[41] Emad Felemban, Chang-Gun Lee, and Eylem Ekici. MMSPEED: Multipath multi-speed protocol for qos guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[42] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 46–55, Providence, RI, USA, May 2003.

[43] José-F Martínez, Ana-B Garcí, Iván Corredor, Lourdes López, Vicente Hernández, and Antonio Dasilva. Qos in wireless sensor networks: Survey and approach. In *Proceedings of the 2007 Euro American conference on Telematics and information systems (EATIS'07)*, pages 1–8, Faro, Portugal, 2007.

[44] Gang Zhou, Jian Lu, Chieh-Yih Wan, Mark D. Yarvis, and John A. Stankovic. BodyQoS:adaptive and radio-agnostic QoS for body sensor networks. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM)*, pages 565–573, Phoenix, AZ, USA, Apr. 2008.

[45] Gang Zhou, Chieh-Yih Wan, Mark D. Yarvis, and John A. Stankovic. Aggregator-centric QoS for body sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 539–540, Cambridge, MA, USA, Apr. 2007. Poster.

[46] Zhen Ren, Gang Zhou, Andrew Pyles, Matthew Keally, Weizhen Mao, and Haining Wang. BodyT2: Throughput and time delay performance assurance for heterogeneous bsns. In *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM)*, Shanghai, China, Apr. 2011.

[47] Zhiqiang Liu, Bin Liu, Chang Chen, and Chang Wen Chen. Energy-efficient resource allocation with qos support in wireless body area networks. In *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM'15)*, pages 1–6, Dec. 2015.

[48] Sharbani Pandit, Krishanu Sarker, Md. Abdur Razzaque, and A. M. Jehad Sarkar. An energy-efficient multiconstrained qos aware mac protocol for body sensor networks. *Multimedia Tools and Applications*, 74(14):5353–5374, July 2015.

[49] Nedal Ababneh, Nicholas Timmons, and Jim Morrison. A cross-layer qos-aware optimization protocol for guaranteed data streaming over wireless body area networks. *Telecommunication Systems*, 58(2):179–191, 2015.

[50] Norbert Noury. A smart sensor for the remote follow up of activity and falldetection of the elderly. In *Proceedings of the 2nd International IEEE EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, pages 314–317, 2002.

[51] U. Lindemann, A. Hock, M. Stuber, W. Keck, and C. Becker. Evaluation of a fall detector based on accelerometers: A pilot study. *Medical and Biological Engineering and Computing*, 43(5):548–551, Oct. 2005.

[52] Weihao Qu, Feng Lin, Aosen Wang, and Wenyao Xu. Evaluation of a low-complexity fall detection algorithm on wearable sensor towards falls and fall-alike activities. In *Proceedings of the 2015 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–6, Dec. 2015.

[53] T. N. Gia, I. Tcarenko, V. K. Sarker, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Iot-based fall detection system with energy efficient sensor nodes. In *Proceedings of the 2016 IEEE Nordic Circuits and Systems Conference (NORCAS)*, pages 1–6, Nov. 2016.

[54] Yoosuf Nizam, Mohd Norzali Haji Mohd, and M. Mahadi Abdul Jamil. A study on human fall detection systems: Daily activity classification and sensing techniques. *International Journal of Integrated Engineering*, 8(1):35–43, 2016.

[55] Minoo Rashidpour, Fardin Abdali Mohammadi, and Abdolhossein Fathi. Fall detection using adaptive neuro-fuzzy inference system. *International Journal of Multimedia and Ubiquitous Engineering*, 11(4):91–106, 2016.

[56] Laurence Z. Rubenstein and Karen R. Josephson. The epidemiology of falls and syncope. *Clinics in Geriatric Medicine*, 18(2):141–158, May 2002.

[57] A. K. Bourke, J. V. O'Brien, and G. M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait and Posture*, 26:194–199, 2007.

[58] M.N. Nyan, Francis E.H. Tay, and E. Murugasu. A wearable system for pre-impact fall detection. *Journal of Biomechanics*, 41(16):3475–3481, Dec. 2008.

[59] A. Díaz, M. Prado, L. M. Roa, J. Reina-Tosina, and G. Sánchez. Preliminary evaluation of a full-time falling monitor for the elderly. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 2180–2183, San Francisco, USA, Sept. 2004.

[60] M. Prado, J. Reina-Tosina, and L. Roa. Distributed intelligent architecture for falling detection and physical activity analysis in the elderly. In *Proceedings of the 2nd Joint EMBS/BMES conference*, pages 1910–1911, Houston, USA, Oct. 2002.

[61] M. J. Mathie, J. Basilakis, and B. G. Celler. A system for monitoring posture and physical activity using accelerometers. In *Proceedings of the 23rd Annual International Conference of the IEEE EMBS*, pages 3654–3657, Istanbul, Turkey, Oct. 2001.

[62] Marrit Kangas, Antti Konttila, Ilkka Winblad, and Timo Jämsä. Determination of simple thresholds for accelerometry-based parameters for fall detection. In *Proceedings of the 29th Annual International Conference of the IEEE EMBS*, pages 1367–1370, Lyon, France, Aug. 2007.

[63] Do Un Jeong, Se Jin Kim, and Wan Young Chung. Classifiation of posture and movement using a 3-axis accelerometer. In *Proceedings of the 2007 International Conference on Convergence Information Technology*, pages 837–844, Gyeongju, Korea, 2007.

[64] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. Mobile phone-based pervasive fall detection. *Journal of Personal and Ubiquitous Computing*, 14(7):633–643, Oct. 2010.

[65] A. Lisowska, G. Wheeler, V. C. Inza, and I. Poole. An evaluation of supervised, novelty-based and hybrid approaches to fall detection using silmee accelerometer data. In *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 402–408, Dec. 2015.

[66] Norbert Noury, Thierry Hervé, Vincent Rialle, Gilles Virone, Eric Mercier, Gilles Morey, Aldo Moro, and Thierry Porcheron. Monitoring behavior in home using a smart fall sensor and position sensors. In *Proceedings of the 1st International IEEE EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, pages 607–610, Lyon, France, Oct. 2000.

[67] N. Noury, P. Barralon, G. Virone, P. Boissy, M. Hamel, and P. Rumeau. A smart sensor based on rules and its evaluation in daily routines. In *Proceedings of the 25th Annual International Conference of the IEEE EMBS*, pages 3286–3289, Cancun, Mexico, Sept. 2003.

[68] Amit Purwar, Do Un Jeong, and Wan Young Chung. Activity monitoring from real-time triaxial accelerometer data using sensor network. In *Proceedings of the 2007 International Conference on Control, Automation and Systems*, pages 2402–2406, COEX, Seoul, Korea, Oct. 2007.

[69] Peter Leijdekkers, Valérie Gay, and Elaine Lawrence. Smart homecare system for health tele-monitoring. In *Proceedings of the 1st International Conference on the Digital Society*, Guadeloupe, French Caribbean, Jan. 2007.

[70] A. K. Bourke and G. M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering and Physics*, 30:84–90, Jan. 2008.

[71] Agustinus Borgy Waluyo, Wee-Soon Yeoh, Isaac Pek, Yihan Yong, and Xiang Chen. Mobisense: Mobile body sensor network for ambulatory monitoring. *ACM Transactions on Embedded Computing Systems (TECS)*, 10(1):13:1–13:30, Aug. 2010.

[72] A. M. Sabatini, G. Ligorio, A. Mannini, V. Genovese, and L. Pinna. Prior-to- and post-impact fall detection using inertial and barometric altimeter measurements. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(7):774–783, July 2016.

[73] Thomas Riisgaard Hansen, J. Mikael Eklund, Jonathan Sprintle, Ruzena Bajcsy, and Shankar Sastry. Using smart sensors and a camera phone to detect and verify the fall of elderly persons. In *Proceedings of the 2005 3rd European Medical and Biological Engineering Conference*, Prague, Czech Republic, Nov. 2005.

[74] Ali Maleki Tabar, Arezou Keshavarz, and Hamid Aghajan. Smart home care network using sensor fusion and distributed vision-based reasoning. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 145–154, Santa Barbara, California, USA, Oct. 2006.

[75] Jia-Luen Chua, Yoong Choon Chang, and Wee Keong Lim. A simple vision-based fall detection technique for indoor video surveillance. *Signal, Image and Video Processing*, 9(3):623–633, 2015.

[76] Erik E. Stone and Marjorie Skubic. Fall detection in homes of older adults using the microsoft kinect. *IEEE Journal of Biomedical and Health Informatics*, 19(1):290–301, Jan 2015.

[77] F. De Backere, F. Ongenae, F. Van den Abeele, J. Nelis, P. Bonte, E. Clement, M. Philpott, J. Hoebeke, S. Verstichel, A. Ackaert, and F. De Turck. Towards a social and context-aware multi-sensor fall detection and risk assessment platform. *Computers in Biology and Medicine*, 64:307–320, Sept. 2015.

[78] http://www.wellawaresystems.com.

[79] http://www.lifelinesys.com/content/lifeline-products/classic-pendant.jsp.

[80] http://www.quietcaresystems.com/index_alt.htm.

[81] Joseph J. Gallo and Barry D. Lebowitz. The epidemiology of common late-life mental disorders in the community: Themes for the new century. *Psychiatric Services*, 50(9):1158–1166, Sept. 1999.

[82] Song-Iee Hong, Leslie Hasche, and Sharon Bowland. Structural relationships between social activities and longitudinal trajectories of depression among older adults. *The Gerontologist*, 49(1):1–11, 2009.

[83] Datong Chen, Jie Yang, Robert Malkin, and Howard D. Wactlar. Detecting social interactions of the elderly in a nursing home environment. *ACM Transactions on Multimedia Computing, Communications and Applications*, 3(1):1–22, Feb. 2007.

[84] Song-Iee Hong, Leslie Hasche, and Sharon Bowland. Structural relationships between social activities and longitudinal trajectories of depression among older adults. *The Gerontologist*, 49(1):1–11, 2009.

[85] Shu Chen and Yan Huang. Recognizing human activities from multi-modal sensors. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI'09)*, pages 220–222, Dallas, TX, USA, June 2009.

[86] Gerald Bieber and Christian Peter. Using physical activity for user behavior analysis. In *Proceedings of the 1st International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)*, pages 94:1–94:6, 2008.

[87] Shu Chen and Yan Huang. Recognizing human activities from multi-modal sensors. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 220–222, June 2009.

[88] Nicky Kern, Bernt Schiele, and Albrecht Schmidt. Multi-sensor activity context detection for wearable computing. In *Ambient Intelligence*, pages 220–232.

[89] Sébastien Faye, Raphael Frank, and Thomas Engel. Adaptive activity and context recognition using multimodal sensors in smart devices. In *Proceedings of the 7th International Conference on Mobile Computing, Applications, and Services (MobiCASE)*, pages 33–50, Berlin, Germany, Nov. 2015.

[90] Sébastien Faye, Nicolas Louveton, Gabriela Gheorghe, and Thomas Engel. A two-level approach to characterizing human activities from wearable sensor data. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 7(3):1–21, Sept. 2016.

[91] Brett Adams, Dinh Phung, and Svetha Venkatesh. Sensing and using social context. *ACM Transactions on Multimedia Computing, Communications and Applications*, 5(2):11:1–11:27, Nov. 2008.

[92] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys'09)*, pages 165–178, Kraków, Poland, June 2009.

[93] Emiliano Miluzzo, Nicholas D. Lane, Shane B. Eisenman, and Andrew T. Campbell. CenceMe - injecting sensing presence into social networking applications. In *Smart Sensing and Context*, volume 4793 of *LNCS*, pages 1–28. 2007.

[94] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Micro Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys'08)*, pages 337–350, Raleigh, NC, USA, Nov. 2008.

[95] Gill Hubbard, Susan Tester, and Murna G. Downs. Meaningful social interactions between older people in institutional care settings. *Aging and Society*, 23(1):99–114, 2003.

[96] Kota Tsubouchi, Osamu Saisho, Junichi Sato, Seira Araki, and Masamichi Shimosaka. Fine-grained social relationship extraction from real activity data under coarse supervision. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 183–187, Osaka, Japan, 2015.

[97] Jamie A. Ward, Gerald Pirkl, Peter Hevesi, and Paul Lukowicz. Towards recognising collaborative activities using multiple on-body sensors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 221–224, Heidelberg, Germany, 2016.

[98] Raghu K. Ganti, Praveen Jayachandran, Tarek F. Abdelzaher, and John A. Stankovic. SATIRE: A software architecture for smart attire. In *Proceedings of MobiSys'06*, Uppsala, Sweden, June 2006.

[99] Muhannad Quwaider and Subir Biswas. Body posture identification using hidden markov model with a wearable sensor network. In *Proceedings of the ICST 3rd International Conference on Body Area Networks (BodyNets'08)*, pages 1–8, Tempe, AZ, USA, Mar. 2008.

[100] Nam Pham and Tarek Abdelzaher. Robust dynamic human activity recognition based on relative energy allocation. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'08)*, pages 525–530, Santorini Island, Greece, June 2008.

[101] Zhen Yu He and Lian Wen Jin. Activity recognition from acceleration data based on discrete cosine transform and svm. In *Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 5041–5044, San Antonio, TX, USA, Oct. 2009.

[102] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Face-to-face social activity detection using data collected with a wearable device. In *Proceedings of the 4th Iberian Conference on Pattern Recognition (IbPRIA'09)*, pages 56–63, Póvoa de Varzim, Portugal, June 2009.

[103] Nicholas D. Lane, Petko Georgiev, Cecilia Mascolo, and Ying Gao. Zoe: A cloud-less dialog-enabled continuous sensing wearable exploiting heterogeneous computation. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 273–286, Florence, Italy, 2015.

[104] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, Sept. 2010.

[105] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In *Proceedings of the 6th ACM conference on Embedded Network Sensor Systems (SenSys)*, pages 337–350, 2008.

[106] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 165–178, 2009.

[107] Brett Adams, Dinh Phung, and Svetha Venkatesh. Sensing and using social context. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 5:11:1–11:27, Nov. 2008.

[108] Simon Flutura, Johannes Wagner, Florian Lingenfelser, Andreas Seiderer, and Elisabeth André. Mobilessi: Asynchronous fusion for social signal interpretation in the wild. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 266–273, Tokyo, Japan, 2016.

[109] Diane J. Cook, Aaron Crandall, Geetika Singla, and Brian Thomas. Detection of social interaction in smart spaces. *Cybernetics and Systems*, 41(2):90–104, Feb 2010.

[110] Paul Lukowicz, Alex "Sandy" Pentland, and Alois Ferscha. From context awareness to socially aware computing. *Pervasive Computing*, 11(1):32–41, Jan. 2012.

[111] Yurong Xu, Yi Ouyang, Zhengyi Le, James Ford, and Fillia Makedon. Mobile anchor-free localization for wireless sensor networks. In *Proceedings of DCOSS*, Santa Fe, NM, USA, June 2007.

[112] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *Proceedings of the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 46–51, Taipei, Taiwan, Nov. 2007.

[113] Răzvan Musăloiu-E. and Andreas Terzis. Minimising the effect of wifi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, 3(1):43–54, 2007.

[114] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, USA, Nov. 2004.

[115] Patrik Moravek, Dan Komosny, Milan Simek, Mojmir Jelinek, David Girbau, and Antonio Lazaro. Investigation of radio channel uncertainty in distance estimation in wireless sensor networks. *Telecommunication Systems*, pages 1–10, 2011.

[116] https://en.wikipedia.org/wiki/Hierarchical_clustering.

[117] NXP Laboratories UK Ltd. *Co-existence of IEEE 802.15.4 at 2.4 GHz*, Nov. 2013.

[118] Crossbow. *MICAz-Based ZigBee and WiFi Coexistence: Avoiding RF Interference Between WiFi and Zigbee.*

[119] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. Surviving wi-fi interference in low power zigbee networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 309–322, 2010.

[120] Jon T. Adams. An introduction to IEEE STD 802.15.4. In *Proceedings of the 2006 IEEE Aerospace Conference*, 2006.

[121] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, Sept. 2011.

[122] Alan C. Shaw. *Real-Time Systems and Software.* John Wiley & Sons, Inc., 2001.

[123] Muhannad Quwaider, Jayanthi Rao, and Subir Biswas. Transmission power assignment with postural position inference for on-body wireless communication links. *ACM Transactions on Embedded Computing Systems (TECS)*, 10(1):14:1–14:27, Aug. 2010.

[124] Martin van de Goor. Indoor localization in wireless sensor networks. Master's thesis, Radboud University Nijmegen, 2009.

[125] Michael Büchler, Silvia Allegro, Stefan Launer, and Norbert Dillier. Sound classification in hearing aids inspired by auditory scene analysis. *EURASIP Journal on Applied Signal Processing*, pages 2991–3002, 2005.