Enabling Energy Efficient Sensing and Computing Systems

by

He Ba

Submitted in Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Wendi B. Heinzelman

Department of Electrical and Computer Engineering

Arts, Sciences and Engineering

Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2015

Biographical Sketch

The author was born in Beijing, China. He received his Bachelor of Science degree in Electrical Engineering from Beijing Institute of Technology in 2008 and his Master of Science degree from the University of Rochester in 2011. He pursued his doctoral research under the direction of Professor Wendi Heinzelman. He worked as an Informatics Intern at UCB Pharma from February 2012 to June 2012. He worked as a Systems Engineering Intern at ASSIA Inc. from June 2014 to October 2014. His research interests lie in the areas of wireless sensor networks, signal processing and mobile computing. After graduation, he will start working at KPMG as a Software Engineer in Big Data from March 2015.

The following publications were a result of work conducted during doctoral study:

He Ba, Ilker Demirkol, and Wendi Heinzelman, "Feasibility and Benefits of Passive RFID Wake-up Radios for Wireless Sensor Networks," in IEEE Global Telecommunications Conference, 2010.

He Ba, Jeff Parvin, Luis Soto, Ilker Demirkol, and Wendi Heinzelman, "Passive RFID-based Wake-up Radios for Wireless Sensor Network," in Wirelessly Powered Sensor Networks and Computational RFID, Springer Publishers, 2012.

He Ba, Na Yang, Ilker Demirkol, and Wendi Heinzelman, "BaNa: A Hybrid Approach for Noise Resilient Pitch Detection," IEEE Statistical Signal Processing Workshop, 2012.

Li Chen, He Ba, Wendi Heinzelman and Andre Cote, "RFID Range Extension with Low-power Wireless Edge Devices," in Proceedings of International Conference on Computing, Networking and Communications, 2013.

Li Chen, Stephen Cool, He Ba, Wendi Heinzelman, Ilker Demirkol, Ufuk Muncuk, Kaushik Chowdhury and Stefano Basagni, "Range Extension of Passive Wake-up Radio Systems through Energy Harvesting," in Proceedings of IEEE ICC, 2013.

He Ba, Wendi Heinzelman, Charles-Antoine Janssen and Jiye Shi, "Mobile Computing - A Green Computing Resource," in Proceedings of IEEE Wireless Communications and Networking Conference, 2013.

Tolga Soyata, He Ba, Wendi Heinzelman, Minseok Kwon and Jiye Shi, "Cloudlets: Extending the Utility of Mobile Computing," in Communication Infrastructures for Cloud Computing: Design and Applications, IGI Global, 2013.

Rajani Muraleedharan, Ilker Demirkol, Ou Yang, He Ba, Surjya Ray, Wendi Heinzelman, "Sleeping Techniques for Reducing Energy Dissipation," in The Art of Wireless Sensor Networks, Springer Berlin Heidelberg, 2014.

Na Yang, He Ba, Weiyang Cai, Ilker Demirkol, Wendi Heinzelman, "BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2014.

Minseok Kwon, Zuochao Dou, Wendi Heinzelman, Tolga Soyata, He Ba, Jiye Shi, "Use of Network Latency Profiling and Redundancy for Cloud Server Selection," in IEEE International Conference on Cloud Computing, 2014.

Colin Funai, Cristiano Tapparello, He Ba, Bora Karaoglu, and Wendi Heinzelman, "Extending Volunteer Computing through Mobile Ad Hoc Networking," in IEEE Global Telecommunications Conference, 2014.

Cristiano Tapparello, Colin Funai, Shurouq Hijazi, Abner Aquino, Bora Karaouglu, He Ba, Jiye Shi, and Wendi Heinzelman, "Volunteer Computing on Mobile Devices: State of the Art and Future Research Directions," in Enabling Real-Time Mobile Cloud Computing through Emerging Technologies, IGI Global, 2015

Acknowledgements

Throughout my study at the University of Rochester, I have received infinite help and support from a number of people, without whom this dissertation would not be completed.

First and foremost, I would like to express my deepest appreciation to my advisor, Prof. Wendi Heinzelman. Her knowledge, wisdom, insight, patience and working attitude will not only lead me towards my doctoral degree but will also benefit me beyond my graduation.

My gratitude also goes to Prof. Mark Bocko, Prof. Zhiyao Duan and Prof. Kai Shen for serving as my thesis committee members and providing invaluable comments and suggestions to my research and thesis.

I would like to express my thanks to Dr. Jiye Shi, for offering me the internship opportunity at UCB Pharma and for providing insightful guidance and support during our collaboration since 2012.

Over the years, I have had the pleasure to work with many exceptional faculty members and researchers. Among them, I would like to specially thank Prof. Melissa Sturge-Apple and Prof. Zeljko Ignjatovic from Project CONNECT, Dr. Ilker Demirkol from Project GENIUS, Prof. Tolga Soyata, Prof. Minseok Kwon and Dr. Rajani Muraleedharan from Project MOCHA, and Dr. Cristiano Tapparello from Project GEMCloud.

I am grateful to all my brilliant colleagues in the Wireless Communications and Networking Group. Specifically, I would like to acknowledge Ou Yang, Tianqi Wang, Chen-Hsiang Feng, Surija Ray, Bora Karaoglu, Li Chen, Na Yang and Colin Funai for their inspirations and collaborations. They and many other friends have

made my life in Rochester bright, colorful and filled with memories that I will always cherish.

Last but not least, I would like to give my special thanks to my dear parents Hong Ba and Xiang Shen. Without their love, encouragement and support, I would not be able to come to the US and achieve my goals.

Abstract

Wireless electronic devices are becoming more and more powerful while remaining portable and affordable. Some of the smartphones and tablets on the market today are equipped with multi-core CPUs and GPUs and have comparable computing capabilities to PCs, along with the improved network bandwidth and connectivity provided by cellular networks. Given this state of current technology, in this dissertation we develop a variety of techniques to enable energy efficient sensing and computing systems. As an example of such a system, consider a personal healthcare system, where wireless sensors are used to gather physiological data and send the data to a local cloudlet. The local cloudlet preprocesses the data and transmits the data to a remote cloud server for computation and storage purposes. Data analysis results are sent back to the cloudlet or directly to the user's smartphone for display.

One of the challenges in developing such a system is the data acquisition and processing. For applications like emotion classification or personal health, sensed data are not always gathered in clean environments and therefore are often corrupted by noise. Noisy sensing signals must be processed to improve the signal-to-noise ratio (SNR) in order to extract relevant information. For example, in an emotion classification application, speech data may contain babble noise from people talking in the background. In order to extract pitch, which is a key feature in emotion classification algorithms, from noisy speech data, we developed a hybrid pitch detection algorithm named BaNa. The BaNa algorithm combines the idea of using the ratios of harmonic frequencies and the Cepstrum approach to find the pitch from a noisy signal. We tested our BaNa algorithm on real human speech samples corrupted by various types of realistic noise. Evaluation results show the

high noise resiliency of BaNa compared to other state-of-the-art pitch detection algorithms.

The second challenge comes from considering the energy availability. Wireless sensors are usually battery powered and hence have limited lifetime. To extend the battery life of a sensor node, power management approaches have to be utilized. The energy of a node can be saved by putting its radio and other components into sleep mode occasionally. To wake up a sensor node so that it can perform its functionalities, traditionally, a duty cycling approach is used, where an internal timer fires to wake up the sensor node from the sleep state. In this case, the sensor's energy efficiency may suffer from idle listening since it has no knowledge of the channel while sleeping. We created a passive wake-up radio sensor node named WISP-Mote by using a programmable RFID tag as an external wake-up radio for a Tmote Sky sensor node. The wake-up radio reduces the energy wasted on idle listening and hence improves the energy efficiency of the sensor node. We characterized the WISP-Mote's performance by measuring its energy consumption for different operations and assessing its wake-up probabilities in different environments for various WISP-Mote to reader distances. MATLAB simulation results show that the energy efficiency of a sensor network using the WISP-Motes is much greater than when using traditional duty-cycling nodes.

Computation of energy efficient sensing and computing systems can be local on the node, or, for more intense applications, computing can be off-loaded to external computing resources, such as cloud-based resources, to save the energy of the node. However, a traditional cloud is composed of powerful but energy-hungry workstations. The growth of the population of mobile devices such as smartphones and tablets provides a huge amount of idle computing power. We describe the design and implementation of a mobile computing system prototype named GEMCloud that utilizes energy efficient mobile devices (e.g., smartphones and tablets) as computing resources. The computing power and energy efficiency

of the mobile devices are evaluated through comprehensive experiments. The results show that a cloud computing system with enough mobile devices working cooperatively is able to save 55% to 98% of the energy consumption of conventional server-based clouds while providing comparable computing speed.

By addressing the challenges of data processing, energy efficient operation and computation off-loading, we have provided the next step forward for energy efficient sensing and computing systems.

Contributors and Funding Sources

This work was supported by a dissertation committee consisting of Professors Wendi Heinzelman (advisor), Mark Bocko and Zhiyao Duan of the Department of Electrical and Computer Engineering, Professor Kai Shen of the Department of Computer Science and Dr. Jiye Shi from UCB Pharma. The following chapters of this dissertation were jointly produced, and were funded by multiple sources. My participation and contributions to the research as well as funding sources are as follows.

I am the primary author of Chapter 2 with the exception of Section 2.4, which was based on collaborations with Luis Soto and Jeff Parvin. I collaborated with Dr. Ilker Demirkol and Dr. Wendi Heinzelman for this work. The work described in this chapter was published in Elsevier Ad Hoc Networks in 2013. Part of this work was published in the IEEE Global Telecommunications Conference in 2010. Section 2.4 appeared as a chapter in Wirelessly Powered Sensor Networks and Computational RFID, Springer Publishers, 2012. This work was funded in part by NSF CNS-1143662 and was supported in part by a generous donation of equipment from Intel Research.

I am the primary author of Chapter 3. The research work described in Chapter 3 was based on my collaborations with Na Yang, Weiyang Cai, Dr. Ilker Demirkol and Dr. Wendi Heinzelman. The tests of the BaNa algorithm on Android were based on an app implemented by Thomás Horta. The work described in this chapter was published in IEEE Transaction on Audio, Speech and Language Processing journal in 2014. Part of this work was published in IEEE Statistical

Signal Processing Workshop in 2012. This work was supported by the Eunice Kennedy Shriver National Institute of Child Health and Human Development Grant R01HD060789.

Chapter 4 was based on my collaborations with Dr. Tolga Soyata, Dr. Wendi Heinzelman, Dr. Minseok Kwon and Dr. Jiye Shi. I am the primary author of this chapter except the work described in Section 4.1 and Section 4.2. This work appeared as a chapter in the book Communication Infrastructures for Cloud Computing: Design and Applications, IGI Global, 2013. This work was funded in part by UCB Pharma, and by CEIS, an Empire State Development-designated Center for Advanced Technology.

I am the primary author of Chapter 5 and Chapter 6. I collaborated with Dr. Wendi Heinzelman, Charles-Antoine Janssen and Dr. Jiye Shi. Part of this work was published in IEEE Wireless Communications and Networking Conference in 2013. This work was funded in part by UCB Pharma, and by CEIS, an Empire State Development-designated Center for Advanced Technology.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 The Growth of Personal Mobile Devices

With recent technology advancements, nowadays wireless electronic devices have become more and more powerful, while retaining the important features of portability and affordability. The smartphone is a perfect example. The 2010 released Motorola Backflip Android smartphone has only a 528 MHz ARM 11 single-core CPU [10], while the 2011 released Samsung Galaxy Nexus has a 1.2 GHz TI OMAP4460 dual-core CPU [11], a flagship at the time. By the end of 2012, a smartphone is able to be equipped with a 1.7 GHz quad-core CPU as the one on the HTC One X Plus [12]. The screen size has increased from the 3.1 inch with 320 x 480 pixels of the 2010 Motorola Backflip to the 4.7 inch with 720 x 1280 pixels of the 2012 HTC One X Plus, providing a much better visual experience, while the weight of the smartphone remains almost the same (133 g vs. 135 g). With all of these improvements, the price of a smartphone remains affordable, normally ranging from about $300 to about $600 without a carrier contract or $0-$200 with a 2-year contract with the carrier.

At the same time, the cellular network (infrastructure) is evolving as well. By the end of 2012, according to Verizon Wireless [13], Verizon's 4G LTE had covered

470 cities and nearly 80% of the population in the US. AT&T [14] has also built up a large 4G LTE network, covering 134 cities and over 285 million people. Along with other carriers, the 4G network has already had a wide coverage and is expanding fast. Although the network bandwidths offered by different carriers vary, in general, the current 4G network provides a downloading speed of at least around 5-10 Mbps [15]. The increasing network bandwidth and extensive network coverage provide the basis for a variety of potential applications.

The powerful computation capability, improved user experience, increased portability and affordability, and better network bandwidth and connectivity all have helped portable computing devices such as smartphones and tablets to become increasingly popular. The popularity of these devices results in an enormous amount of total computing power that could be potentially utilized. These portable wireless computing devices, along with the existing wireless-connected PCs and laptops, form a ubiquitous computing environment that could be harnessed to provide an unlimited amount of computing power to the user.

1.2 The Envisioning of Energy Efficient Sensing and Computing Systems

The powerful computing speed, fast connectivity and sensing capability possessed by smartphones and other personal wireless electronic devices allow us to envision a connected sensing and computing system. Imagine the following scenario in the near future:

Tyrion is a diabetic patient who requires an insulin injection at certain times throughout the day. He wears a wireless glucose monitor on his wrist, which monitors and reports his blood glucose level to his smartphone in real time. The most recent glucose level data received by the smartphone are stored locally on

the smartphone. The historical glucose level data are transmitted to and stored in the cloud for future analytic usage by Tyrion's doctor in another city. Previously, Tyrion had to inject insulin once every several hours. To prevent himself from forgetting the injections, Tyrion had to set up a timer to remind himself. Now, thanks to the real-time monitoring, the smartphone automatically analyzes his glucose level and predicts the time when he needs to inject insulin. When that time comes, the smartphone sends a command to an insulin injector carried by Tyrion, which is also wirelessly connected to the system, to inject an accurate amount of insulin.

The above case represents an example of using a smartphone and a wireless sensor as a smart sensing and computing system in personal healthcare. With the development of sensing devices, there are numerous applications such as this that can be created. In the near future, we envision an energy efficient sensing and computing system for home applications, as shown in Figure 1.1.

In this system, sensors to detect a user's daily activities are located in the bed, couch, door and other places. Physiological data are gathered via wearable sensor

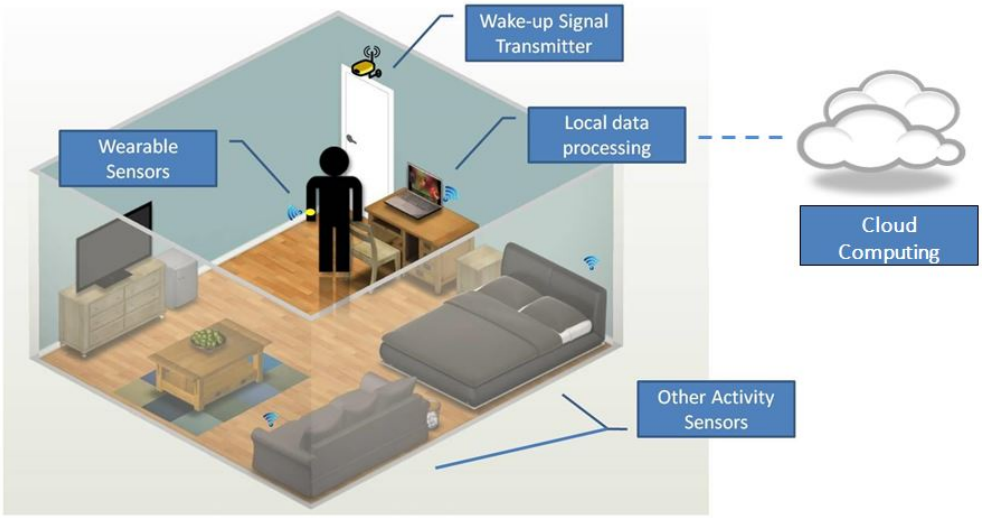


Figure 1.1: Smart Sensing and Computing at Home Illustration.

devices that are equipped on the user's body. Using passive wake-up technology, the sensors do not have to sense or transmit data unless necessary, which helps to preserve the stored energy on the sensors. The sensor data are transferred to a local cloudlet or nearby computing server (e.g., the laptop in Figure 1.1), which provides intermediate processing of the raw data to reduce the data size, improve the signal-to-noise ratio (SNR) and increase the security. The processed data then are transmitted to available cloud resources for computation and storage purposes. The cloud resources may include remote data centers, local computers such as desktops and laptops, and mobile devices such as smartphones and tablets. The decisions on where to send the data for processing may be based on algorithms that consider multiple criteria, such as the energy and time cost for transmitting the data, the energy and time cost for completing the computations, the monetary cost for using these resources and the security level of the place to which the data are being transferred. Using proper algorithms, the system is able to achieve better energy efficiency while not sacrificing much quality of service. After completing the data processing, the results are transmitted back to the cloudlet and displayed at the user's smartphone, tablet, laptop or desktop. As introduced in the above diabetic patient's example, the result may automatically trigger an actuator to work.

We can imagine a large number of such applications that may benefit from this system. Another example is real-time stress and emotion detection. Previously, due to the large amount of data and computations to process the data, stress or emotion detection has been done off-line. With the help of cloud resources from both the local devices and the remote cloud servers, the detection results may get back to the user in real-time, providing the user the possibility to take actions to release the high stress or deal with the negative emotions in time.

1.3 Design and Development Challenges

While the applications described above all seem promising, there are a number of technical challenges that must be addressed before these applications can become reality.

The first challenge relates to data acquisition and processing. Due to the mobility of portable devices, their environments are usually unstable, which results in the difficulties of acquiring clean data. For example, speech data recorded by a microphone may contain background noise from the people chatting next to the mobile phone user [16]; or ECG data from an ECG monitor may be corrupted by the static generated by the movements of body and clothes [17]. Therefore, it is necessary to preprocess the data in order to enhance the quality of the raw data (i.e., signal-to-noise ratio (SNR) of the data). It is also important to seek out approaches or algorithms to process the noisy data and retrieve the relevant information from the noisy data.

Since portable devices are normally powered by batteries and only have limited battery lives, it is crucial to preserve energy [18]. To achieve better energy efficiency, software and hardware approaches can be used. Software approaches include introducing a better power management strategy for the device (e.g., duty-cycling of the sensors and radios equipped on the device, reduction of CPU frequency depending on the computation load, controlling of screen brightness according to the environmental brightness, offloading computations to other devices) and improvement of the transmission protocols of the wireless transceiver (e.g., switching to a more energy efficient transmission channel whenever necessary). Hardware approaches may include the design of more energy efficient circuits (e.g., using more advanced manufacturing process technology such as 22nm technology), and the addition of extra components (e.g., wake-up radio technology for assistance of power management, energy harvester to scavenge energy from the

environment).

1.4 Dissertation Contributions and Organization

In this dissertation, we describe our work developing approaches to support an energy efficient sensing and computing system. More specifically, this dissertation includes approaches to obtain quality data from noisy environments, to save the energy on sensing devices, and to provide energy efficient computing resources. Our primary contributions include:

- Development of WISP-Mote, a passive wake-up radio sensor node. We performed field tests to characterize the wake-up range of the WISP-Mote, and we conducted simulations to compare the performance of the WISP-Mote network to that of a network of sensors employing duty-cycling [18–21]. Two example applications are also provided to show the benefits of using WISP-Mote in specific scenarios.

- Development of a noise resilient pitch detection algorithm named BaNa. We compared the performance of BaNa with other state-of-the-art pitch detection algorithms using real speech data and music data with 8 types of noise from real scenarios. We implement a pitch detector using the BaNa algorithm on the Android platform and discuss the possibility of real-time pitch detection on mobile devices [16, 22].

- Evaluation of the state-of-the-art in the area of mobile cloud computing. We compare different approaches that enhance application performance via cloud-based execution and highlight the research and technological challenges in different approaches [23].

- Development of an energy efficient mobile cloud computing system named GEMCloud that utilizes the computation resource from smartphones and

tablets. We provided energy efficiency comparisons of GEMCloud with traditional cloud servers [24].

- We conduct a public study on mobile volunteer computing using GEMCloud. Data analytics from gathered data shows the feasibility of using mobile devices for volunteer computing. We studied volunteers' behavior based on the device type and their personal preferences and discussed the factors that have impacts on the volunteers' behavior.

The organization of the rest of this dissertation is as follows. In Chapter 2, we introduce the design of a passive wake-up radio sensor node and provide characterizations of the wake-up range, followed by network performance evaluations in multiple scenarios. Chapter 3 introduces a noise resilient pitch detection algorithm. Performance evaluations are provided using real speech data corrupted by 8 different types of additive noise. In Chapter 4, we summarize the challenges of the design and development of mobile cloud computing and provide a survey of the current work on mobile cloud computing. In Chapter 5, we propose an energy efficient mobile cloud computing platform and perform energy efficiency evaluations in comparison with traditional cloud computing platforms. In Chapter 6, we discuss the data collected from a public study using our mobile cloud computing platform. Chapter 7 concludes the dissertation and provides future research directions.

2 Energy Savings for Sensors Using Passive Wake-Up Radios

2.1 Introduction

In our envisioned sensing and computing system, energy efficiency is one of the key focuses. Especially for wireless sensor nodes, due to the portability requirements, usually they are powered by batteries and thus have very limited lifetime if no power management is performed. In some extreme cases such as implanted sensors, it is impossible to exchange the batteries frequently. In other wireless sensor applications, energy efficiency of the sensors is also crucial due to the cost of labor for exchanging batteries.

When looking into the energy consumption of a sensor node, radio transmission and reception are the two major sources of energy drain. When a node is active and waiting to receive data, it wastes energy on idle listening. Since traffic loads are usually low in WSNs, such idle listening can waste enormous amounts of energy unless efficient communication mechanisms are employed.

To extend the lifetime of a sensor node, we can turn off its radio and set its microcontroller (MCU) into a sleep mode when it is idle and wake it up when

there are possible transmissions. To wake up a sensor node, there are generally two approaches: a scheduled approach where a timer is set and the firing of the timer wakes up the node, namely duty cycling, and an on-demand approach where the node is woken up by a radio signal, namely radio wake-up.

In duty cycling, a node, or only part of the node (e.g., its radio component if using technologies such as wake-on-radio [25]) is periodically set into the sleep mode and can manage the trade-off between energy consumption and data latency, by setting the duty cycle value accordingly. With lower duty cycles, nodes will consume less energy at the cost of higher latency for data delivery. Once a node wakes up during the active part of its duty cycle, it must listen to the channel for a period of time to determine whether other nodes or the sink are available for communication. This introduces complexities and adds overhead to the medium access control (MAC) protocol.

Using radio wake-up techniques, such overhead can be reduced. A wake-up signal triggers a node to wake up from the sleep mode and start reception activities. Normally, the wake-up signal is sent or received by a secondary radio transceiver. In order to improve energy efficiency, the energy consumption of this extra wake-up radio tranceiver should be extremely low. The energy benefit of using radio wake-up in comparison with duty-cycling is that nodes do not waste energy on idle listening of the main radio, since they are only awakened by neighboring nodes when there is a request for communication. In addition, using a wake-up signal reduces the overhead in control traffic since a node woken up through a wake-up radio knows that another node is ready to receive data.

Wake-up radio receivers can be categorized as active and passive based on whether the receiver uses a connected power supply. Active wake-up radio receivers require a continuous power supply while passive wake-up radio receivers harvest energy to power themselves from the wake-up radio signal transmitted by the sender. Active wake-up radio receivers have a relatively better sensitivity, in

other words, their wake-up range is relatively longer. On the other hand, passive wake-up receivers operate within a relatively smaller range but do not require any attached power source.

One possibility to achieve a passive wake-up radio is to use a passive radio-frequency identification (RFID) tag as the wake-up signal receiver and an RFID reader as the wake-up radio transmitter, as off-the-shelf RFID tags and readers are readily available. Despite the low cost of this solution, there are two major challenges for implementing this approach in real-world applications: the limited wake-up range compared to the main radio communication range, and the energy cost for the wake-up signal transmitter. The former creates problems in terms of network coverage, while the latter makes it impossible for a battery-powered sensor node to wake up another node, i.e., multi-hop wake-up is unrealistic using this approach. However, these types of devices can be beneficial in scenarios where there is a mobile data sink (i.e., a data MULE [26]) or where the sensors are mobile (e.g., sensors on a person) and come in contact with a fixed sink at some point. For these scenarios, the sink can wake up the sensor nodes to query them for data whenever they are in the wake-up range.

In this chapter, we describe and characterize a passive RFID-based wake-up radio sensor node that we developed, and we discuss and analyze the use of our passive wake-up radios in various scenarios. Our passive RFID wake-up device, which we call a WISP-Mote, is created by combining a WISP (Wireless Identification and Sensing Platform) [27], an RFID tag developed by Intel Research, and a Tmote Sky [28] sensor node. We characterized the WISP-Mote's performance by measuring its energy consumption for different operations and assessing its wake-up probabilities in different environments for various WISP-Mote to reader distances. Finally, we performed MATLAB simulations to show the benefits of using WISP-Motes by comparing the performance of a sensor network with WISP-Motes with a standard duty-cycling architecture. Our results show that the energy

efficiency using the WISP-Motes is much greater than when using duty-cycling, without any loss in performance.

The rest of this chapter is organized as follows. In Section 2.2, we review the current state of the art in radio wake-up technology. Section 2.3 introduces our RFID wake-up sensor devices, including the hardware implementation and the energy consumption measurements. Field tests to determine the wake-up probability in relation to the distance from the RFID reader are presented in Section 2.4, followed by simulation results and performance analyses when using the WISP-Motes in different network scenarios in Section 2.5. As applications are the motivators for research in wireless sensor networks, Section 2.6 describes various potential applications of our passive wake-up radios and shows the benefits of these wake-up radios through simulations. Finally, Section 2.7 concludes the chapter.

2.2 State of The Art In Radio Wake-up

Remote wake-up is realized via a second receiver triggering the main data receiver when necessary. To gain a benefit in energy efficiency, the extra receiver must be lower power than the main data receiver (or, ideally, require no battery power to operate), because while the main receiver is in the sleep mode, only the wake-up receiver remains on to monitor the wake-up channel continuously or following a duty-cycling scheme.

Wake-up radios can be categorized as active wake-up radios and passive wake-up radios, depending on their energy sources [29]. An active wake-up radio has a better wake-up range than a passive wake-up radio, however, the active wake-up radio requires continuous power supply. On the other hand, a passive wake-up radio operates within a relatively smaller range, but it does not require an external power supply. A passive wake-up receiver harvests energy to power itself from the

wake-up signal transmitted by the sender. In this section, we review the state-of-the-art of both active and passive wake-up radio receivers.

Table2.1 provides a summary of these different wake-up receiver design techniques and a comparison of the energy consumption, sensitivity, and implementation status of these wake-up radio receivers.

2.2.1 Active Wake-up Radio Receivers

Several different low-power active wake-up receivers have been proposed [30–36]. In [30], Otis et al. propose the use of a super-regenerative architecture with a $1.9\,GHz$ bulk acoustic wave (BAW) resonator to reduce the power consumption of the wake-up radio. The power consumption of this radio is $400\,\mu W$ for the receiver and $1.6\,mW$ for the transmitter. This approach is further optimized to create a $65\,\mu W$ wake-up receiver [31], using a $1.9\,GHz$ BAW resonator matching network for RF signal filtering. This wake-up receiver can provide a sensitivity of $-50\,dBm$ at $40\,kbps$ and $-48\,dBm$ at a maximum data rate of $100\,kbps$.

A different approach is developed by Le-Huy and Roy [32] and Von der Mark et al. [33], where zero-bias Schottky diodes are used because they have no bias current through the diodes. The low-power $2.4\,GHz$ wake-up receiver proposed in [32] is designed to work with a directional antenna and pulse width modulation in order to reduce energy dissipation. Simulation results show that the receiver can reach $-50\,dBm$ sensitivity with only $19\,\mu W$ power consumption. A three stage wake-up scheme is introduced in [33]. In this approach, a very low power (on the order of nW) always-on stage is used to trigger an intermediate higher power (on the order of μW) stage for wake-up signal verification. Only if the wake-up signal is confirmed is the main transceiver activated.

Other approaches for active wake-up radios are described in [34] and [35]. Junaid et al. propose a wake-up receiver including a five stage charge pump used

to increase the received signal voltage [34]. The only active parts of the wake-up circuit are the digital comparator and the voltage divider, which consume $350\,nA$ and $526\,nA$, respectively. In [35], Van der Doorn et al. implement a wake-up receiver using only commercial components to reduce the extra hardware costs. Their design consumes 171 μW with $-51\,dBm$ sensitivity. Although there are several hardware proposals for active wake-up radios, not many physical implementations or commercialized products are available today. Recently, Austria Microsystems announced their latest 3-channel low frequency wake-up receiver working at $15 - 150\,kHz$ [36]. This product consumes $8.1\,\mu W$ and can reach a sensitivity of about $-37\,dBm$ (as calculated from the provided specifications).

2.2.2 Passive Wake-up Radio Receivers

Compared with active wake-up receivers, passive wake-up receivers do not require energy from a physically connected power supply; instead, they harvest energy from the transmitted wake-up signal. While this makes passive wake-up radios energy efficient, the wake-up range for passive wake-up radios is relatively shorter, i.e., the receivers' sensitivity is lower. Currently, there are limited studies on passive radio wake-up receivers. Gu et al. propose a passive radio wake-up circuit that theoretically could operate at a range of $10\,ft$ with $5\,ms$ latency, according to SPICE simulation results [37]. If a comparator and an amplifier are added, which respectively consume negligible currents of $350\,nA$ and $880\,nA$, the radio could theoretically reach up to $100\,ft$ with $55\,ms$ latency.

A performance study on the use of passive RFID wake-up radios is given by Jurdak et al. [38, 39]. In their work, an RFID wake-up mechanism is proposed, namely RFIDImpulse, which assumes a commercial RFID reader and a passive RFID tag are attached to each sensor node, providing radio wake-up capability. The performance of the proposed mechanism is investigated through MATLAB simulations and compared with the BMAC protocol [40] and the IEEE

802.15.4 standard [41]. Their results show that RFIDImpulse outperforms both other methods in terms of energy efficiency and transmission rate for low and medium traffic scenarios. However, the analysis is based on an important assumption that all nodes have the capability to wake up their neighbors, which is not feasible currently, due to the considerable amount of energy required by the RFID reader, which sends out the wake-up signal, and its large size. In addition, their energy consumption analysis does not include the energy consumed by the nodes to wake up. In reality, the wake-up energy consumption includes the energy used for MCU boot-up and for radio initiation, which could be comparable to the energy consumed for radio transmission. Furthermore, the implementation described in [38, 39] uses a coil instead of an RFID tag for the purpose of proof of concept.

In this chapter, we introduce a passive RFID-based wake-up device named WISP-Mote and characterize its power consumptions during different operation modes, including waking up. To the best of our knowledge, the WISP-Mote is the first reported complete implementation of a passive radio wake-up device that provides both broadcast-based wake-up and ID-based wake-up. The provided characterization data are measured using our implemented WISP-Mote in various environments.

To have wide network coverage, WISP-Motes are used with mobile sinks in this chapter due to their short wake-up ranges. The mobile sink wakes up the WISP-Motes when it gets within their wake-up range to collect their data. This is similar to the three-tier layered architecture described in [26]. In our scenarios, we investigate random walk along with two other mobility models and present their performance comparison.

Table 2.1: Wake-up Receiver (WuR) Comparisons

Active WuRs	Main techniques	Frequency	Power Consumption	Sensitivity	Implementation
B. Otis et al. [2]	Super-regenerative with BAW matching network	1.9 GHz	400 W	-100.5 dBm	Yes
N. Pletcher et al. [3]	BAW matching network	1.9 GHz	65 W	-50 dBm	Yes
P. Le-Huy et al. [4]	Zero-bias Schottky voltage doubler	2.4 GHz	19 W	-50 dBm	No
S. Von der Mark et.al [5]	Three stage wake-up, Zero-bias Schottky diode used at 1st stage	N/A	nWs to μ Ws	N/A	No
J. Ansari et al. [6]	Five stage charge pump	N/A	2.628 W	N/A	Yes
B. Van der Doorn et al. [7]	Commercial filter and amplifier	868 MHz	171 W	-51 dBm	Yes
Austria Microsystems [8]	N/A	15-150 kHz	8.1 W	-37 dBm	Yes
Passive WuRs	**Main techniques**	**Frequency**	**Power Consumption**	**Sensitivity**	**Implementation**
L. Gu et al. [9]	Zero-bias Schottky diode	433 MHz	0 or 3.69 W	N/A	No
H. Ba et al. [23]	Passive RFID (WISP)	902-928 MHz	0	-10 dBm	Yes

N/A: Data not available

2.3 RFID Wake-up Sensor Device

In this section, we describe the implementation of our WISP-Mote and provide measurements of its energy consumption in different modes.

2.3.1 Design and Implementation of the WISP-Mote

In our previous work [18], we introduced our hybrid sensor device with RFID wake-up receiver, namely the WISP-Mote, as shown in Fig. 2.1. We employ an Intel WISP (Wireless Identification and Sensing Platform) as an external wake-up signal receiver for a Tmote Sky mote. A WISP is an RFID tag with sensing and computing capabilities developed by Intel research. Using energy harvesting, it can be powered wirelessly by a UHF RFID reader. In our implementation, we use a UHF Gen2 Speedway RFID reader from Impinj [42].

The RFID reader sends a continuous wave along with commands according to the standard C1G2 protocol [43] to the tag. The tag sends data back by modulating the reflection coefficient of the backscattered signal (via changing the antenna impedance) [44]. The tag also collects the energy from the received signal and stores it in a capacitor as its power supply. Therefore, using a WISP as a

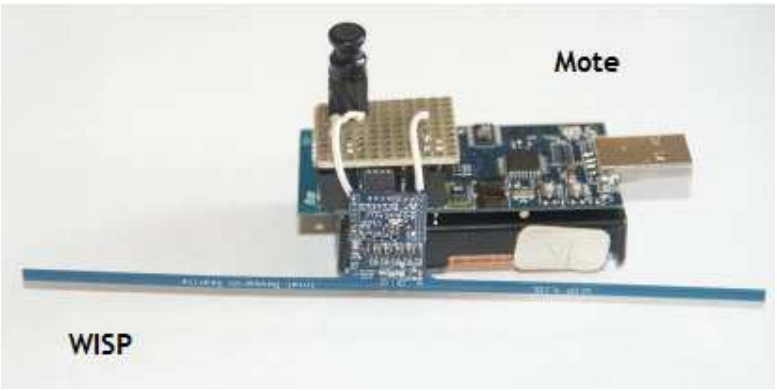


Figure 2.1: A WISP-Mote.

wake-up receiver does not consume extra power from the sensor node. The Tmote Sky mote [45] is a battery-powered wireless sensor node developed by UC Berkeley. By using the WISP to generate an interrupt signal to the Tmote Sky, a passive wake-up sensor node is created.

According to the destination of a wake-up signal, radio wake-up can be categorized as broadcast-based wake-up and ID-based wake-up. Using a broadcast-based wake-up radio, all sensors within range of the transmitted wake-up signal will be woken up, while an ID-based wake-up radio transmits a wake-up signal that contains the intended destination's address and thus only wakes up the node with a matching address. Using the WISP, we can implement both broadcast-based wake-up and ID-based wake-up.

For broadcast-based wake-up, whenever the WISP harvests enough energy from the reader's radio, it sends a pulse to wake up the mote from the sleep state. Thus, any WISP-Mote within the wake-up range of the reader will be awakened. Since in our RFID wake-up scheme there is no communication from the WISP to the reader, we programmed the WISP not to respond to any command sent by the reader. Therefore, we reduced the energy consumption caused by the MCU and other components on the circuit that typically perform the standard RFID communication protocol (e.g., C1G2 protocol). To reduce this energy consumption, we disable all functions of the C1G2 protocol on the WISP that are not required for the wake-up of the sensor node. Thus the only responsibility of the WISP for broadcast-based wake-up is to harvest energy from the reader and send an interrupt signal to the mote. By minimizing the energy required for the WISP, we maximize the wake-up range of the WISP-Mote.

In a dense network, this broadcast-based wake-up may bring a large number of collisions on the data channel, since all the nodes within the wake-up range will be awakened by the reader, and none of them is aware of the other nodes awakened at the same time. To reduce unnecessary wake-ups and to reduce collisions caused

by these unnecessary wake-ups, we need the ability to wake up a certain node or a specified class of nodes. With this intention, we programmed the WISP to let it generate a trigger pulse only after receiving a packet that has its ID or Class number in the packet header. This functionality requires the demodulator to work and additional computation by the WISP's MCU, both of which consume extra energy from the WISP. Therefore, for any given distance, the wake-up probability of the ID-based wake-up is expected to be smaller than or equal to the wake-up probability for the broadcast-based wake-up. For a wake-up receiver with addressing capability, broadcast-based wake-up can be achieved by assigning a particular ID as a broadcast address.

To determine the performance of our WISP-Mote, we performed field tests of both broadcast-based and ID-based wake-up in various environments. Details of this characterization are discussed in Section 2.4. For both broadcast-based and ID-based wake-up, it is preferred that the nodes not be awakened by the sink when not necessary, i.e., when the nodes do not have any buffered data to send. To achieve this, we programmed the sensor nodes to disable the interrupt functionality of the wake-up signal input port when they have no data to send. Thus the node will wake up only when it has buffered data and it receives a wake-up signal. Otherwise, the node will remain in the low-power sleeping state. This reduces unnecessary energy waste.

2.3.2 Energy Consumption Measurements

Unlike active wake-up radios, which constantly consume energy, RFID wake-up radios do not consume any energy from the sensor node. This further enhances its energy efficiency. The Tmote Sky datasheet [45] provides the current consumptions in typical operating conditions, but lacks information about energy consumed when the node is waking up, which is essential for the energy consumption analysis of RFID-based wake-up sensor networks. In addition to current

Table 2.2: Power Consumption Measurements of a T-mote Sky Node

Operation	Average current consumption	Duration
Wake-up	$10.4\,mA$	$5\,ms$
Transmit 12 byte packet	$18.2\,mA$	$30\,ms$
Receive and idle listening	$20.2\,mA$	
Sleep	$0.2\,mA$	

consumptions in transmitting and receiving, we measured the current and time consumed in booting and radio initiation, i.e., when the node is waking up from the sleep state. The results are shown in Table 2.2. Our measurements are consistent with those from the Tmote Sky datasheet. The results show that besides radio transmission and reception, a node's wake-up also consumes energy that cannot be ignored. These measurements are used in the energy analysis for our sensor network.

2.3.3 Sensor Network System

Besides the WISP-Motes, in some applications, the sensor network may also include one or more base stations (BSs) and data MULEs (MULEs). MULEs are mobile data collectors that are equipped with both a mote and an RFID reader, and therefore have the capability to wake up the nearby WISP-Motes and to receive data from them. We assume that when the MULEs are close to a BS, they dump all of the collected data to the BS, or they can act as a gateway and send the data immediately e.g., using a cellular network. The MULE must have the ability to move and sufficient energy to power the RFID reader. In a real system, any moving agent, e.g., a vehicle, an animal or a human, that carries an RFID reader and a mote with adequate power supply can perform as a MULE.

This system is designed for a delay-tolerant network with energy constraints on the sensor nodes. The major advantage of this network is high energy efficiency for the sensor nodes. Energy waste is reduced by decreasing the number of unnecessary wake-ups and the time used for sensing the channel. The packet delay in the data MULE architecture is related to both the mobility behavior of the MULEs and the wake-up range of a MULE. Therefore, in order to evaluate the performance of our sensor network, the effective wake-up range of the WISP-Mote is required, which will be addressed in the next section.

In some other applications, data collectors can be set at specific locations and wake up the nodes when the nodes move close to the data collectors. These types of applications are discussed in Section 2.6.

2.4 Characterization of WISP-Mote

Field tests to determine the capabilities of our WISP-Mote are important to guide the design of appropriate protocols and the performance evaluation of the overall system. We perform field tests to characterize the wake-up probability of a WISP-Mote as a function of height and distance to the reader.

There are two main factors that have a very large effect on the wake-up probability in a real world implementation. The first is the distance from the reader to the WISP. The energy that a WISP is able to harvest is inversely proportional to the square of the distance due to path loss (assuming a free-space environment). When the harvested power is enough for the WISP to drive its MCU, the WISP will perform its function (either decoding the packet to check for the ID or automatically waking up the mote), otherwise it will remain asleep.

The second factor is the environment where the system is located. The environment plays a critical role because reflections can have large effects on the wake-up probabilities at different areas in the three dimensional space in front of

the reader. The line-of-sight signal and reflections meet, and create constructive areas and destructive areas depending on their phase differences. In certain locations, there are destructive effects, causing dead zones where the node is unable to be awakened. There will also be constructive effects resulting in locations with much higher wake-up probabilities than the locations that are closer to the reader.

Because the ID-based wake-up scheme has higher energy consumption than the broadcast-based one, theoretically it should have a slightly reduced effective range in all environments. The main goal of this testing was to explore what is the effective range of both schemes, and to provide numerical support for later analyses of the trade-offs between wake-up radio based networks and duty-cycling networks.

This section will cover three different environments with both ID-based and broadcast-based wake-up to demonstrate the effects of distance and environment. We measure the wake-up probability as a function of distance and height to a reader in a hallway and an open-air environment, and the average wake-up probability as a function of distance in an office environment. The measurements are used in the network simulations described in Sections 2.5 and 2.6.

2.4.1 Experiment Setup

We set up our experiments as follows. The reader is raised off of the ground to a height of $84\,cm$ to reduce reflective effects due to the ground close to the reader. The height of $84\,cm$ was chosen purely for convenience in the experimental set-up. Due to the necessity to vary both distance from the reader and height, the WISP-Mote was attached to a tripod, which allows both x-axis movement, towards and away from the reader, and y-axis movement, up and down with respect to the reader.

For the broadcast-based wake-up scheme, the data collector broadcasts a

generic wake-up signal that causes every WISP-Mote receiving this signal to accumulate energy to wake up. After waking up, the WISP-Mote sends a packet to the data collector, which also includes a wake-up count. After sending the packet, the WISP-Mote promptly returns to sleep. In our single WISP-Mote field tests, no ACKs or back-offs are implemented, since the backward link quality is good (mote-to-data collector) compared to the forward link quality (data collector-to-tag) and since there is no contention in the network.

To calculate the wake-up probability, we set up the system as follows. Since we cannot control the commercial RFID reader to send a wake-up signal at particular times, instead we have the RFID reader send a continuous wave. To count the number of times that a WISP-Mote is able to be awakened by this continuous wave, we periodically enable the interrupt from the wake-up signal input port once every 0.5 seconds and disable it after the mote is awakened by the WISP. As a result, WISP-Motes can wake up as long as the wake-up receiver is harvesting enough power and the interrupt is enabled. The reader is then run for a fixed amount of time, in this case 100 seconds, and the final wake-up count sent by the WISP-Mote is recorded. We can then calculate the wake-up probability of the investigated scenario by dividing the observed number of wake-ups to the total possible wake-ups, i.e., 200 wake-ups.

The ID-based wake-up experiments were conducted in an identical manner, except that instead of simply broadcasting a generic wake-up signal, the data collector transmits an ID, against which the WISP compares its own ID before deciding whether to wake up the mote or not.

2.4.2 Wake-up Probability

Open Environment

The purpose of the open environment tests was to extract the WISP-Mote wake-up characteristics in an area with very few reflecting surfaces. This environment could be a large indoor area, or an outdoor area, but for testing purposes a large gymnasium was used.

Fig. 2.2 shows the test results for the broadcast-based wake-up in an open environment . The y-axis of the graph shows the vertical distance between the WISP-Mote and the data collector, and the x-axis shows the horizontal distance between the WISP-Mote and the data collector. The data collector is located at point (0, 0) on the graph. The colors at each point represent the wake-up probability at that particular point. As seen in Fig. 2.2, the WISP-Mote has almost 100 % wake-up probability for all points within $2.5\,m$, and after that point, reflections from the ground start to have a significant effect on the wake-up probability. These reflections appear to match the two-ray ground model, and there are clearly areas where destructive interference creates dead zones, as well as areas where constructive interference enables 100 % wake-up probability far from the data collector.

Fig. 2.3 shows the test results for the ID-based wake-up in the open environment. As expected, there has been a slight decrease of the wake-up range from the $3\,m$ of the broadcast-based scheme down to $2.5\,m$ for the ID-based one at a height of $0.2\,m$ above the data collector. However, generally the ID-based scheme has a similar performance to the broadcast-based scheme in this environment. As in the broadcast results, the effect of the wake-up signal reflected from the ground is also evident in these results.

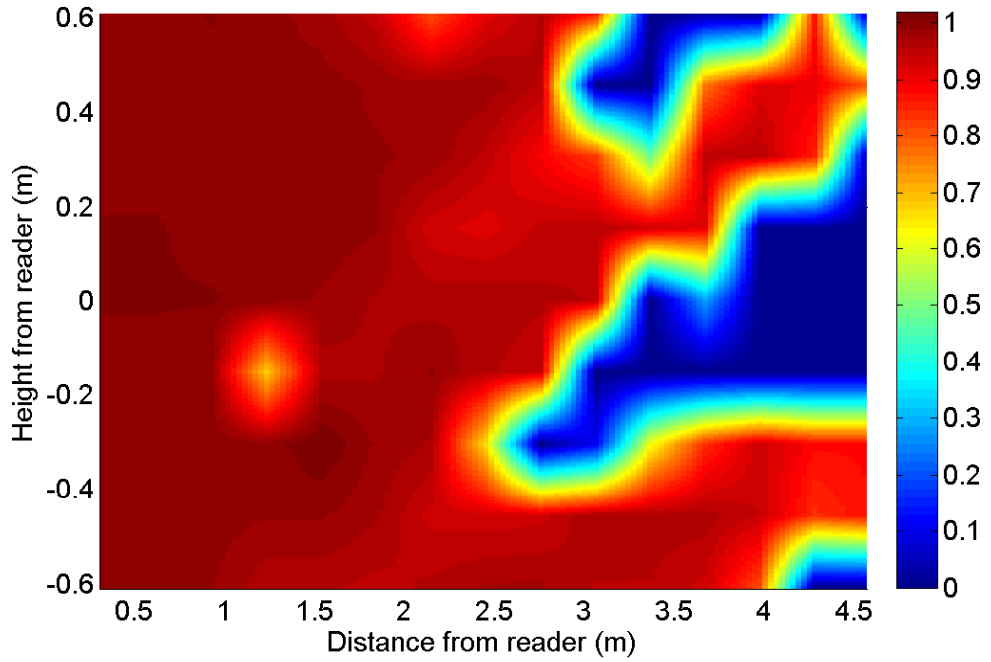


Figure 2.2: Broadcast-based wake-up probabilities in an open environment.

Closed Environment

When such pronounced reflective effects are witnessed in an environment with only one reflective surface, it can be hypothesized that even more significant and far more unpredictable reflections will occur in a closed environment. The next set of tests took place in a long hallway.

As can be seen in Figs. 2.4 and 2.5, a closed environment vastly increases the amount of interference among the multiple copies of the received signal. For the broadcast-based wake-up, 100 % wake-up only occurs at any height for only very small distances from the reader, and for ID-based wake-up, even at close distances, the impact of reflections is obvious.

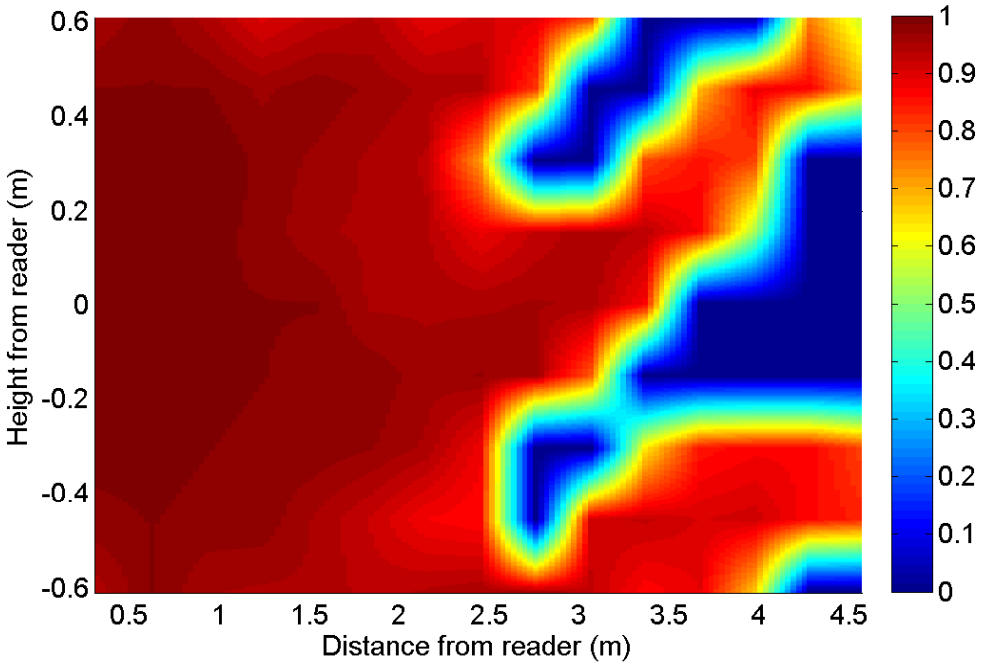


Figure 2.3: ID-based wake-up probabilities in an open environment.

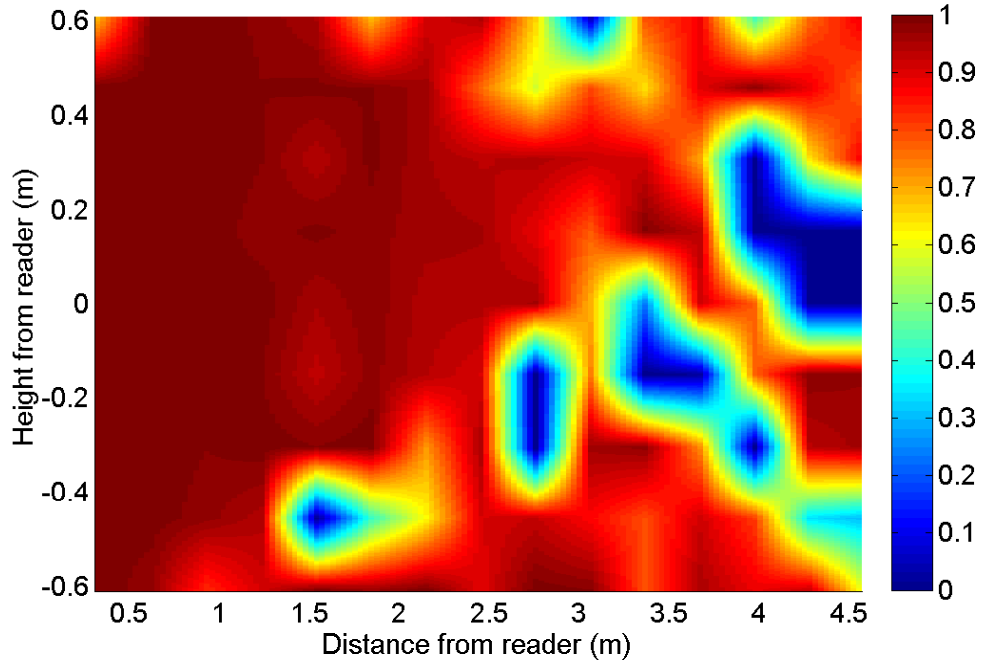


Figure 2.4: Broadcast-based wake-up probabilities in a closed environment.

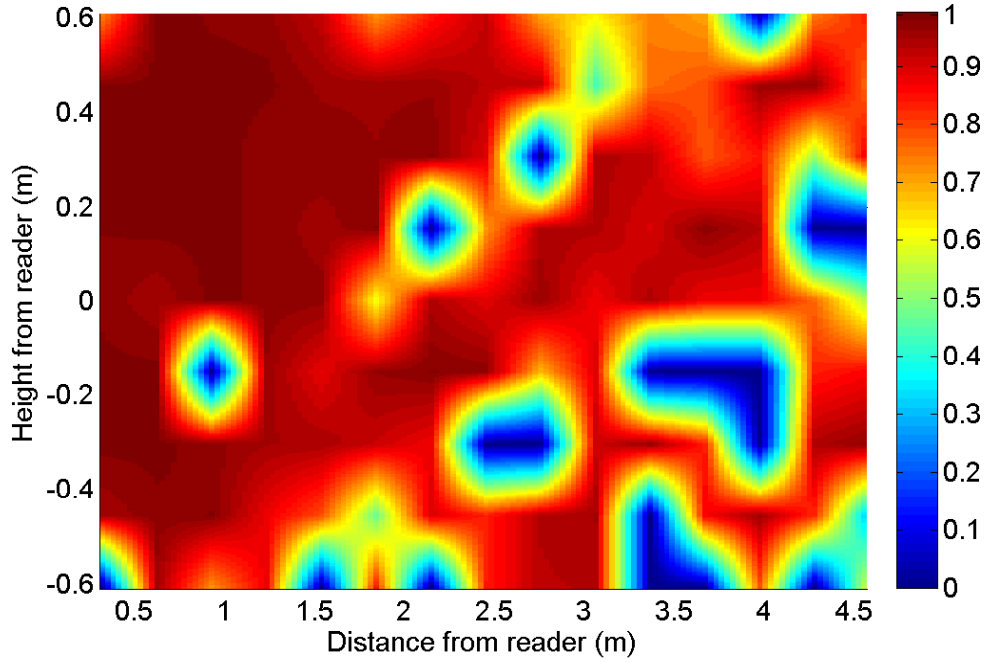


Figure 2.5: ID-based wake-up probabilities in a closed environment.

Cluttered Environment

In real-life applications, it is more likely that the sensor network will be deployed in a cluttered environment such as an office, a warehouse or a store. For the cluttered environment, it was infeasible to set up the in depth systematic tests as for the previous two environments, due to physical obstacles. Instead, the average wake-up probability was tested at different distances from the data collector. For each distance on the plots, ten measurements were taken in various spots within an office and then the average value is plotted.

Although not shown in the same detail as the previous graphs, Fig. 2.6 does illustrate that even more interference is occurring in the cluttered environment than in the previous two environments for both broadcast-based and ID-based wake-up. No wake-up was recorded at or outside of the 3 meter mark, anywhere in the room. As the data show, the WISP-Mote has a relatively stable wake-up

range in an open environment, a volatile but similar range wake-up in a closed environment, and a relatively shorter wake-up range in a cluttered environment.

2.4.3 Broadcast-based vs. ID-based Wake-up

There are quite a few meaningful conclusions we can draw from this data about the differences between ID-based and broadcast-based wake-up, and also the effectiveness of the system as a whole. In every environment, comparing the average wake-up probabilities as a function of the distance from the data collector, we can see that the broadcast-based scheme results in slightly higher average wake-up probabilities. While the ID-based wake-up scheme sacrifices performance by a very small amount in terms of wake-up probability, it has the capability of waking up a particular class of nodes or an individual node.

If there are only specific nodes that need to be woken up, then ID-based wake-up can lead to significant energy savings, as it does not force all nodes to wake up and contend for the medium. In a scenario where all nodes serve the same purpose, then the usefulness of having ID-based wake-up may be reduced, but in the case where different nodes have different sensors and information, it may

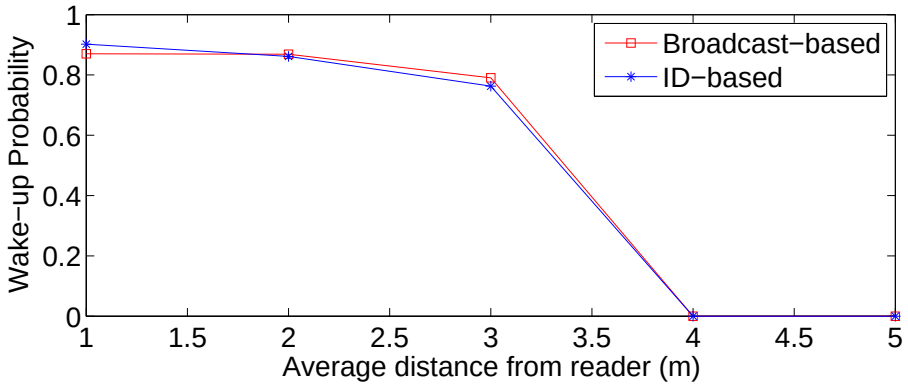


Figure 2.6: Broadcast-based wake-up and ID-based wake-up probabilities in a clustered environment.

lead to very large savings in energy. In most scenarios, the very small sacrifice in wake-up range is less crucial than the increased functionality and possible energy savings. Detailed simulation-based performance analysis of the wake-up radio schemes will be provided in Sections 2.5 and 2.6.

To implement the ID-based wake-up, in some scenarios, the data collector has to know the node's ID so that it can wake up a specific sensor. One approach to enable the data collector to determine the nodes' IDs is to set a broadcast ID that can let the data collector wake up any node in the network. The data collector first uses the broadcast ID to wake up all the nodes in the nearby area and obtain their IDs. Then the data collector uses these node IDs to wake up each individual node one by one to acquire data from them if needed. Another approach that can be used in other scenarios, such as a health monitoring scenario, is to assign a group ID to classify various types of nodes (e.g., all heart monitor sensors are assigned an ID of 1, all blood oxygen sensors are assigned an ID of 2, etc.). The data collector only needs to know the group ID and is able to wake up the interested group of nodes on demand (e.g., any heart monitor in the area).

2.5 Simulations

As we introduced previously, the two challenges in the design of passive wake-up radio systems that limit their use in existing sensor network applications are the large energy cost of the wake-up transmitter and the short wake-up range. The power amplifier needed to send the wake-up signal requires a high power budget. However, most sensor network applications consist of low-power (usually battery-powered) sensor nodes and are expected to run for years. Notwithstanding the cost (which is also important in real systems), the huge power drain renders it hard to equip every node with a wake-up signal transmitter, e.g., in our system, the RFID reader. On the other hand, the limited wake-up range shortens the one-

hop range. Therefore, currently, to achieve large network coverage, mobility is a necessity. Scenarios where the data collector moves to the sensors or the sensors move to the data collector are possible applications for our radio wake-up system. In the simulations in this section, we consider a data MULE scenario to compare the performance between our WISP-Mote RFID wake-up scheme and standard duty-cycling.

2.5.1 Simulation Setup

We use MATLAB to simulate the performance of the WISP-Mote network and several duty-cycling networks (with duty cycles between 10 % and 0.1 %). In our simulations, we deploy static nodes according to a uniformly random distribution in a $200\,m \times 200\,m$ network, and we simulate 1 hour of the network activities. We measure the latency, undelivered packet ratio, number of collisions, and energy as our performance metrics. For each option (WISP-Motes and duty-cycling), we run 20 simulations, where each simulation has a different random network deployment, and we plot the average value of each of the performance metrics.

We compare two sensor network systems, one with standard sensor nodes that employs duty-cycling and the other with the WISP-Motes that employs radio wake-up. Both systems have Data MULEs, i.e., mobile data sinks with a mote, an RFID reader (in the WISP-Mote network), and a very large power source. MULEs start with random locations and move according to a random direction mobility model. Each MULE uniformly randomly selects a speed from $[5\,m/s, 15\,m/s]$ and moves towards a direction chosen uniformly randomly from $[0, 2\,\pi]$. Whenever the MULE reaches the field boundary, it chooses another speed and direction randomly.

In the duty-cycling scenario with standard sensor nodes, each MULE broadcasts a beacon signal, which is actually a short packet, to indicate its presence to

the nodes. On the other hand, in the radio wake-up scenario with WISP-Motes, each MULE sends out a wake-up signal continuously. The sensors will wake up with a probability related to the distance between the sensor and the MULE. The wake-up probability is calculated by the average value at each distance according to our field test results in Section 2.4. We have six wake-up probability models corresponding to broadcast-based wake-up and ID-based wake-up in an open environment, in a closed environment, and in a clustered environment. We consider the broadcast-based wake-up probability model in an open environment in this section to provide general performance comparisons. Other wake-up probability models are used in other example applications in Section 2.6.

We describe the algorithms for both the WISP-Mote scenario and for the duty-cycling scenario in Algorithms 1 and 2. A non-persistent CSMA MAC protocol is used for both scenarios. It is important to note that both scenarios may benefit from more efficient protocols. However, designing a MAC layer protocol is not the focus of this dissertation. We choose a simple wake-up radio MAC protocol and a simple duty-cycling MAC protocol to facilitate the understanding of the benefits of using a wake-up radio in a sensor network.

In the WISP-Mote sensor network, once a sensor node is awakened by the MULE, it performs carrier sensing. If the channel is free, it transmits a packet. If not, it will back off for a random time.

For duty-cycling, a MULE has to announce its presence by sending out a beacon signal. When the MULE is not communicating with any sensor node, it sends a beacon signal once every 8 time slots and listens for responses during the remaining 7 time slots (in our simulations, the transmission of a packet takes 6 slots and an ACK packet takes 1 slot). Once a sensor node is in the active part of its duty cycle, it will first listen to the channel for the MULE's beacon signal for 8 time slots in order to guarantee not missing the beacon signal if a MULE is nearby. If no beacon is received during this period of time, the node will go back

to sleep. To reduce the chance of collisions, if the node receives a beacon signal, it will randomly select a time slot from the following 7 time slots to transmit. However, note that collisions may still occur due to the hidden terminal problem and due to the fact that nodes that are woken up by the same beacon may happen to select the same slot for transmission.

For both scenarios, the node will receive an acknowledgement (ACK) once the packet is received by the MULE. Based on the reception of an ACK, the sensor node can deduce two important facts: 1) the MULE is still close by; and 2) the channel was free of collisions. Therefore, in the next time slot, it is highly probable that the MULE is still within the communication range of the node. Therefore, a node that receives an ACK and has additional packets to send will continue to send packets, once again following the non-persistent CSMA MAC protocol regardless of whether it receives a wake-up signal. After the node starts sending the second continuous packet, it will stop sending and go back to sleep or its duty-cycling schedule when a collision occurs.

For energy efficiency considerations, when a node does not have any buffered data, for the WISP-Mote wake-up scheme we disable the wake-up interrupt, and for the duty-cycling schemes we stop the wake-up timer, as there is no need for the node to wake up. In addition, for both schemes we apply a binary exponential backoff to reduce the chances of collisions when an ACK is not received after sending data.

According to the algorithms, it is obvious that the wake-up radio sensor network does not consume energy on listening to the beacon signal. Therefore, when the MULE is not nearby, the wake-up radio node has zero energy consumption while the duty-cycling node still wakes up periodically to check for a beacon signal. However, we cannot simply assert that the wake-up radio network has better energy efficiency because energy efficiency is also affected by collisions and dropped packets, which are more complicated to analyze. We will discuss this (as well as

other performance metrics) in the rest of this section and 2.6.

To simplify the simulations, we made the following assumptions.

- Propagation delay is ignored.
- Link is ideal (i.e., a packet is correctly received unless a collision happens).
- We assume that MULEs have the ability to communicate directly to the BS and ignore the MULE-BS latency. Therefore, packet delay is counted from the time a packet is generated until the time it is delivered to a MULE.
- We only consider the delay of delivered packets.
- We ignore the energy costs for sensing activities as they will not impact the performance evaluation.
- The mote's communication range is set to $40\,m$ based on experiments described in [46] as well as our own field experiments.

In all simulations (except the limited buffer simulations), we evaluate the network performance in terms of: (1) the average latency per delivered packet, (2) the number of packets that remain in the buffer at the end of the simulation time (undelivered packet ratio, UPR), (3) the average collisions per delivered packet (where collisions are counted whenever two or more nodes try to send packets to the same MULE at the same time), and (4) the average energy consumption for delivering a packet. In the limited buffer simulations, we investigate the average buffer size and the packet delivery ratio (PDR) instead of (2) and (3).

Algorithm 1 WISP-Mote Algorithm

WISP-Mote Algorithm	Energy cost
for node $i = 1$ to N	
if i has buffered data	
if i is awakened by the MULE	E_Wakeup
or i has successfully transmitted a pkt	
sense the channel	E_CS
if the channel is free	
send one packet to the MULE;	E_TxPkt
if ACK received	E_RxACK
keep awake; $\mathrm{C} = 3$;	
end	
elsif channel is busy orACK is not received	
back off for a random number of time slots	
chosen from$[1, 2^C]$; C++;	
end	
end	
end	
end	
$*$C $\in$ [3, 8]	

Algorithm 2 Duty-Cycling Algorithm

Duty Cycling Algorithm	Energy cost
for node $i = 1$ to N	
if i has buffered data	
if i is in the active period	E_Wakeup
or i has successfully transmitted a pkt	
listen for beacon for up to 8 slots	E_Beacon
if beacon signal received	
back off for [1, 7] slots	
sense the channel	E_CS
if the channel is free;	
send one packet to the MULE	E_TxPkt
if ACK received	E_RxACK
keep awake; $\mathrm{C} = 3$;	
end	
elsif channel is busy orACK is not received	
back off for a random number of time slots	
chosen from$[1, 2^C]$; C++;	
end	
end	
end	
end	
$*$C $\in$ [3, 8]	

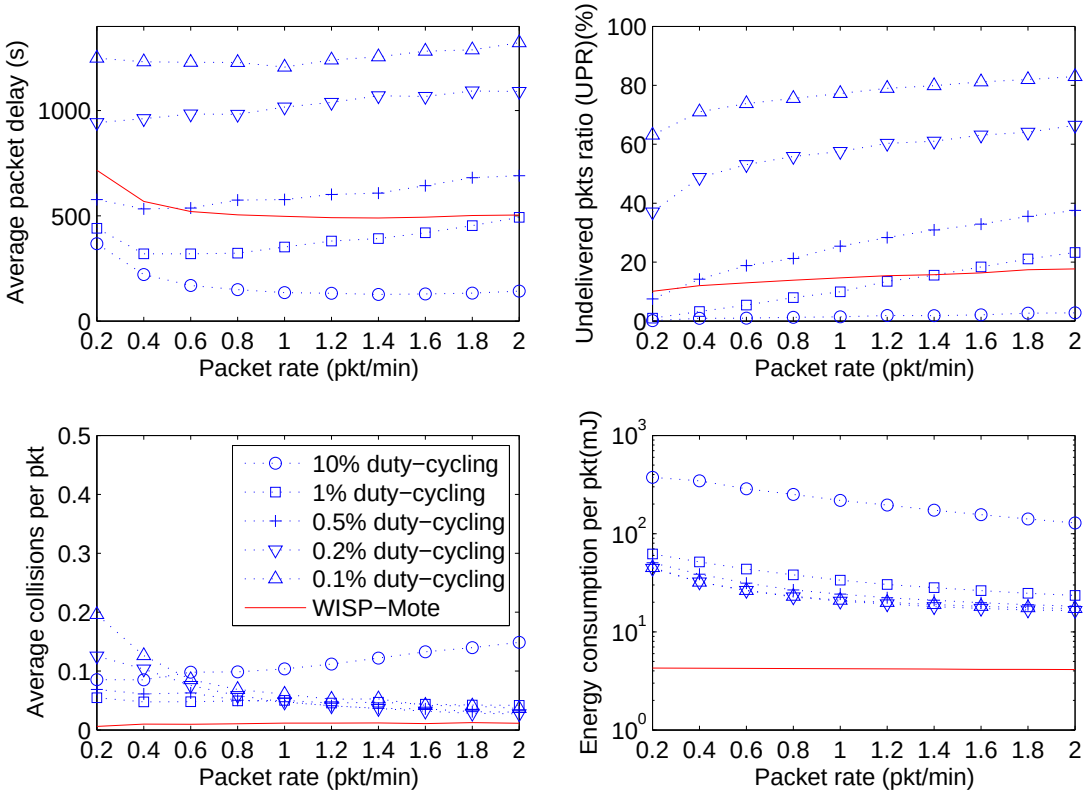


Figure 2.7: WISP-Mote vs. duty-cycling with increasing packet rate (0.002 $nodes/m^2$, 1 data MULE, unlimited buffer).

2.5.2 Performance Results

Effects of Packet Generation Rate

In this set of simulations, we deploy 80 nodes in a $200\,m \times 200\,m$ area, each node has an unlimited buffer size (the effect of limited buffer size is investigated in Section 2.5.2), and we use 1 data MULE. The packet generation rate is varied between $0.2\,pkt/min$ and $2\,pkt/min$. Fig. 2.7 shows the performance comparisons of six sensor network scenarios, five of them employing duty-cycling with different duty cycle values and a radio wake-up scenario with WISP-Motes. The results show that the 10% duty-cycling scheme has the best performance in terms of

packet delay and UPR, while its energy efficiency is very poor compared to the other scenarios. At $0.2\,pkt/min$, 10% duty-cycling consumes about 6 times more energy than the other duty-cycling schemes for delivering 1 packet ($380.8\,mJ$ vs. $62.0\,mJ$) and about 88 times more energy than the WISP-Motes network ($380.8\,mJ$ vs. $4.3\,mJ$). This low energy efficiency is caused by a large amount of idle listening.

From Fig. 2.7, we can see that the latency, UPR and energy performance of both the WISP-Mote network and all duty-cycling networks are relatively stable for all packet generation rates investigated. The sensor nodes with lower duty cycles are not able to deliver packets to the MULE fast enough. Therefore, at the end of simulation, these nodes have more undelivered packets in their buffers, and hence a higher UPR. When the packet generation rate increases, the latency per delivered packet and the UPR only increase slightly. This is because in both algorithms, once a node successfully transmits a packet, the node is able to continuously transmit multiple packets until the MULE is out of the node's communication range or a collision occurs. As the results show, the WISP-Mote node has fewer collisions to successfully transmit a packet. This is because the short wake-up range of the WISP-Mote leads to fewer nodes awake at the same time compared to the duty-cycling network.

The major advantage of the sensor networks employing passive radio wake-up is their energy efficiency. From the energy consumption results, we can see that 10% duty cycling has poor energy efficiency due to regular beacon signal listening, idle listening and collisions. The energy consumption per packet for the WISP-Mote network ($4.3\,mJ$) is about 1/4 of the lowest energy consumption per packet for all the duty-cycling scenarios investigated ($16.4\,mJ$). For two $2800\,mAh$ alkaline batteries, the WISP-Mote can transmit more than 7 million packets, if not counting the sensing energy consumption. At 1 packet per minute data rate, the lifetime of a WISP-Mote may last more than 13.4 years, while the lifetime for

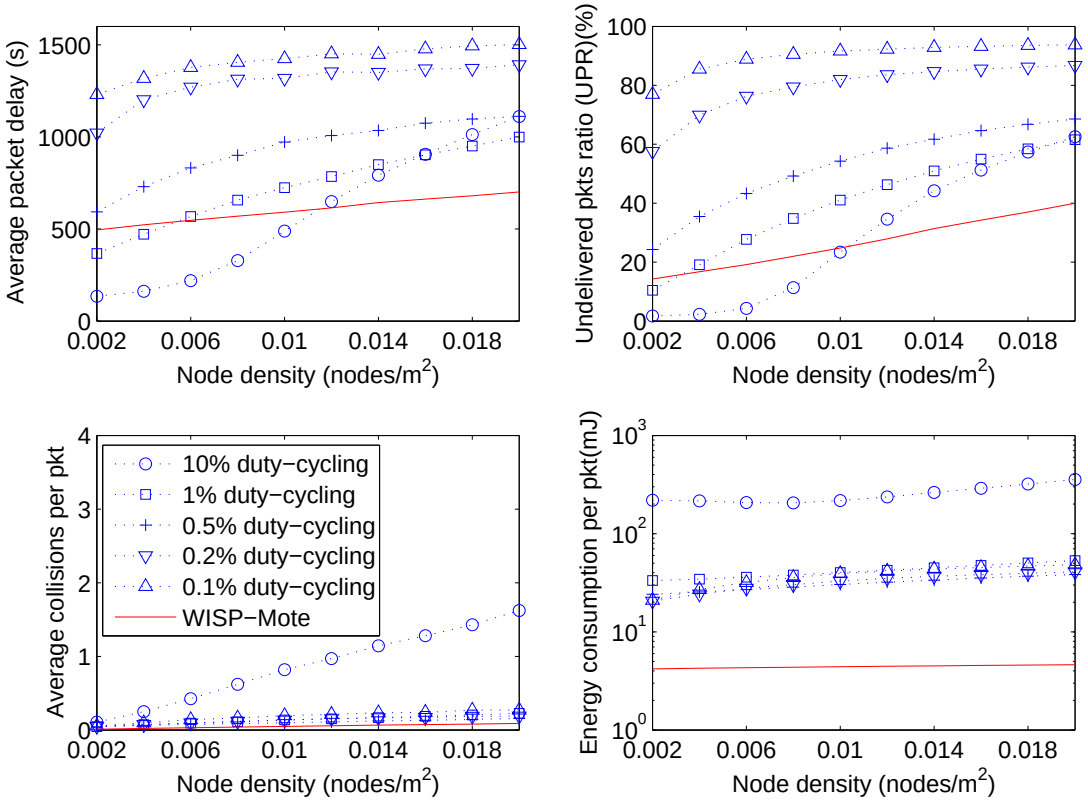


Figure 2.8: WISP-Mote vs. duty-cycling with increasing node density (1 pkt/min, 1 data MULE, unlimited buffer).

the best duty-cycling approach is only 3.5 years.

Effects of Node Density

Fig. 2.8 shows the performance of the network as the node density increases from 0.002 to 0.02 $node/m^2$, with a packet generation rate of 1 pkt/min, unlimited buffer size, and 1 data MULE. In other words, we increase the network from 80 nodes to 800 nodes, all of which are distributed in the 200 m × 200 m area. In our simulations, the nodes are not synchronized. For the duty-cycling sensor network, every node has a timer that is set randomly at the beginning of the simulation. Therefore, the lower the duty cycle is, the less chance nodes wake up at the same

time, which will reduce collisions.

As we can see, the 10% duty-cycling scenario is greatly affected by node density in that the chance of collision for 10% duty-cycling is much higher than the other cases. Therefore, as the node density increases, the number of collisions of 10% duty-cycling increases significantly from 0.1 collisions per delivered packet to 1.6 collisions per delivered packet. Additionally, the packet delay increases from $134.8\,s$ to $1110\,s$ and the UPR increases from 1.7% to 62.6%. While the other duty-cycling networks have smaller probabilities of collisions, but we can see that their performances decrease much faster than the WISP-Mote network's performance.

At a high node density of $0.02\,node/m^2$, the WISP-Mote network outperforms the duty-cycling networks in delay, UPR, collisions and energy efficiency. It is expected that with even higher node density, the performance differences will be even more significant. Because of the limited wake-up range, the chance of the hidden terminal problem and multiple nodes being simultaneously awakened and selecting the same transmission slot is very low. Thus, the performance of the WISP-Mote sensor network decreases slower than the duty-cycling sensor networks when the node density increases.

Effects of Buffer Size

In the previous simulations, we assumed each sensor node had an unlimited buffer. In other words, the node will never drop a packet even if it cannot deliver all the packets before a new one arrives, and hence the node accumulates more and more packets. This, however, is not realistic and is not reasonable for certain applications, e.g., applications that only care about the recently updated sensed data. Thus, in this subsection we consider limited buffer sizes and investigate the effect of buffer size on the network performances.

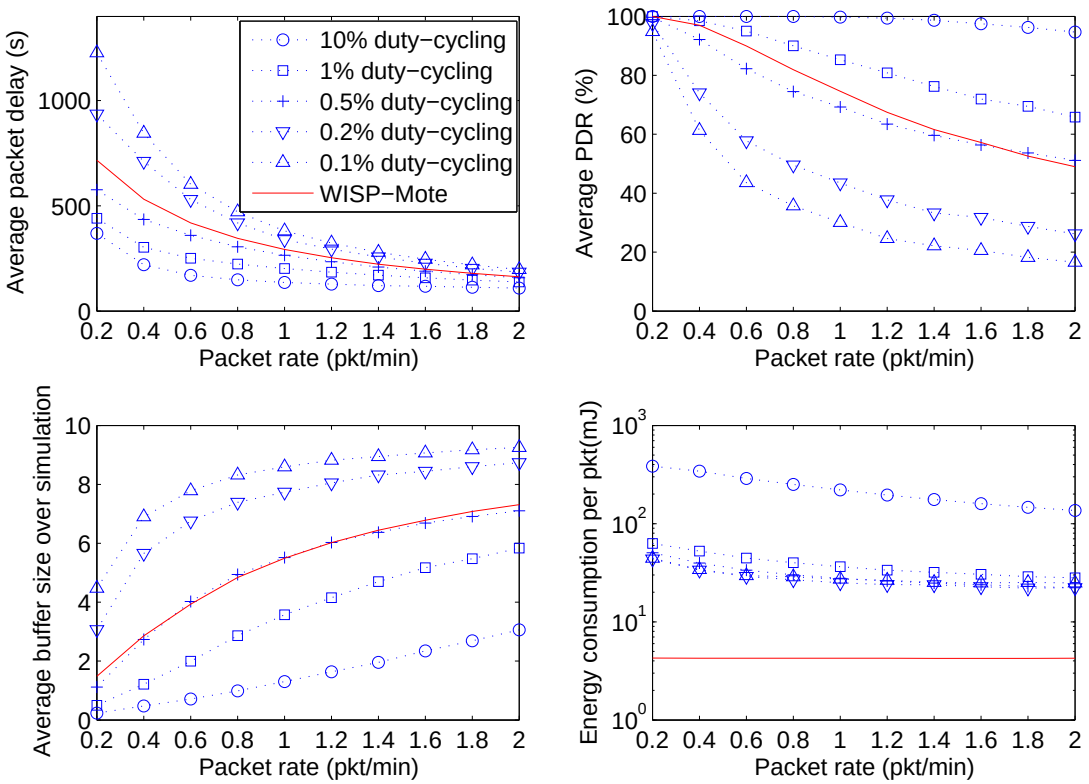


Figure 2.9: WISP-Mote vs. duty-cycling with limited buffer size (0.002 $nodes/m^2$, 1 data MULE, buffer size = 10 $pkts$).

In these simulations, we set the buffer size to be a small value, 10 packets, in order to show the impact of increasing packet rate given a fixed-size buffer. In real life, to have a higher packet delivery performance, the buffer size must be larger at the expense of longer packet latency. As the average packet delay only includes delivered packets, dropped packets are excluded from the packet latency calculations.

We can see from Fig. 2.9 that the packet delay decreases as the packet rate increases. The reason is that more and more old packets are dropped due to the limited buffer size and these dropped packets are not included in the calculation of the packet latency. The upper right subplot of Fig. 2.9 shows that the PDRs consistently decrease as the packet rate increases, which is reciprocal to the average

buffer usage in the lower left subplot. The energy consumptions for the duty-cycling and WISP-Mote schemes are consistent with the previous results.

Effects of the Number of Data MULEs

The data MULE sensor network performance depends on both the properties of the sensor nodes and the characteristics of the data MULEs. The packet delay and UPR are related to how fast the MULE(s) can sweep over the entire network. The moving speed of a data MULE is clearly correlated to the network performance. The faster the MULEs move, the lower the latency and the undelivered packet ratio that can be achieved.

A more interesting investigation is the effect of the number of MULEs. Besides packet collisions, the beacon signal and ACKs sent by the MULEs can also collide. Since the WISP-Mote does not need to receive a beacon signal to be informed of the MULE's appearance, it will not be affected by beacon signal collisions. Furthermore, the wake-up range is much smaller than the communication range in the WISP-Mote network. This greatly reduces the probability of MULE-node link collisions for the multi-MULE scenarios.

As Fig. 2.10 shows, the latency and UPR both perform better when there are more MULEs in the network. When there are multiple MULEs on the field, there are chances that one node is talking to multiple MULEs. In this case, the ACKs will collide and the node will have to retransmit the packet. We also consider these as collisions. At the same time, when the number of MULEs increases, the nodes can transmit packets more efficiently. Therefore, the average collision per packet delivery remain stable when there are multiple MULEs in the field.

For the duty-cycling scenario, the energy consumption to deliver one packet drops because the nodes are wasting less energy on idle listening when there are more MULEs (since they do not wake up when they have no data to send). On the

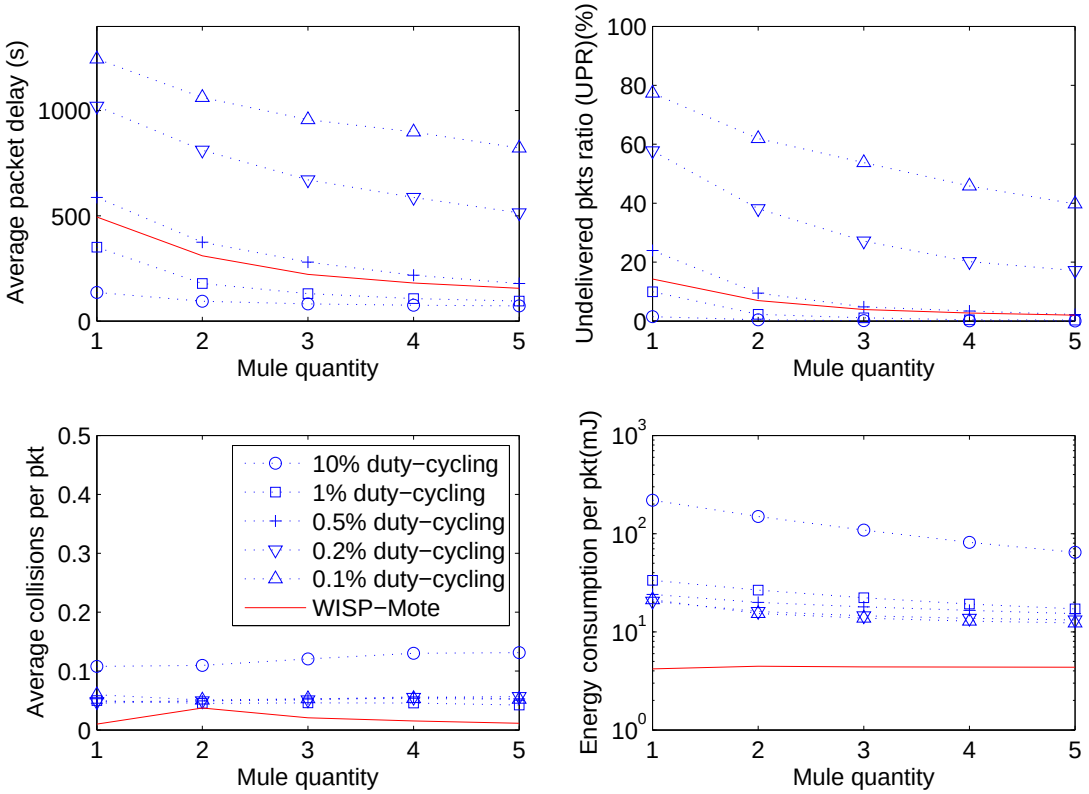


Figure 2.10: WISP-Mote vs. duty-cycling with increasing MULE quantities (0.002 $nodes/m^2$, 1 pkt/min, unlimited buffer size).

other hand, the energy efficiency for the WISP-Mote network remains quite stable. From the results, we can conclude that trading off with more infrastructure cost, we can achieve better performance by placing more data MULEs in the network.

Effects of the Mobility Model of the Data MULEs

The MULE's mobility pattern is another factor that has impact on the sensor network performance. We can expect that the best prescheduled route is to let the MULE directly go to the nodes one by one, but that would require pre-knowledge of the network topology (which may not be available for certain types of applications), and would introduce extra complexity to the MULE.

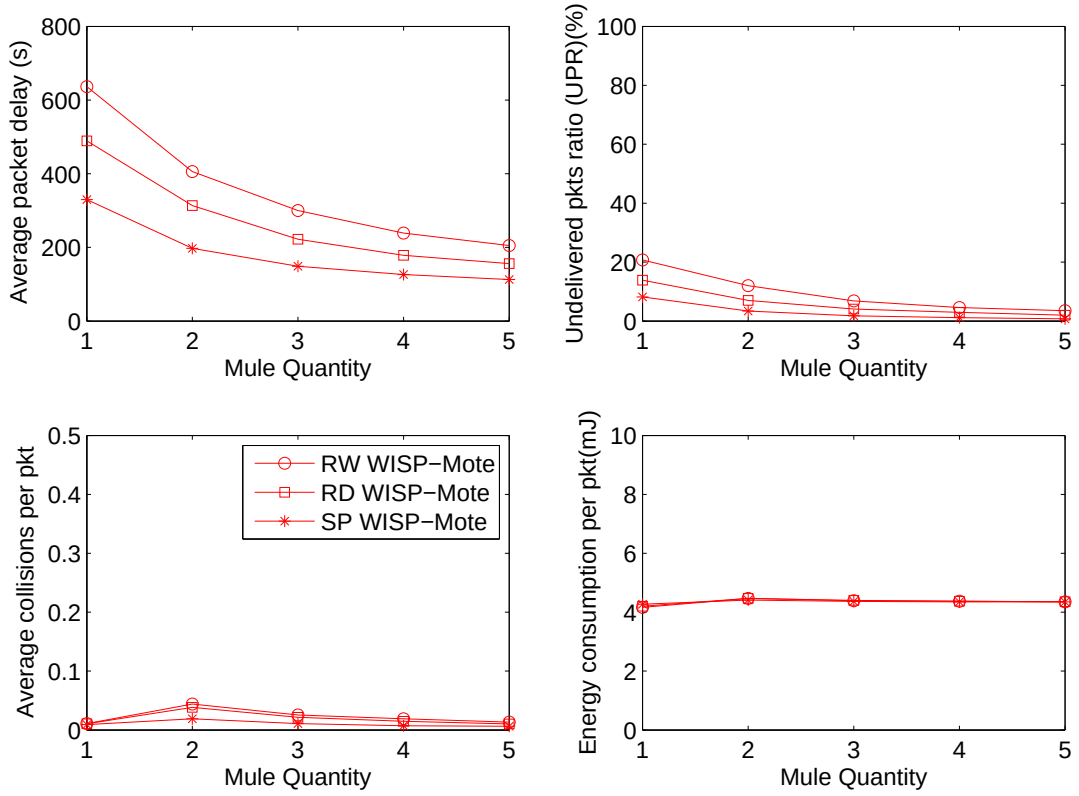


Figure 2.11: Comparisons among different mobility models for the data MULE; RW=Random Walk, RD=Random Direction, SP=Snake Path (0.002 $nodes/m^2$, 1 pkt/min, unlimited buffer size).

In our simulations, we compare three different mobility models for the WISP-Mote scenario: Random Walk, Random Direction, and Snake Path. The Random Direction algorithm has been described in Section 2.5.1. For the Random Walk algorithm, each MULE selects a speed from $[5\,m/s,\ 15\,m/s]$ and moves towards a direction chosen from $[0,\ 2\,\pi]$ for a period of time chosen from $[0,\ 10\,s]$. All these selections are done in a uniformly random manner from the intervals given. In the Snake Path algorithm, each MULE sweeps over the entire field by following a snake-shaped route with a constant speed of $10\,m/s$.

As we can see from the results, the MULE with Snake Path algorithm has a higher efficiency of covering the entire network. The Random Walk algorithm

is slightly worse than the Random Direction algorithm. When there are five MULEs, the network with 5 MULEs using the Snake Path algorithm can deliver 99.3% of the packets with an average latency of 112 s per packet, whereas the Random Direction mobility model has a latency of 156 s with 98.0% of the packets delivered. As for collisions, MULEs with prescheduled routes also outperform the other two approaches. Therefore, using data MULEs with prescheduled routes is the most efficient for collecting data from the sensors.

2.6 Applications That Can Benefit From WISP-Motes

The applications that can benefit from passive wake-up radio sensor networks are determined by the characteristics of the network in terms of transmission range, asymmetric energy consumption values, and the equipment costs of the receiver and the transmitter. Considering the hardware costs and the energy constraints, it is expected that there will only be a few powerful nodes in a sensor network that have the capability to wake up the other nodes in realistic applications. Even if all the sensors are assumed to have enough energy, the wake-up range is relatively short due to path loss and the efficiency of power harvesting at the receiver. As a result, to cover a large area of sensor nodes, either the wake-up signal transmitter has to be mobile to wake up the sensor nodes and collect data, or the sensor nodes have to be mobile to move to the base station to deliver data. Based on these features and constraints, we present several potential real-world applications that can benefit from passive wake-up sensor networks.

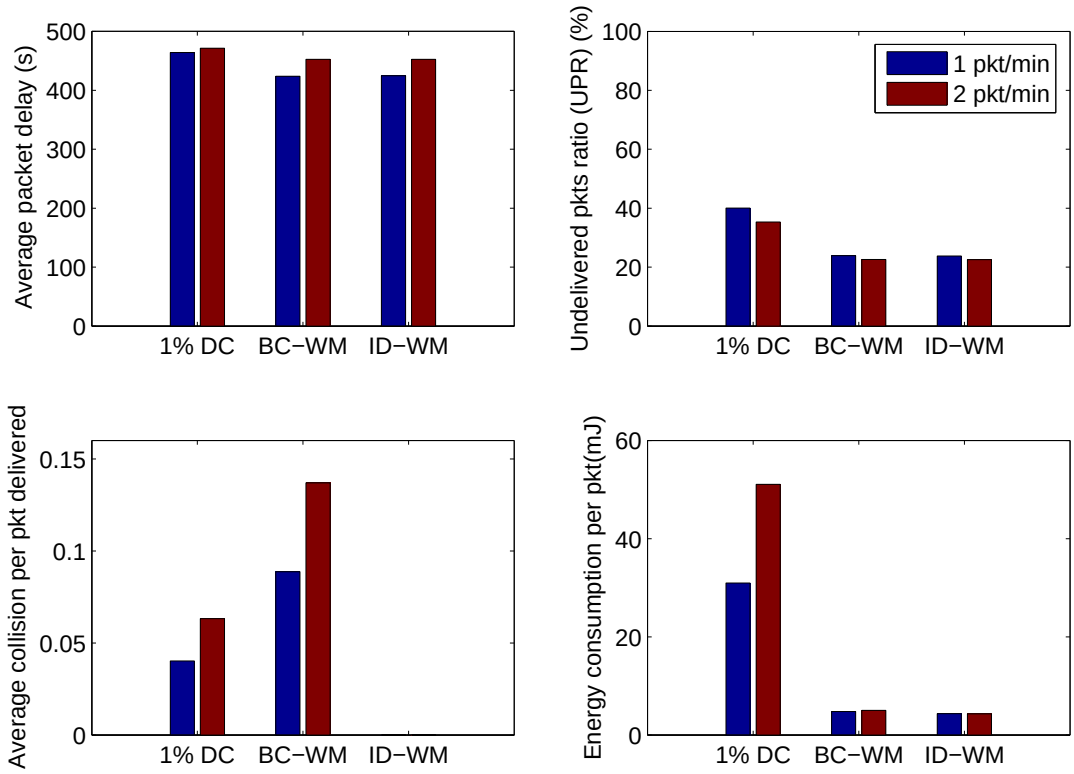


Figure 2.12: Patient monitoring scenario (unlimited buffer), "BCWM stands for Broadcast-based WISP-Mote, "ID-WM stands for ID-based WISP-Mote.

2.6.1 Health Monitoring Applications

Body area networks bring inexpensive health monitoring to different environments. Wake-up radio technology has great potential for such applications. In a home health monitoring application, such as the one we introduced at the beginning of this dissertation, instead of using traditional health monitors, which connect to a user's body with wires limiting his/her daily activities, each user can be equipped with various wireless sensor nodes for collecting different physiological information, e.g., heart rate, blood pressure, blood glucose, and insulin level. Wake-up radio transmitters are placed at specific locations such as kitchen and bedroom. With ID-based wake-up radio, only specific sensors will be awak-

ened and gather data or perform data transmissions when the user enter these locations. For example, when the user enters the kitchen, glucose sensor will be waken up and start monitoring glucose level; when the user goes to bed, EEG sensor will be awakened and start monitoring the user's brain wave. By reducing unnecessary sensing and transmission activities, the sensor node lifetime may be greatly improved.

Similarly, patient monitoring in long-term care facilities (e.g., a sanatorium or an assisted living facility), is another potential application. In these places, patients are cared for over a relatively long period of time and require non-emergency but long-term health monitoring (e.g., the data from the patients only need to be collected by nurses several times per day). Wireless body sensors are placed on patients' body as in the home health monitoring application. With all the equipments (the wake-up transmitter, the data receiver, the computer, and the power source) carried in a trolley, a nurse can easily gather all the sensor data from patients while periodically traveling throughout the patients' rooms.

We performed simulations for this scenario to compare the performance of duty-cycling, broadcast-based wake-up and ID-based wake-up. 1% duty-cycling is chosen for the duty-cycling scenario, as it has similar latency and packet delivery performance as the wake-up scheme in the previous data MULE scenario simulations. The wake-up probability models used for broadcast-based wake-up and ID-based wake-up are built based on our closed environment field test results.

In the simulations, we assume there are 36 rooms, each with an area of $20\,m \times 20\,m$. A "nurse" goes from room to room with a speed of $1\,m/s$ to collect data from the 4 sensors on each patient. We also set the nurse with 20% probability of pausing while moving.

The ID-based wake-up is beneficial in this scenario, since there are multiple nodes on each patient's body. If a broadcast-based wake-up is used, all of the nodes on a patient will be awakened with probability calculated based on our

closed environment field test results, which would cause congestions. Therefore, waking up the nodes one by one based on their type (or patient ID in other cases) will greatly reduce the chance of collisions and hence extend the nodes' lifetime.

The simulation results confirm our expectations. As we can see from Fig. 2.12, the delay for the three scenarios are generally about the same. However, both the WISP-Mote networks can deliver more packets than the 1% duty-cycling and consume about 5 times less energy. In terms of collisions, broadcast-based wake-up, as expected, has the highest number of collisions, which somewhat decreases its energy efficiency. ID-based wake-up, on the other hand, has zero collisions and consumes a little less energy ($4.3\,mJ/pkt$) than broadcast-based wake-up ($4.8\,mJ/pkt$). Overall, in the health-monitoring scenario, ID-based wake-up is the most energy efficient scheme with competitive performance in terms of delay and UPR.

2.6.2 Environmental Applications

Wildlife monitoring is one of the potential applications for passive wake-up radio sensor networks. A branch of zoology research investigates the behavior of or interactions between species. It is important to gather information on individual animals such as their locations or physiological data, as well as environmental information such as temperature and humidity, to understand the effects of the environment and influences from other species.

The cost of collecting data by equipping animals with sensor nodes is much lower than the cost of other approaches, e.g., volunteers. However, it is usually difficult for scientists to retrieve the sensor nodes from wild animals for battery replacement or recharging individually. Hence, the energy-efficient operation of the sensor nodes is crucial, which is the main motivation for using the radio wake-up technique.

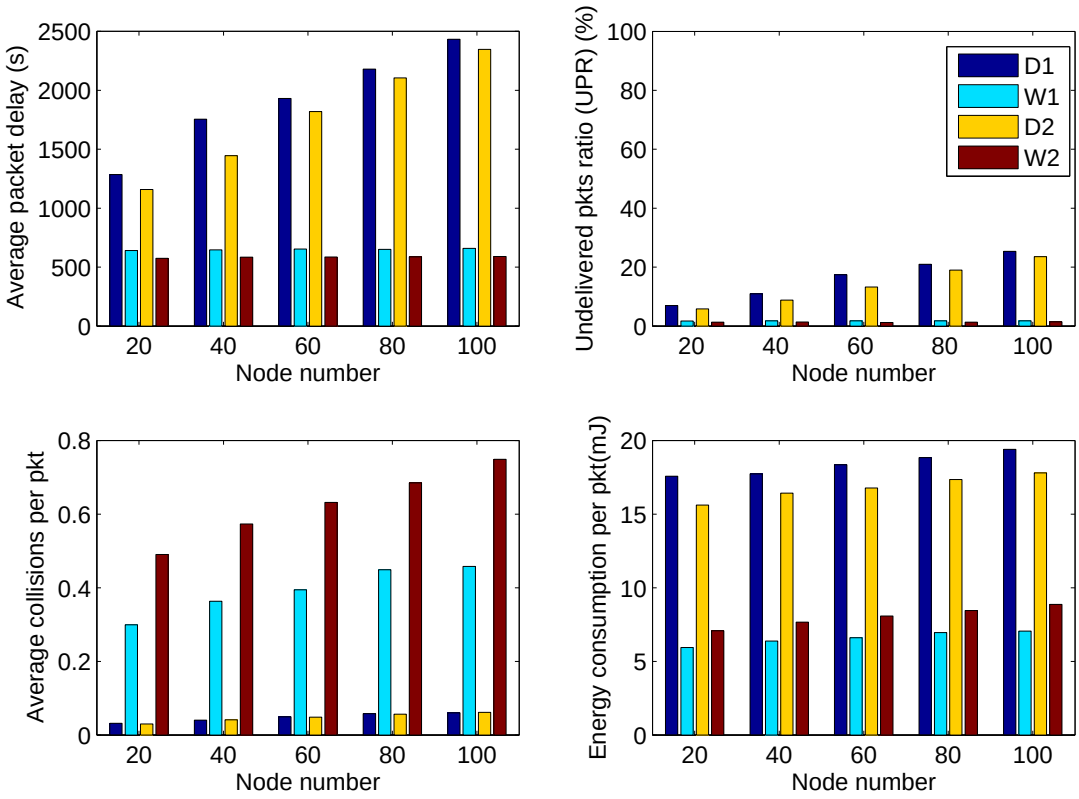


Figure 2.13: Animal monitoring scenario; D1= 1% Duty-cycling with R = 1, W1=WISP-Mote with R = 1, D2=1% Duty-cycling with R = 2, W2=WISP-Mote with R = 2 (0.2 pkt/min, buffer size unlimited)

Scientists may put sensor nodes on the animals, e.g., using a sensor collar, as done by researchers in the ZebraNet project [47], and place data collectors (equipment including a wake-up signal transceiver, data transceiver, and large energy supply) at places where the animals are expected to congregate, such as ponds and rivers, or where the animals are expected to roam. When the animals get close to the data collector, the radio on them will be awakened and start transmitting the gathered sensor data to the data collector. Since the sensor nodes may last for years, the battery retrieval and replacement costs can be saved.

We simulate this scenario by assuming a 1000 m by 1000 m area with a pond at the center. Assuming animals will tend to come to the pond for water, we

set a data collector at the pond sending out a broadcast-based wake-up signal continuously. Animals equipped with WISP-Mote sensor devices roam around this area and irregularly go to the pond for water. When the animals are roaming, they follow a random walk mobility model with average speed of $5\,m/s$. After a uniform random period of time with mean value of t_r, they move toward the pond and stay there for another uniform random amount of time with mean of t_s.

The ratio $\mathrm{R} = t_s \,/\, t_r$ impacts the network performance as is shown by the results below. A larger ratio of R represents a longer stay at the pond, and as we can see from Fig. 2.13, this results in a lower packet latency. For duty-cycling, lower packet delay means less idle listening. In other words, the energy waste is reduced and hence the energy efficiency is increased.

On the other hand, for the WISP-Mote scenario, the average number of collisions before a successful packet delivery is much higher than for duty-cycling. Yet, the undelivered packets and the energy consumption of the WISP-Mote scenario is much lower than the 1% duty-cycling scenario. This scheme will suffer from collisions if the number of nodes is large. As we can see, in this scenario, when R and the number of nodes increase, the average number of collisions per packet increases dramatically, which results in a decrease in the energy efficiency. One way to improve the network performance is to add a random delay before a WISP-Mote node performs carrier sensing. The maximum value of the delay should be adjusted according to the number of nodes in the network, which provides a good estimate of the probability of collision in practice. Since our goal for this chapter is not to design an application specific protocol, we do not optimize the WISP-Mote algorithm here.

2.6.3 Other Potential Applications

Daily activity monitoring is one application similar to the one envisioned at the beginning of this dissertation. In this application, passive wake-up radio receivers can be put on wearable sensors and smartphones (which nowadays are usually equipped with several sensors including accelerometer, gyroscope, proximity sensor, GPS, temperature sensor, light sensor and so forth, in order to track the smartphone's motion, location and environmental parameters). When the user enters certain areas where wake-up radio transmitters are located, the sensors or smartphones could be awakened to perform sensing activities. While the user is outside of the area, the sensors and smartphones will stay in a low power mode in order to save energy.

Intelligent Transportation System (ITS) is a very promising application that aims to improve traffic safety while providing other benefits such as reducing fuel consumption and latency. Using transportation facilities to collect sensor data throughout the city, i.e., urban monitoring [48], can be part of the ITS. The sensor data may include traffic conditions and weather information that will be helpful for congestion avoidance and improving traffic safety. Battery powered sensor nodes can be easily attached to traffic lights, bus stations, toll booths, and traffic signs and may work for years due to the energy efficiency improvements from using wake-up radios. Buses, trains and garbage trucks are good candidates for performing data collection duties, as they can cover a fairly wide area [49, 50] and they have no energy limitations so that transmitting a powerful wake-up signal is not a problem.

2.7 Conclusions

In this chapter, we presented a novel device that utilizes current commercial sensor nodes and programmable RFID tags to implement a passive wake-up radio sensor

network. Field tests in various environments are performed to characterize the wake-up probability as a function of distance between the wake-up transmitter and receiver. The results indicate that the wake-up radio performs the most stable as distance increases in an open environment compared to a closed environment and a cluttered environment. The results also prove that the wake-up range is very limited compared to the ZigBee-compliant sensor mote communication range. In addition, the radio wake-up transmitter requires high energy consumption that cannot be applied to all the sensor nodes considering the hardware cost and the energy budget. Both of these introduce a coverage problem.

To extend the sensor network coverage, mobility has to be introduced into passive wake-up radio based networks. We conduct detailed investigations on a data MULE scenario to study the impacts of packet generation rate, node density, node buffer size, MULE quantity, and MULE mobility model on network performance. The simulation results indicate that in the mobile scenarios we investigated, the passive wake-up radio sensor network achieves significantly better energy efficiency than duty-cycling. Furthermore, potential applications of passive wake-up radio sensor networks are discussed, and simulations are performed based on two specific application scenarios. The results reveal that passive wake-up radio sensor networks have comparable packet latency and packet delivery performance while greatly improving the energy efficiency.

Increasing the RFID reader-to-tag communication range will improve the performance of passive wake-up radio networks. The directional antenna and beam forming technique mentioned in [32] provide a possible solution to extend the wake-up range without increasing the wake-up transmitter power. As another approach, Omni-ID Corporation [51] has developed a patented technology called a “plasmatic structure” that captures incoming RF waves and creates a region of highly concentrated energy around the RFID, greatly extending the operational range.

3 Noise Resilient Pitch Detection from Speech Data

3.1 Introduction

In many applications for energy efficient sensing and computing systems, sensing signals are noisy and must be processed to improve the signal-to-noise ratio or to extract relevant information. For example, in the stress and emotion classification application described in Chapter 1, sensing data from a human subject must be obtained. These data may include speech data, image data of facial expression and body expressions, electrocardiograph (ECG) and electroencephalograph (EEG). From the sensing data, features that contain important emotion information can be acquired. For example, from speech data, we can extract pitch, intensity, formant, linear prediction coefficients (LPC), linear predictive cepstral coefficients (LPCC), mel-frequency cepstral coefficients (MFCC) and linear frequency cepstrum coefficients (LFCC). However, if the original data is corrupted by noise, obtaining these features from the noisy speech data is challenging. In this chapter, we focus on the detection of pitch, which is one of the most essential features in emotion classification from speech [52], from a noisy speech signal.

For human speech, pitch is defined by the relative highness or lowness of a tone as perceived by the human ear, and is caused by vibrations of the vocal

cords. Since pitch is a subjective term, in this chapter, we use the objective term fundamental frequency (F_0), which is an estimate of pitch. If there were perfectly periodic speech signals, F_0 would be the inverse of the period of voiced speech. However, the interference of formant structure for speech signals, or the interference of spectral envelope structure for music signals, makes the accurate detection of F_0 difficult. Also, due to the aperiodicity of the glottal vibration itself and the movement of the vocal tract that filters the source signal, human speech is not perfectly periodic [53]. Additionally, accurate F_0 detection is difficult when the speech signal is corrupted with noise. Therefore, F_0 detection has always been a challenge in speech signal analysis.

A variety of speech-based applications can benefit from a more precise and robust F_0 detection algorithm. For example, F_0 detection is essential in automatic speech recognition, where pitch-accent patterns can be used to improve recognition performance [54], or homophones can be differentiated by recognizing tones [55]. For synthesized speech to be natural and intelligible, it is crucial to have a proper F_0 contour that is compatible with linguistic information such as lexical accent (or stress) and phrasing in the input text. Therefore, F_0 modeling can be used for speech synthesis [56] [57]. F_0 and azimuth cues can be jointly used for speech localization and segregation in reverberant environments [58]. Moreover, in emotion detection or other affective measurement, it has been found that prosodic variations in speech are closely related to one's emotional state, and the F_0 information is important for the identification of this state [59]. A warning system has been developed in [60] to detect if a driver exhibits anger or aggressive emotions while driving, using statistics of the F_0 value and other metrics. Some health care providers and researchers implemented F_0 detectors and other behavior sensing technologies on mobile devices, such as smartphones, for behavioral studies [61] [62] or patient monitoring, such as the clinical trials conducted by the University of Pittsburgh for detecting depression in patients [63]. However,

for these types of applications, the vehicle noise captured by the detector or the ambient noise captured by mobile devices may strongly influence the F_0 detection performance.

F_0 detection also plays a very important role in music signal analysis and music information retrieval, and has a broad range of applications. Music notation programs use F_0 detection to automatically transcribe real performances into scores [64]. Reliable F_0 extraction from humming is critical for query-by-humming music retrieval systems to work well [65]. Music fingerprinting technology also uses F_0 information for music identification among millions of music tracks [66]. F_0 detection in noisy music is also challenging. Music may be recorded in noisy environments such as in a bar or on the street. Noise may also be introduced by the recording device itself. One challenge is that the F_0 generated from tonal musical instruments spans a large range, normally from 50 Hz to 4,000 Hz [67]. For musical signals with high F_0, the wide range of possible F_0 candidates increases the likelihood of finding a wrong F_0 value. The other challenge is that, unlike for human speech, the sound for musical signals can last for several seconds, thus the overlapped musical tones can also be considered as noise. Due to these reasons, when performing F_0 detection in real scenarios, the quality of the input signal may be greatly degraded. Therefore, F_0 detection of musical signals in noisy environments is necessary.

Adding noise may introduce spectral peaks in the spectrum of the speech signal or distort the shape of the speech peaks, depending on the type and level of the noise. The key to detecting F_0 from noisy speech or music is to differentiate speech or music spectral peaks from noise peaks. In Fig. 3.1, we plot the spectrum of one frame from a clean speech file and the same frame with babble noise at 0 dB SNR. By examining this frame, we can see that F_0 is located at 192 Hz. By comparing the spectrum of the clean speech and the noisy speech, we can see that the added noise peaks distort the shape of the speech peaks, causing the highest point of

the peak to be shifted. For the noise at 0 dB SNR, the amplitudes of the noise peaks can even be comparable with the amplitudes of the speech peaks. However, their locations in the frequency domain are not periodic, and the distribution of the noise peaks in the frequency range varies for different types of noise. Thus, the locations of the spectral peaks are affected less by the additive noise than the amplitudes of the peaks. Therefore, the ratios of harmonic frequencies are essential to find F_0 from a noisy signal.

Also, as seen in the spectrum of the noisy speech shown in Fig. 3.1, the first four harmonics are located at 391 Hz, 581 Hz, 760 Hz, and 958 Hz. The spectral peak located at 485 Hz is from the noise signal. We can see that the harmonics are not exactly spaced at integer multiples of the fundamental frequency F_0 in the frequency domain, and the higher order harmonics have larger drift than the lower order harmonics. Therefore, we need to set a tolerance range to account for the drifts when calculating the ratios of harmonic frequencies.

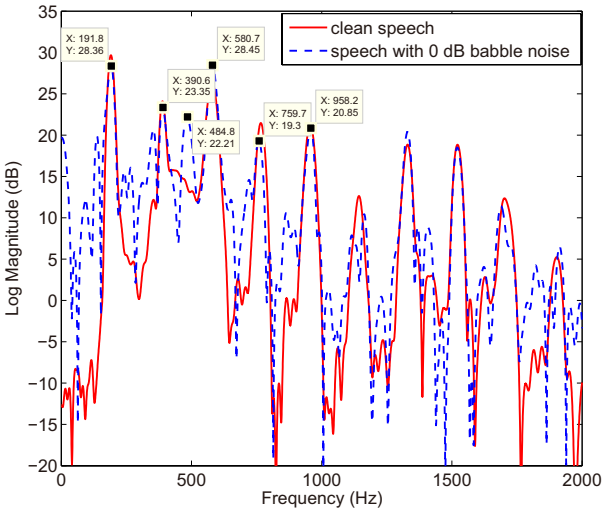


Figure 3.1: Spectrum of one frame of clean speech and speech with babble noise at 0 dB SNR.

As existing F_0 detectors, such as Cepstrum [68], HPS [69], and Praat [70], do not perform well when the input data is noisy, we are motivated to design a noise resilient F_0 detection algorithm that is better suited for practical uses. This chapter is based on our previous work [71], which proposed the BaNa algorithm for F_0 detection in speech signals. The BaNa algorithm is a hybrid F_0 detection algorithm that combines the idea of using the ratios of harmonic frequencies with tolerance ranges and the Cepstrum approach to find F_0 from a noisy signal. In this chapter, we discuss F_0 detection for both speech and music signals, and we describe the simple modifications of BaNa required for music F_0 detection. We show that using the ratios of harmonic frequencies with pre-tuned tolerance ranges for F_0 detection enables the algorithm to be resilient to additive noise. We also show that incorporating Cepstrum and post-processing techniques can improve the F_0 detection performance.

In addition, we extend the work in [71] by evaluating the BaNa algorithm on a range of speech databases and by comparing it with seven classic and state-of-the-art F_0 detection algorithms. We test the proposed BaNa algorithm on real human speech and music samples corrupted by various types and levels of realistic noise. Evaluations show the high noise resilience of BaNa compared to the classic and state-of-the-art F_0 detection algorithms. For noisy speech at 0 dB SNR, BaNa achieve 20% to 35% Gross Pitch Error (GPE) rate for speech and 12% to 39% GPE rate for music. Also, we discuss issues with implementing BaNa on a smartphone platform. Test results on a real device show that our implementation of BaNa can process recorded speech files with a reasonable speed, opening the door for real-time F_0 detection on mobile platforms.

The rest of the chapter is organized as follows. Section 3.2 provides a brief survey of well-known F_0 detection algorithms and also some of the most recent F_0 detection algorithms. Section 3.3 describes the BaNa algorithm for F_0 detection in noisy speech. Experimental settings and extensive experimental results com-

paring the BaNa algorithm with several classic and state-of-the-art F_0 detection algorithms using different speech databases are presented in Section 3.4 and Section 3.5, respectively. Section 3.6 presents the slight modifications of the BaNa algorithm to improve its performance for F_0 detection in noisy music. We describe an implementation of the BaNa F_0 detection algorithm in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.2 Related Work

Among the well-known classic F_0 detection algorithms, autocorrelation function (ACF) and cross correlation are among the most widely used time domain methods. A number of algorithms have been developed based on these two approaches. Average Magnitude Difference Function (AMDF) [72] is a variation of ACF, which calculates a formed difference signal between the delayed signal and the original one. Since the AMDF algorithm does not require any multiplications, it is desirable for real-time applications. Praat [70] considers the maxima of the autocorrelation of a short segment of the sound as F_0 candidates, and chooses the best F_0 candidate for each segment by finding the least cost path through all the segments using the Viterbi algorithm. YIN [73] uses a novel difference function similar to autocorrelation to search for the F_0 period. It further refines the F_0 detection result using some post-processing methods. Two types of modified difference function used in YIN are proposed in [74]. The RAPT F_0 tracker [75], on the other hand, is a variation of cross correlation, which computes the F_0 by extracting the local maxima of the normalized cross correlation function.

In the frequency domain, F_0 is found by searching for harmonic peaks in the power spectrum. The Cepstrum method [76] [68] is among the most popular algorithms. Cepstrum is found by computing the inverse Fourier transform of the log-magnitude Fourier spectrum, which captures the period in the speech

harmonics, and thus shows a peak corresponding to the period in frequency.

Schroeder's frequency histogram [69] enters all integer submultiples of all the peaks in the spectrum in a histogram. Since F_0 is the integer submultiple of all the harmonics, in an ideal case, the entry with the highest weight in Schroeder's frequency histogram is the correct F_0. As pointed out in [77], Schroeder's frequency histogram is susceptible to octave errors, as F_0 and $F_0/2$ will have the same weight in Schroeder's frequency histogram. In cases where noise peaks are selected, Schroeder's histogram will make mistakes by selecting the greatest common divisor of both the harmonics and the noise peaks.

The Harmonic Product Spectrum algorithm (HPS) [69] multiplies the original signal with downsampled signals, thus in the frequency domain, the spectra of all the downsampled signals line up the peaks at the F_0 value for isolation. Another harmonic summation method is proposed in [78], which modifies the HPS method for multiple F_0 estimation in polyphonic music. The harmonic components' energy distribution is used, and F_0 candidates are selected using a competition mechanism. The algorithm is tested on three different instruments. However, for these harmonic summation methods, noise peaks with high amplitudes can be easily mistaken for harmonic peaks at low SNR scenarios. Since our proposed BaNa algorithm only relies on the locations of the harmonic peaks to calculate the frequency ratios of those spectral peaks, with the peak's amplitude information only being considered for peak selection, we show in Section 3.4.3 and Section 3.6.5 that the BaNa algorithm is more robust than the HPS algorithm for noisy speech and noisy music.

The PEFAC (Pitch Estimation Filter with Amplitude Compression) algorithm proposed in [79] is another frequency-domain F_0 detection algorithm for noisy speech. This approach estimates F_0 by convolving its power spectral density in the log-frequency domain with a filter that sums the energy of the F_0 harmonics while rejecting additive noise that has a smoothly varying power spectrum. Also,

amplitude compression is applied before filtering to attenuate narrow-band noise components.

Some F_0 estimators operate in the time-frequency domain by applying time-domain analysis on the output of a filter bank. In the algorithm proposed by Jin and Wang [80], a new frequency channel selection method is proposed to select less corrupted channels for periodicity feature extraction. F_0 scores for each F_0 state are derived given the periodicity features and are given to a hidden Markov model for F_0 state tracking.

Recently, an increasing number of F_0 detection algorithms have been designed using a data-driven statistical approach to combat noise, such as the algorithms described in TAPS [81], Wu [82], and SAFE [83]. In [81], Huang et al. propose an F_0 estimation method that uses Temporally Accumulated Peaks Spectrum (TAPS). Since the harmonic structure of voiced speech changes more slowly than the noise spectrum over time, spectral peaks related to F_0 harmonics would stand out after temporal accumulations. Clean and noisy speech data is required to train the peak spectrum exemplar set and the Gaussian mixture model.

The Wu algorithm [82] is also a statistical approach, which integrates a new method for extracting periodicity information across different channels, and a Hidden Markov Model for forming continuous F_0 tracks. The modeling process incorporates the statistics extracted from a corpus of natural sound sources. The SAFE algorithm [83] also uses a data-driven approach to model the noise effects on the amplitudes and locations of the peaks in the spectra of clean voiced speech.

However, these data-driven approaches may not always be practical. Since these algorithms are trained with known noise types and specific noise levels, the noise information of the test sample is also required as input to the model. However, this information is not always available, since the user often does not know the type of noise, and it is even harder to measure the noise level. The proposed BaNa algorithm, on the other hand, does not require any prior information about

the noise.

Though most F_0 detection algorithms were developed for F_0 detection in speech, a number of the aforementioned algorithms have also been used in music. The YIN and Praat algorithms are evaluated in [84] for synthetic signal and real-time guitar signal F_0 tracking. In [85], F_0 detection performance of the HPS algorithm and its variation called Cepstrum-Biased HPS are compared for interactive music. Clean cello and flute pieces are used in the experiments. However, robust F_0 detection in noisy music is still a topic that needs to be explored.

In this chapter, we perform an extensive quantitative comparison analysis to show the performance, in terms of Gross Pitch Error (GPE) rate, for our proposed F_0 detection algorithm, BaNa, and several of the aforementioned algorithms (YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu) for noisy speech and music signals.

3.3 BaNa F_0 Detection Algorithm for Speech

In this section, we describe the BaNa hybrid F_0 detection algorithm for speech.

3.3.1 Preprocessing

Given a digital speech signal, preprocessing is performed before the extraction of the F_0 values. In the BaNa algorithm, we filter the speech signal with a bandpass filter. Let F_0^{min} and F_0^{max} denote the lower limit and upper limit for F_0 values of human speech, respectively. The lower bound of the bandpass filter is set to F_0^{min}. The upper bound is set to $p \cdot F_0^{max}$, where p is the number of spectral peaks captured that will later be used for F_0 detection.

3.3.2 Determination of the F_0 candidates

Since harmonics are regularly spaced at approximately integer multiples of F_0 in the frequency domain, we use this characteristic of speech in the proposed BaNa algorithm to achieve noise resilience. If we know the frequency of a harmonic and its ratio to F_0, then F_0 can be easily obtained. However, even if a harmonic is discovered, its ratio to F_0 is unknown. Therefore, we propose an F_0 detection algorithm that looks for the ratios of potential harmonics and finds the F_0 based on them by applying the following steps.

Step 1: Search for harmonic peaks

Spectral peaks with high amplitudes and low frequencies are preferred to be considered for F_0 candidates, since peaks with high amplitudes are less likely to be caused by noise, and peaks with low frequencies are easier to be identified to be harmonics by calculating the ratios. Peaks with high frequencies may be high order harmonics, which cannot be used to calculate harmonic ratios, since we only consider the ratios of the first p harmonics. Therefore, we consider the p peaks with amplitudes higher than a certain threshold and with the lowest frequencies to derive F_0 candidates. We use the peak detection algorithm provided in [86] to search for the peaks in the spectrum. In [86], spectral peaks with small amplitudes are filtered out by setting a peak amplitude threshold, and peaks located very close to dominant peaks are smoothed by setting a threshold of the window width for smoothing. Settings that we use for the number of selected peaks p, the peak amplitude threshold parameter, and the window width parameter for the smoothing function in the peak detection function are presented in Table 3.2.

Let $\hat{F}_i$ and $\mid \hat{H}_i \mid$ represent the frequencies and the magnitudes of the p spectral peaks with the lowest frequencies whose magnitudes are above a certain threshold, where $i = 0, \cdots, p-1$. We place the p peaks in ascending order of frequency to obtain the set of $\hat{F}_i$, denoted as $\hat{F}$. For most human speech, energy concentrates in

the low frequency part, thus some or all of the p peaks are likely to be at the first p harmonics, which are at $m \times F_0$, $m = 1, \cdots, p$. For each frame, F_0 candidates are derived from the ratios of the frequencies of $\hat{F}$ using the following algorithm.

Step 2: Calculate F_0 candidates

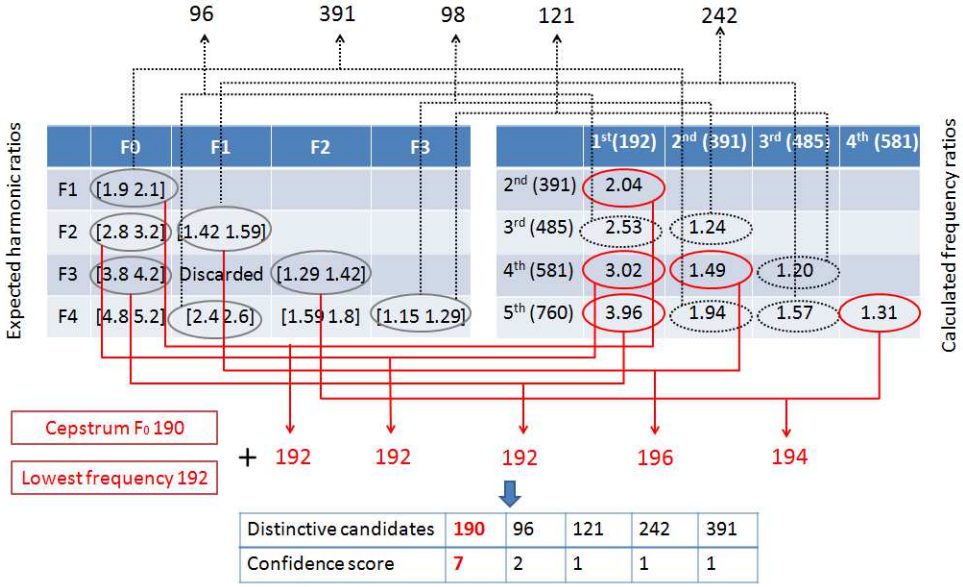


Figure 3.2: Tolerance ranges for harmonic ratios when the number p of selected spectral peaks is set to 5, and an example to illustrate the procedure for determining the F_0 candidates.

We calculate the ratios $R_{ij} = \hat{F}_j / \hat{F}_i$ for all $\hat{F}_i, \hat{F}_j$, where $i < j$, and $i, j = 0, \cdots, p-1$. Take the number of selected spectral peaks $p = 5$ for example. If a calculated ratio R_{ij} falls into any tolerance range of the harmonic ratios shown in the left table in Fig. 3.2, we are able to find to which harmonics $\hat{F}_i$ and $\hat{F}_j$ correspond. For harmonic ratios with small numbers, we set adjacent tolerance ranges to be bounded with each other, i.e., the upper bound of the tolerance range of the ratio of $\hat{F}_4$ and $\hat{F}_3$ is the same as the lower bound of the tolerance range of the ratio of $\hat{F}_3$ and $\hat{F}_2$, which is $(5/4 + 4/3)/2 = 1.29$, as shown in Fig. 3.2. For harmonic ratios with large numbers, the width of the tolerance range is set to

0.2 or 0.4, depending on the ratio number. We show in Section 3.4.3 that these tolerance range numbers are tuned to achieve the best F_0 detection performance.

A potential F_0 candidate can be obtained by dividing the harmonic by its ratio to F_0: $\tilde{F} = \hat{F}_i/m$, where $m = 1, \cdots, p$. Note that due to the imperfect periodicity of human speech, the harmonics may not be exactly on integer multiples of F_0, and we observed that higher order harmonics have even larger drift than lower order harmonics in practice. Therefore, we set a smaller ratio tolerance range for lower order harmonics, and we set a larger ratio tolerance range for higher order harmonics. In total, C_2^p ratio values are calculated between every pair of $\hat{F}$. Since both ratios of F_1/F_0 and F_3/F_1 are equal to 2, it is not trivial to differentiate to which harmonics this ratio belongs. In our algorithm, we assume it belongs to F_1/F_0 and calculate the F_0 candidate based on that.

In order to combat these octave errors, the proposed BaNa algorithm adds two other F_0 candidates. One added candidate is the spectral peak with the smallest frequency value, since we have found that in some cases only the F_0 peak is high enough to be detected. The other added F_0 candidate is the F_0 value found by the Cepstrum method. The reason is that the p spectral peaks we choose mainly belong to low frequency values. For some rare cases, the higher order harmonics (e.g., 5th to 10th harmonics) are found to yield higher spectral peak values compared to the low order harmonics. In that case, the spectral peaks at low frequencies are more vulnerable to noise. However, since the Cepstrum method depicts the global periodicity of the spectrum, and considers all spectral peaks, it can help to detect the F_0 in those rare cases. In Section 3.5.2, we show the benefit of including the spectral peak with the smallest frequency value and the Cepstrum F_0 as additional candidates.

The number of F_0 candidates derived from the ratio analysis and the two added candidates, K, is then less than or equal to $C_2^p + 2$. Among these K F_0 candidates, the ones that are out of the F_0^{min} to F_0^{max} human speech range

are discarded, and the number of candidates is reduced from K to K'. If no candidate is derived from the ratios, we set the F_0 value to 0 Hz. We then order the K' candidates in ascending order of frequency. F_0 candidates that are within ξ Hz of each other are considered to be "close" candidates. For each of these K' candidates $\tilde{F}_k$, where $k = 1, ..., K'$, we count the number of "close" candidates U_k, and select the one with the largest number of "close" candidates to be a "distinctive" candidate $\check{F}_d$, where $d = 1, ..., D$, and D is the number of "distinctive" candidates. The "distinctive" candidate and its "close" candidates are deleted from the candidate list. If there is more than one candidate that has the same largest number of "close" candidates, we select the one with the smallest frequency to be the "distinctive" candidate. We continue the same procedure for the remainder of the list until the list is empty. We set the number of "close" candidates, including the chosen candidate itself, to be the confidence score V_d for the corresponding "distinctive" candidate $\check{F}_d$. Among the D "distinctive" candidates, where $D \leq K'$, the ones with higher confidence scores are more likely to be F_0.

In Fig. 3.2, we use the frame shown in Fig. 3.1 to illustrate the process of calculating F_0 candidates. In Fig. 3.1, the dotted line represents the spectrum of one frame of speech with 0 dB babble noise. The five selected spectral peaks that have the lowest frequencies are located at 192 Hz, 391 Hz, 485 Hz, 581 Hz, and 760 Hz. The 485 Hz peak is caused by the noise signal, and the remaining four peaks are from the speech signal. We map each calculated frequency ratio in the right table in Fig. 3.2 to one expected harmonic ratio in the left table in Fig. 3.2. For example, the ratio of the 5^{th} and 4^{th} spectral peaks is $\hat{F}_5/\hat{F}_4 = 760/581 = 1.31$, which maps to the [1.29 1.42] frequency ratio tolerance range for the expected frequency ratio of the 3^{rd} and 2^{nd} harmonics. Therefore, the F_0 candidate is derived as 581/3=194 Hz. In this example, all calculated frequency ratios are mapped to one expected harmonic ratio in the left table, which results in 10 F_0

candidates. The Cepstrum candidate and the peak with the lowest frequency are added as two additional F_0 candidates, which are 190 Hz and 192 Hz, respectively.

If we use the parameters shown in 3.4.3, all the 12 candidates are within the $F_0^{min} = 50$ Hz to $F_0^{max} = 600$ Hz range for F_0. Candidates that are within $\xi = 10$ Hz of each other are considered to be "close" candidates. In Fig. 3.2, the bottom table shows the "distinctive" candidates and their confidence scores. The 190 Hz candidate has the highest confidence score, which is very close to the ground truth F_0, i.e., 191 Hz calculated from the corresponding clean speech signal. In Fig. 3.2, correct F_0 candidates are listed on the bottom and are marked by solid red lines. Incorrect F_0 candidates are listed on the top and are marked by dotted black lines. We can see that the candidates calculated from the 485 Hz noise peak are all incorrect candidates.

3.3.3 Selection of F_0 from the candidates

In Section 3.3.2, the "distinctive" candidates of individual frames are obtained independently. However, the F_0 values of neighboring frames may correlate, since the F_0 values of human speech exhibit a slow time variation, and hence, large F_0 jumps among subsequent frames are rare. Therefore, we use the Viterbi algorithm [87] for post-processing to go through all the candidates in order to correct F_0 detection errors, similar to the post-processing used in the Praat algorithm [70]. We aim to find a path that minimizes the total cost, which consists of two parts: the frequency jumps between the candidates of two consecutive frames, and the inverse of the confidence scores of each "distinctive" candidate.

Let $\check{F}_i^n$ denote the i^{th} "distinctive" F_0 candidate of frame n and N_{frame} denote the number of frames in the given speech segment. Moreover, let p_n denote the index of the chosen F_0 candidate for the n^{th} frame. Thus, $\{p_n \mid 1 \leq n \leq N_{frame}\}$ defines a path through the candidates. For each path, the path cost is defined to

be

$$PathCost\left(\{p_n\}\right) = \sum_{n=1}^{N_{frame}-1} Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right), \tag{3.1}$$

where $p_n = i$ and $p_{n+1} = j$. The *Cost* is used to calculate the cost of adjacent frames. We define the function *Cost* by using the F_0 differences between the adjacent frames and the confidence score of the candidates. The F_0 difference is modeled similarly with the *transition cost* defined in the Praat algorithm [70]. The larger the F_0 difference, the higher the *Cost* should be. We present the F_0 difference in the Mel scale, which is a perceptual scale of F_0 judged by listeners. The perceived F_0 in the Mel scale has a logarithm relation with the F_0 in frequency, as shown in (3.2):

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right). \tag{3.2}$$

Therefore, in the cost function, the F_0 difference in frequency is modeled as the logarithm of the F_0 division in the Mel scale. The other part of the cost function is modeled using the confidence score. We assign a lower cost to those F_0 candidates with higher confidence scores, thus we use the inverse of the confidence score in the expression of the cost function. A weight w is introduced to balance the two parts. The setting for this value is shown in Table 3.2. Then, *Cost* is defined mathematically as

$$Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \mid \log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}} \mid + w \times \frac{1}{V_i^n}, \tag{3.3}$$

where V_i^n is the confidence score of the i^{th} "distinctive" F_0 candidate of the n^{th} frame.

We use the Viterbi algorithm to find the minimum cost path, i.e., the path that reduces the F_0 jumps the most, while giving priority to the F_0 candidates with higher confidence scores. The optimal path is found for each voiced part in the speech. The Praat algorithm also uses the Viterbi algorithm to choose F_0 from several F_0 candidates for each frame. However, the F_0 candidates of Praat are

local maxima of the autocorrelation of each frame, which have the same confidence score to be selected as F_0. On the other hand, the F_0 candidates in BaNa have different confidence scores, and thus F_0 candidates derived from noise spectral peaks are less likely to be selected as F_0. Therefore, the cost function of BaNa's Viterbi algorithm shown in (3.3) is different from that in the Praat algorithm. The complete BaNa algorithm that describes the selection of the peaks and the calculation and selection of the F_0 candidates is given in Algorithm 3.

Algorithm 3 The BaNa F_0 Detection Algorithm

// ***For frame*** n***:***
// ***Select harmonic peaks***
select $\hat{F}^n$: the p peaks with lowest frequencies
// ***Calculate*** F_0 ***candidates***
number of candidates $K \leftarrow 0$
for i =1 to p, $j = i + 1$ to p **do**
    ratio $R_{ij} = \hat{F}_j^n / \hat{F}_i^n$
    for m =1 to p, $m' = m + 1$ to p **do**
        if R_{ij} falls in the left table of Fig. 3.2 and close to $\frac{m'}{m}$ **then**
            $K \leftarrow K + 1$, $\tilde{F}_K^n \leftarrow \hat{F}_i^n / m$
        end if
    end for
end for
$K \leftarrow K + 1$, add spectral peak with the lowest frequency $\hat{F}_1^n$: $\tilde{F}_K^n \leftarrow \hat{F}_1^n$
$K \leftarrow K + 1$, add Cepstrum F_0: $\tilde{F}_K^n \leftarrow Cepstrum\ F_0$
discard $\tilde{F}^n$ that are out of the F_0^{min} to F_0^{max} range
$K' \leftarrow$ number of remaining candidates $\tilde{F}^n$
number of "distinctive" candidates $D^n \leftarrow 0$

Algorithm 4 The BaNa F_0 Detection Algorithm (Continued)

while $\exists \tilde{F}^n \neq null$ **do**
  for k =1 to K' **do**
    if $\tilde{F}_k^n \neq null$ **then**
      num. of "close" candidates of $\tilde{F}_k^n$: $U_k \leftarrow 0$
      for $l = 1$ to K' **do**
        if $\mid \tilde{F}_l^n - \tilde{F}_k^n \mid \leq \xi$ Hz **then**
          $U_k \leftarrow U_k + 1$
        end if
      end for
    end if
  end for
  $D^n \leftarrow D^n + 1$, $V_D^n \leftarrow \max U_k$
  "distinctive" candidate $\check{F}_{D^n}^n \leftarrow \tilde{F}^n$ with max U_k
  $\tilde{F}^n$ with max $U_k \leftarrow null$
  all "close" candidates of $\tilde{F}^n$ with max $U_k \leftarrow null$
end while
*// **For all frames within a voiced segment:***
*// **Choose** F_0 **from "distinctive" candidates***
for n =1 to number of frames N_{frame} **do**
  for i, j =1 to D^n **do**
    $Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \mid \log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}} \mid + w \times \frac{1}{V_i^n}$
  end for
end for
return $\{p_n\}$ of $\min\{PathCost\} \leftarrow Viterbi\,(Cost)$, where path $\{p_n\}$ denotes F_0 for all frames

For each frame, the time complexity to calculate K F_0 candidates by calculating frequency ratios of p selected peaks is $\mathrm{O}(p^2)$. The time complexity to calculate D 'distinctive' candidates from K' remaining candidates is $\mathrm{O}\left(K'^3\right)$, which is the most complex process. The time complexity to use the Viterbi algorithm to choose the final F_0 from 'distinctive' candidates is $\mathrm{O}(D^2)$.

3.4 Experimental Settings for BaNa F_0 Detection For Speech

In this section, we present the speech and noise databases we use for F_0 detection performance evaluation, the error measurement metric, and parameter tuning of the proposed algorithm.

3.4.1 Speech and Noise Databases

Noisy speech samples can be generated by adding noise recorded in noisy environments to clean speech samples. Using this approach, the ground-truth F_0 values can be obtained from the clean speech. An alternative approach is to use speech samples directly recorded in real noisy environments, such as the SPEECON database [88], where additive noise, reverberations, and channel distortions are present. The ground-truth F_0 values in the SPEECON database are derived by manually F_0-marked recordings from a close speaking microphone with relatively little noise (clean speech). Several F_0 detection algorithms use the SPEECON database to evaluate their performance [89] [90] [91].

In this work, we use noisy speech samples generated from clean speech and different types of additive noise.

The clean speech samples we use are taken from four English speech databases: LDC [2], Arctic [92], CSTR [4], and KEELE [5]. Since female speakers normally

have higher F_0 values than male speakers, approximately an equal number of speech samples from male and female speakers are chosen from these databases. Also, since the frequency characteristics in speech differ from person to person, we select speech samples from all the available speakers within these databases. Table 3.1 presents the specifications of these speech databases.

Table 3.1: Evaluated speech databases and their features. Parameters are tuned using samples from the Arctic database.

Speech databases	**Emotion**	**# of speakers**	**# of selected samples**	**% of voiced frames**	**Has F_0 ground truth?**
Arctic [92]	neutral	4	10	54.2%	No
LDC [2]	various	7	20	50.4%	No
CSTR [4]	neutral	2	100	50.3%	Yes
KEELE [5]	neutral	10	10	50.4%	Yes

The LDC database is the Emotional Prosody Speech and Transcripts Database from Linguistic Data Consortium. It is chosen because it includes speech samples with strong emotions such as hot anger and elation, for which the F_0 values may change dramatically even within a short utterance. In the BaNa algorithm, the difference of F_0 values for neighboring frames is taken into consideration by the Viterbi algorithm. Therefore, the LDC database helps to investigate whether this discontinuity in F_0 values may influence the performance. The Arctic, CSTR and KEELE databases all contain speech samples with neutral emotion. All the speech samples used for the evaluation are included in the BaNa toolkit [93].

To test the noise resilience of the investigated algorithms, eight types of noises are added to the original signals with different SNR levels. The noise database

we use is the NOISEX-92 noise database [94], in which we choose six different types of real life background noise: speech babble (labeled as *babble* in the figures for performance comparison), destroyer engine room noise (*engine*), destroyer operations room noise (*operation*), factory floor noise ($factory$), vehicle interior noise (*vehicle*), high frequency radio channel noise ($highfreq$), as well as two common types of noise: white noise (*white*) and pink noise (*pink*). To generate noisy speech with a certain SNR value, the signal energy is calculated only on the voiced part, and the noise is amplified or attenuated to a certain level to meet the target SNR value.

3.4.2 Error Measurement Metric

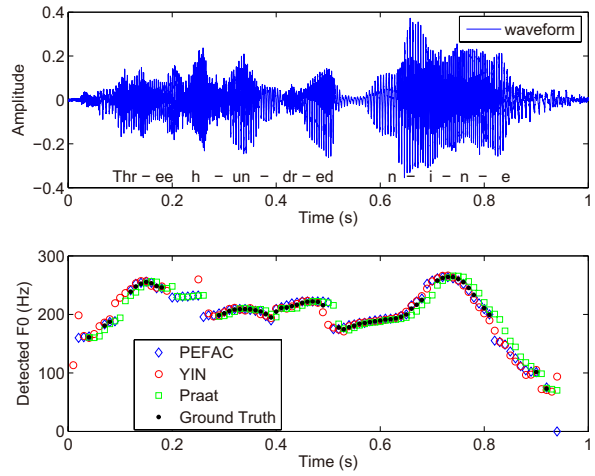
For the noisy speech data, if the detected F_0 deviates more than 10% from the ground truth value, it is counted as a gross pitch error. Otherwise, it is counted as a fine pitch error. The Praat algorithm also uses the 10% deviation range in their error measurement in [70]. Gross Pitch Error (GPE) rate is the percentage of incorrectly detected F_0 values in voiced speech segments. GPE rate has been widely used as the error measurement metric for F_0 detection [81] [83] [95]. Mean and standard deviation of Fine Pitch Errors (FPE) are also used in this study. FPE is calculated by the relative deviation of the detected F_0 from the ground truth F_0, with the unit in percent, for any pitch that does not represent a Gross Pitch Error [96] [97].

The F_0 ground truth values for the CSTR and KEELE databases are provided, which are obtained from the simultaneously recorded laryngograph signals. We downloaded the speech data and the ground truth values for the CSTR and KEELE databases from the SAFE toolkit [98], and then shifted the ground truth values in time as needed to line up with the F_0 detected by all the algorithms tested. For the LDC and Arctic databases with no F_0 ground truth provided, we use auto-labeled F_0 values of the original clean speech as the ground truth F_0

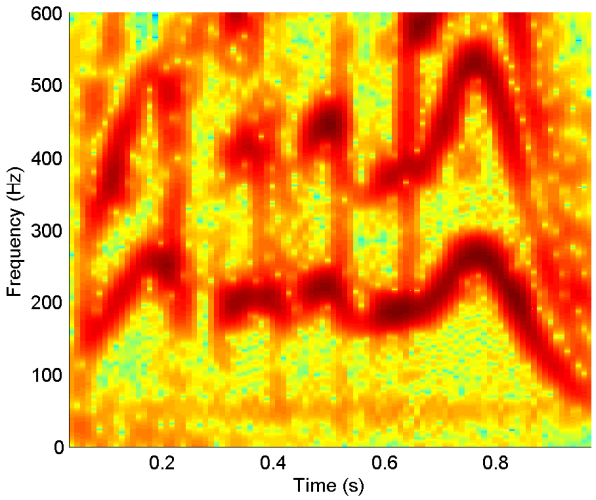
values and the voiced/unvoiced delineation, since the original speech samples are clean and with very little background noise. To best estimate the ground truth F_0 values, we calculate the detected F_0 values of three algorithms: PEFAC, YIN and Praat, which all perform well in F_0 detection for clean speech. For one frame, if the detected F_0 values from all three algorithms are within 10%, we assume that this frame is voiced, and the auto-labeled ground truth is determined by averaging the three detected F_0 values. Otherwise, we assume that the frame is unvoiced and do not detect F_0 for that frame.

Fig. 3(a) shows an example of a clean speech recording of the utterance 'three hundred (and) nine' along with the auto-labeled F_0 values as the ground truth. The word 'and' in the middle is skipped and is not spoken. We can see that for most of this clean utterance, the detected F_0 values from the three algorithms are very close. We use black solid dots to represent the ground truth F_0 values, which are calculated by averaging the detected F_0 values from PEFAC, YIN and Praat. We also note that the detected F_0 values from these three algorithms differ at frames corresponding to unvoiced stop consonants, i.e., 'th' in 'three' and 'h' in 'hundred', and discontinuities, i.e., the spaces between two words. Those frames are regarded as unvoiced and are ignored. For some frames, no F_0 value is shown on the plot for Praat, since Praat has its own voiced/unvoiced frame detection, and those frames are considered as unvoiced by Praat. The corresponding spectrogram is shown in Fig. 3(b), in which the lowest dark red curve verifies the calculated F_0 ground truth in Fig. 3.3a. The frame length used to compute the spectrogram is 60 ms.

The MATLAB code for the BaNa algorithm is available on the University of Rochester Wireless Communications and Networking Group's website [93]. Although the voiced/unvoiced speech detection is not within the scope of this chapter, we provide one version of the MATLAB implementation of the BaNa



(a)



(b)

Figure 3.3: For one clean speech utterance: a) speech waveform and the auto-labeled ground truth F_0 derived from three algorithms: PEFAC, YIN, and Praat, and b) the spectrogram. The frame length used to compute the spectrogram is 60 ms.

algorithm with an automatic voice marker [93]. The voiced/unvoiced speech detector used in this version of the BaNa code is the one implemented in [76] as the voiced/unvoiced speech detector for the Cepstrum F_0 detection algorithm. Frames with a dominant cepstrum peak, with an amplitude higher than the amplitude of the second highest peak by a certain threshold, are considered as voiced frames. However, we have not evaluated the performance of this voiced/unvoiced speech detector on noisy speech. Other voiced/unvoiced speech detectors are also available in the literature [83] [89].

3.4.3 Parameter Tuning

The frame shift is set to 10 ms in order to obtain smooth F_0 detection results. The absolute value of the Fourier transform of the Hann windowed speech signal is calculated, with the FFT size set to $2^{16} = 65{,}536$ to provide good frequency resolution. Candidates that are within $\xi = 10$ Hz of each other are considered to be "close" candidates. Since the F_0 of human speech is normally higher than 50 Hz and can be as high as 600 Hz for children or female voices [99], we set the lower limit and the upper limit for F_0 of human speech to be $F_0^{min} = 50$ Hz and $F_0^{max} = 600$ Hz, respectively.

There are several parameters in the BaNa algorithm that can be pre-tuned to achieve a more accurate estimate of F_0. The Arctic samples are used for the tuning of these parameters, and the set of parameters that provides the lowest GPE rate averaged over all levels of noise and all types of the NOISEX-92 noise [94] is chosen as the parameter set used in this chapter.

The parameter settings are shown in Table 3.2. To obtain a stable estimate of F_0, the frame length is chosen to be at least three times the F_0 period. Since the minimum F_0 we consider for both speech and music is 50 Hz, the frame length is thus $1/50 \times 3 = 0.06$ s, i.e., 60 ms. We also list in Table 3.2 other frame length

values that we have tested. Using the 20 ms frame length, which is one F_0 period at 50 Hz, results in a higher GPE rate. Although using the 90 ms frame length can slightly reduce the GPE rate, the temporal resolution is sacrificed.

Parameters in the spectral peak selection process are also tuned, including the number of spectral peaks p chosen to calculate the F_0 candidates, the spectral peak amplitude threshold and the threshold of the window width for smoothing, which is the width of the smoothing function applied before spectral peak detection. With these parameters being properly set, spectral peaks with low amplitudes and small widths are not chosen. We tested the performance of BaNa by choosing more or fewer spectral peaks, which means possibly more or fewer harmonics, but we found that choosing 5 peaks provides good F_0 detection performance. Also, choosing more spectral peaks increases the complexity in calculating the F_0 candidates.

Other parameters that are tuned are the tolerance range for the harmonic ratios used in the left table of Fig. 3.2, and the weight parameter used in the cost function in (3.3). Note that these parameters represent the optimal set across all noise types and SNR values for the Arctic speech database; they may not be optimal for a given noise type or SNR value or samples from other databases. A user could, of course, optimize the parameters for specific noise conditions, but we will show in Section 3.5 that using these tuned parameters provides good performance without the need for tuning for specific noise environments. Note that for all the other F_0 detection algorithms, we choose their default parameters in the evaluation.

To evaluate the parameter sensitivity of the BaNa algorithm on new types of noise, we use another widely-used noise database [3] with eight types of common ambient noise, including airport, babble, car, exhibition, restaurant, street, subway, and train noise. This noise database was used to construct the AURORA

Table 3.2: Optimal values of tuned parameters, and other values of the parameters for which BaNa algorithm is tested.

Parameters	**Optimal value**	**Other values tested**
Frame length	60 ms	20 ms, 90 ms
Number of chosen spectral peaks p	5	3, 4, 6, 7
Spectral peak amplitude threshold in peak selection	1/15 of the highest peak	1/25, 1/20, 1/10 of the highest peak
Window width for smoothing in the frequency domain in peak selection	50 Hz	40 Hz, 60 Hz, 70 Hz, 80 Hz
Tolerance range for harmonic ratios	Numbers shown in Table I	Narrowed range, extended range
Weight w in the cost function in (3.3)	0.4	0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9

noisy speech database [100] for speech recognition. Note that the AURORA noise database is only used for this parameter sensitivity test. All the remaining performance evaluations in this chapter are performed on noisy speech and noisy music generated using noise samples from the NOISEX-92 noise database [94].

We compare the performance of BaNa on the LDC database [89] by using 1) the set of parameters provided in the chapter, that are tuned on the Arctic database and the NOISEX-92 noise database [94], and 2) the parameter sets that are individually optimized on a specific type of noise from the the AURORA noise database [3] that yields the lowest GPE rates for the LDC database, averaged over 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR values.

As shown in Fig. 3.4, the difference in the performance when using the in-

dividually optimized parameter sets and when using the parameter set selected in the chapter is relatively small for most noise types. These results show that the performance of BaNa is not very sensitive to the specific parameters chosen. Thus, we can trade a slight drop in the GPE performance of BaNa for the benefit of not needing to optimize the parameters for a specific type of noise environment.

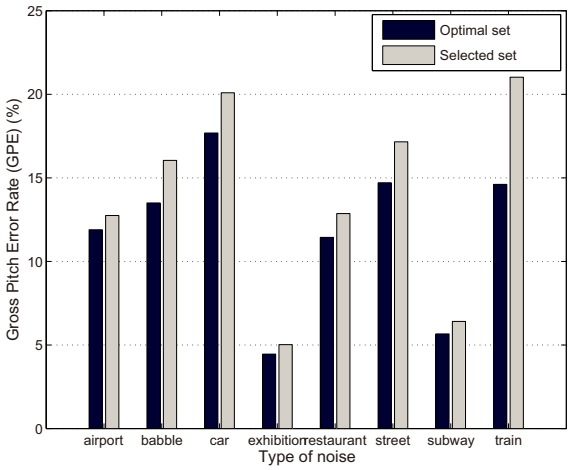


Figure 3.4: GPE rates of BaNa for the LDC database [2] with eight types of AURORA noise [3] averaged over all SNR values, using individually optimized parameter sets that provide the lowest GPE rates for a specific type of AURORA noise, and using the tuned parameter set selected in the chapter. Detected F_0 deviating more than 10% from ground truth are errors.

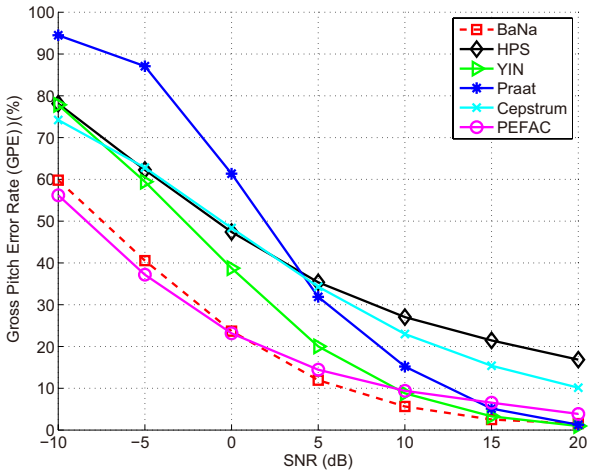


Figure 3.5: GPE rate of the different algorithms for the LDC database [2], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

3.5 F_0 Detection Performance For Speech Signals

In this section, we compare the F_0 detection performance of the proposed BaNa algorithm with that of several classic and state-of-the-art algorithms on speech signals in various noisy environments and for a wide range of SNR values. Seven algorithms are considered due to their popularity or good performance: YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu. These algorithms have been described in Section 3.2. The source code for YIN, Praat, Cepstrum, PEFAC, SAFE, and Wu are from [101], [102], [76], [103], [98], and [104], respectively. We implement the HPS algorithm based on the algorithm described in [69]. F_0 detection in eight different types of noise environments are evaluated, where noisy speech samples are generated by adding background noise to clean real speech samples with different noise power levels to achieve different SNR values.

Note that in our study, we only detect F_0 when only one speaker is speaking

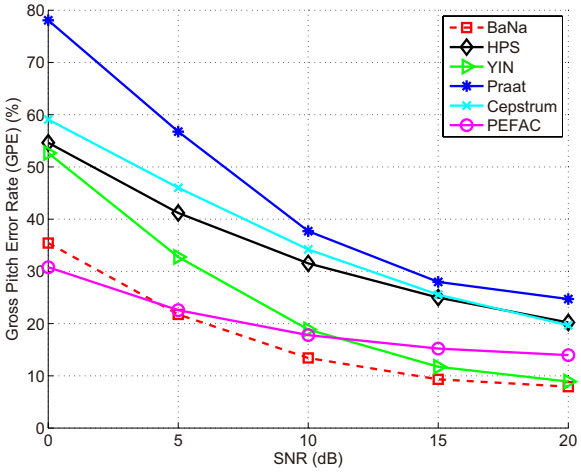


Figure 3.6: GPE rate of the different algorithms for the CSTR database [4], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

or only one instrument is played. If multiple people are speaking or multiple instruments are played at the same time, multiple F_0 values coexist. Multiple F_0 detection, as studied in work such as [105] [106] [107] [108], is not within the research scope of this work.

3.5.1 F_0 Detection Performance for Speech

We test all the F_0 detection algorithms on each one of the speech databases mentioned in Section 3.4.1, except the Arctic database, which was used for tuning the BaNa parameters. The GPE rate is evaluated as a function of SNR value, where the GPE rate is averaged over all types of noise for each SNR value.

For the LDC database with emotional utterances, Fig. 3.5 depicts the results, which shows that the BaNa algorithm achieves the best F_0 detection accuracy, i.e., the lowest GPE rate, among all of the algorithms for 0 dB SNR and above 0 dB SNR. PEFAC performs slightly better than BaNa at -5 dB SNR and -10 dB SNR.

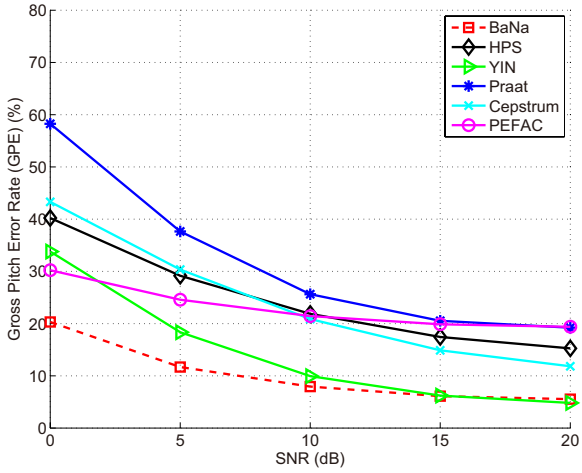


Figure 3.7: GPE rate of the different algorithms for the KEELE database [5], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

BaNa achieves the lowest GPE rate of 20.6%, which is obtained by averaging over -10 dB, -5 dB, 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR levels. Similar to the BaNa algorithm, the HPS algorithm is also based on the ratios of the potential harmonics. However, in real speech, the harmonics are not integer multiples of F_0, which may greatly affect the F_0 detection performance. We can also see that the BaNa algorithm has a very high resilience to severe noise, as it only wrongly detects 23.7% of F_0 values with noise at 0 dB SNR.

For a more stringent evaluation, we have also tested all algorithms on the LDC database using the GPE rate with a 5% deviation range. BaNa performs slightly better than PEFAC for above 5 dB SNR, while PEFAC performs slightly better than BaNa for below 5 dB SNR. The GPE rate for BaNa with a 5% deviation range is 30% at 0 dB, averaged over all 8 types of noise. The mean and standard deviation of Fine Pitch Errors (FPE) are also evaluated, using a 10% deviation range. The mean and standard deviation of FPE for BaNa are both 1.9% at 0 dB, which are only about 0.5% higher than the mean and standard deviation of FPE

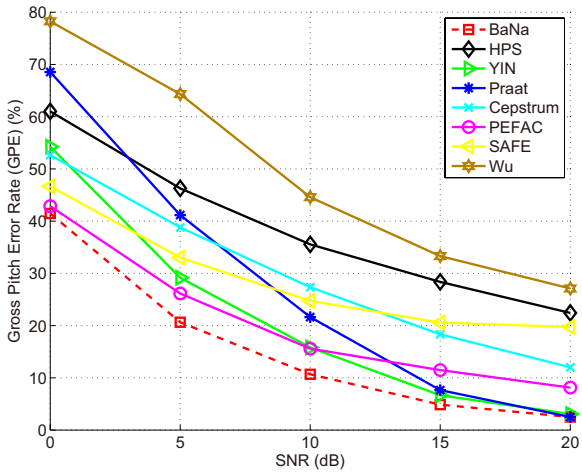


Figure 3.8: GPE rate of the different algorithms for the LDC database [2] for speech with babble noise. Detected F_0 deviating more than 10% from ground truth are errors.

for PEFAC and HPS.

Since the SAFE algorithm is only trained to detect F_0 for speech with babble noise and white noise, we show its performance for these two types of noise at the end of this section, where we also present Wu's results, since it is unclear how to run Wu's code on long speech samples. Therefore, we only test the Wu algorithm for the LDC database. Since the -10 dB SNR and -5 dB SNR scenarios are very severe noisy environments, we present the rest of the F_0 detection results for noise conditions with SNR greater than or equal to 0 dB.

The GPE rates for speech with neutral emotion are shown in Figs. 3.6 and 3.7 for the CSTR and KEELE databases, respectively. Similar results are obtained for the proposed BaNa algorithm and the five other algorithms, with the main difference being that PEFAC at 0 dB SNR performs slightly better than BaNa for the CSTR database. With noise at 0 dB SNR, the GPE rate of BaNa is 35.4% for the CSTR database, and 20.3% for the KEELE database. However, since

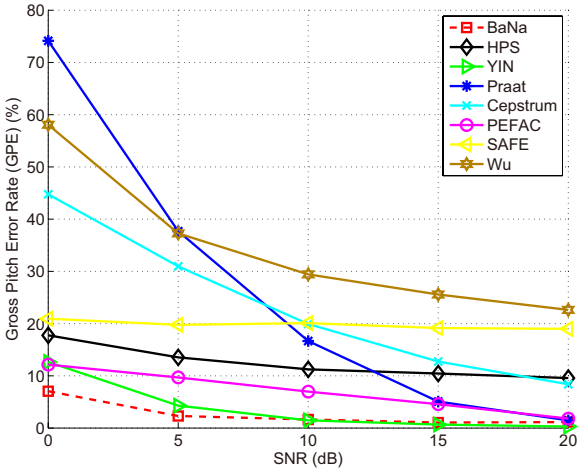


Figure 3.9: GPE rate of the different algorithms for the LDC database [2] for speech with white noise. Detected F_0 deviating more than 10% from ground truth are errors.

the ground truth F_0 values for the CSTR and KEELE databases are based on the laryngograph signals, we checked the ground truth values for a few speech samples and found that there are many spikes and discontinuities in the ground truth F_0 values found by using the laryngograph, especially on the boundaries of voiced and unvoiced frames. We can see from Figs. 3.6 and 3.7 that even at 20 dB SNR, the lowest GPE rate for all algorithms is still greater than 5%. While the ground truth for these databases may include several unvoiced frames and less reliable data, we present these results for the CSTR and KEELE databases in Figs. 3.6 and 3.7 in order to facilitate comparison with other F_0 detection algorithms that use these databases.

Babble noise and white noise are the most common types of noise in speech processing. Since the SAFE algorithm is only trained on babble noise and white noise, we only compare the results of SAFE for these two types of noisy speech. The KEELE database is used for training of SAFE, as in [83], and the LDC database is used for testing. We also show the performance of the Wu algorithm

proposed in [82]. The detected F_0 value is considered to be an error if it deviates more than 10% from the ground truth value, and again we use GPE rate as the error measurement metric. Figs. 3.8 and 3.9 present the GPE rate of the different algorithms for the LDC database for speech with babble noise and white noise, respectively. We can see that the F_0 detection for speech with babble noise is more difficult than F_0 detection for speech with white noise. Results show that BaNa, YIN, and PEFAC provide the lowest GPE rate for F_0 detection for speech with babble and white noise.

Speech with noise at 0 dB SNR is a challenging scenario for F_0 detection. For a head to head comparison, we present the performances of the BaNa algorithm and the closest competing algorithms, PEFAC and YIN, using the LDC database for eight different types of noise at 0 dB SNR in Fig. 3.10. We can see that BaNa has the lowest GPE rate for four out of eight types of noise. For the babble noise, which is a very common type of noise in real life scenarios, the BaNa algorithm achieves a 41.5% GPE rate compared with PEFAC's 42.9% and YIN's 54.3%, even when the speech is only slightly audible by the human ear. We can also see from Fig. 3.10 that the babble noise and the destroyer operations noise cause the worst degradation in the F_0 detection performance. By investigating the spectrum of several noisy speech samples, we found that the high spectral peaks of these two types of noise concentrate in the same frequency range as the spectral peaks of speech. On the other hand, the high spectral peaks of high frequency noise, vehicle noise and white noise are distributed in the frequency range, which is quite different from the spectrum of human speech, making it easier to differentiate speech spectral peaks from noise spectral peaks. Therefore, the GPE rate for speech with these types of noise remains at a relatively low level even at 0 dB SNR.

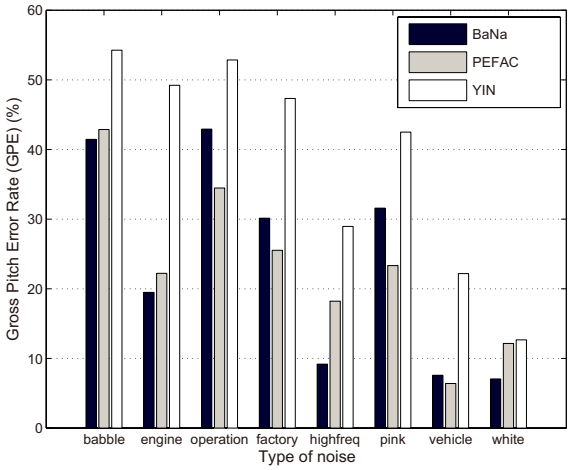


Figure 3.10: GPE rate of BaNa, PEFAC and YIN for the LDC database [2] with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 10% from ground truth are errors.

3.5.2 Breakdown Analysis of the BaNa Algorithm

As we can see from the above F_0 detection performance for speech, the proposed BaNa algorithm has the most advantage at 0 dB SNR across almost all speech databases. To provide additional insights to understand the core design of this noise-resilient algorithm, as well as the differences between BaNa and other algorithms, we provide a breakdown analysis of BaNa here:

- BaNa only considers the frequency ratios among the lower-order harmonics, and also Cepstrum is included as one of the F_0 candidates, thus BaNa is less affected by octave errors than Schroeder's frequency histogram.

- Harmonic summation methods use the amplitudes of spectral peaks to weight the frequency histogram, which is not a noise-resilient approach, since noise peaks with high amplitudes are likely to be chosen as F_0 after the harmonic summation. The BaNa algorithm, on the other hand, only uses the peak amplitude information to choose the spectral peaks, but the F_0 candidates

calculation is solely based on the frequency ratios of the chosen peaks. No peak amplitude information is used at this point, as it may be severely corrupted by noise.

- By providing a tolerance range for these frequency ratios, our algorithm is able to combat the frequency drift of harmonics and shape distortions of harmonic peaks caused by the noise.

- Post-processing using the Viterbi algorithm in BaNa considers the F_0 continuity, which helps to choose the F_0 candidates more accurately.

- Since the F_0 candidates calculated from peak frequency ratios are only based on lower-order harmonics, adding the Cepstrum as an additional candidate helps to capture the general period information for all spectral peaks.

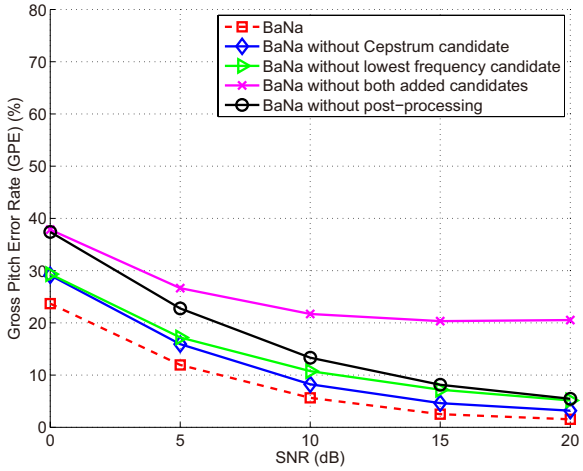


Figure 3.11: GPE rate of BaNa, BaNa without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database, averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

To show the effectiveness of using the Cepstrum candidate and the spectral peak with the lowest frequency as two additional F_0 candidates, and using the Viterbi post-processing, in Fig. 3.11 we plot the GPE rates for the BaNa algorithm, BaNa without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database. BaNa without post-processing means that we choose the F_0 candidate with the highest confidence score to be F_0 for each frame. We can see that using the two added candidates and using post-processing are effective to reduce the GPE rate. We can see that the GPE rate is as high as 20% when SNR is 20 dB without using both added candidates. This is because for some frames, only the F_0 peak's amplitude is high enough to be detected. Therefore, no F_0 candidates are derived from calculating frequency ratios.

By comparing the results for BaNa without post-processing with the results in Fig. 3.11 for the two algorithms that have no post-processing, HPS and Cepstrum, with the results in Fig. 3.5, we can see that BaNa without post-processing still achieves a lower GPE rate. Thus, from the breakdown analysis we conclude that the post-processing is helpful, but it is not the most critical step in determining the performance of BaNa.

3.6 BaNa F_0 Detection Algorithm for Music

In this section, we extend the BaNa algorithm to enable F_0 detection of music signals in noisy environments.

3.6.1 Modifications on BaNa for F_0 Detection for Music

Since speech and music have different frequency characteristics, the BaNa algorithm needs to be slightly modified for F_0 detection in music. In Section 3.3.2,

when detecting F_0 for speech, the p peaks with the lowest frequencies are selected. However, music signals can have high F_0 values, thus the low frequency region can be dominated by noise peaks. Thus, if we still choose the p peaks with the lowest frequencies, noise peaks are chosen incorrectly. Therefore, for music F_0 detection, we select the p peaks with the highest amplitudes in the frequency range considered. We show the benefit of this change in Section 3.6.4.

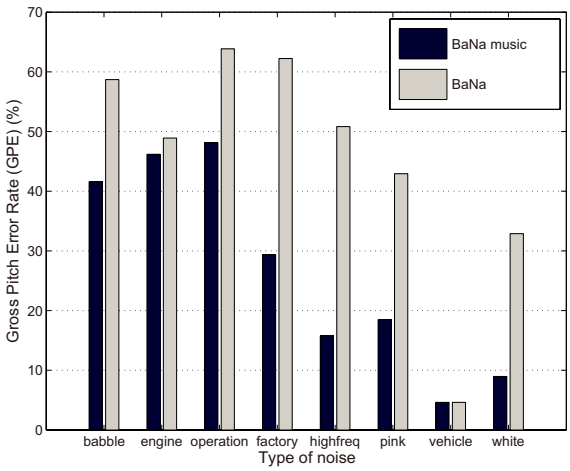


Figure 3.12: GPE rate of BaNa and BaNa music for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

3.6.2 Experimental Settings for F_0 Detection for Music

Due to the variety of spectrum characteristics for different musical instruments, to show the performance of the F_0 detection algorithms for musical instruments, samples from four instruments are used: violin, trumpet, clarinet and piano. These music pieces are selected and downloaded from [109], which were all recorded in a quiet environment. These music pieces include a piece of 3.7 s long violin with

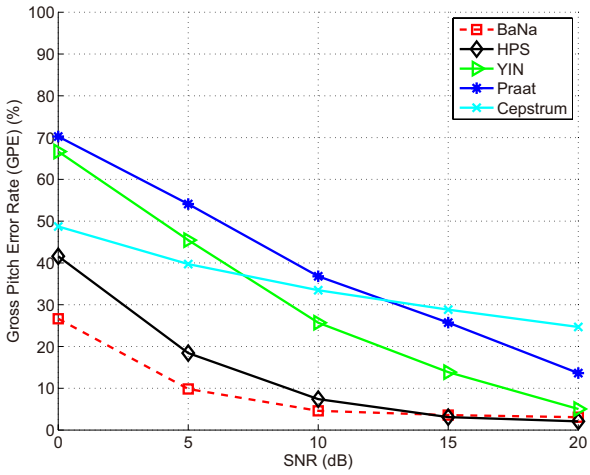


Figure 3.13: GPE rate of the different algorithms for a piece of violin music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

9 notes, a piece of 12.9 s long trumpet with 12 notes, a piece of 5.3 s long clarinet with 4 notes, and a piece of 7.8 s long piano with 8 notes. All the music samples used are also included in the BaNa toolkit [93]. The additive noise is from the same noise database as in Section 3.4.1.

For F_0 detection in music, we use hand-labeled ground truth F_0 values, which are determined by manually inspecting the spectrum and locating the F_0 peaks for each frame. Due to the large F_0 range in music, we use a more stringent F_0 deviation criteria for error measurement. The difference between two neighboring key frequencies is $2^{\frac{1}{12}}$, which is approximately 6%. Thus, we use half of this number, i.e., 3%, as the F_0 deviation criteria, which is also called the musical quarter tone [110]. Thus, detected F_0 values that deviate more than 3% from the ground truth values are counted as errors. This error measurement metric is also used by other studies [110].

3.6.3 Parameter Tuning

According to the music F_0 range specified in [67], the lower and the upper limit for F_0 of music are set to $F_0^{min} = 50$ Hz and $F_0^{max} = 4{,}000$ Hz, respectively. It is set to 50-4,000 Hz for these competing algorithms as well for a fair comparison. The other parameters are the same as those in Table 3.2, and are not further optimized using music signals.

3.6.4 BaNa vs. BaNa Music

To show the effectiveness of the changes made to the BaNa algorithm to be suitable for F_0 detection in music, we plot the GPE rate in Fig. 3.12 for a piece of violin music using both the original BaNa algorithm and the customized BaNa music algorithm with eight different types of noise at 0 dB SNR. The F_0 detection range is set to be the same for the original BaNa algorithm and the customized BaNa music algorithm, i.e., $F_0^{min} = 50$ Hz and $F_0^{max} = 4{,}000$ Hz. We can see that the modifications in the BaNa algorithm for music F_0 detection are necessary, and can greatly reduce the GPE rate for almost all types of noisy music. Note that throughout this section, we just use 'BaNa' to represent the BaNa music algorithm.

3.6.5 F_0 Detection Performance for Music Signals

In this set of experiments, we compare the BaNa algorithm with other algorithms for music F_0 detection. Within the evaluations of the SAFE algorithm in [83], there are no detection results for music. Therefore, we are not able to run the SAFE algorithm here due to the lack of noisy music training data. Also, according

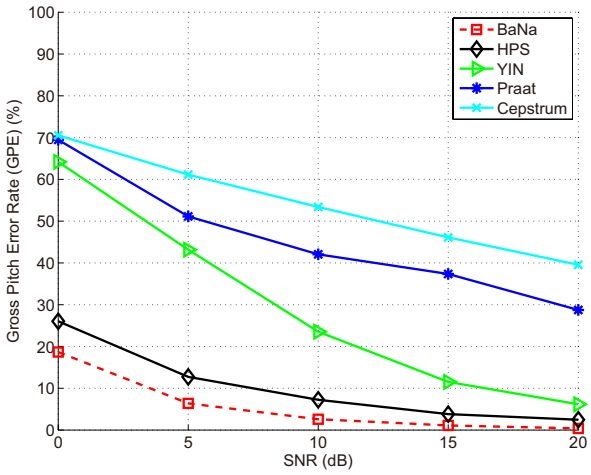


Figure 3.14: GPE rate of the different algorithms for a piece of trumpet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

to the authors of PEFAC [79], PEFAC is not suitable for F_0 detection in music, hence we do not include that here. Also, it is unclear how to use the code for the Wu algorithm [82] to process long audio samples. Therefore, we only compare the proposed BaNa algorithm with YIN, HPS, Praat, and Cepstrum. Figs. 3.13-3.16 show the GPE rates of the different algorithms for violin, trumpet, clarinet, and piano, respectively, averaged over the eight types of noise. Results on all these four instruments show that the BaNa algorithm achieves the lowest GPE rate among all the algorithms. At 0 dB SNR, BaNa achieves the lowest GPE rates, which are 36.1%, 28.1%, 58.3%, and 35.3% lower than the closest performing algorithm, HPS, for violin, trumpet, clarinet, and piano, respectively.

From the above results, we can see that BaNa, HPS, and YIN provide the overall best F_0 detection performance in noisy music. Praat and Cepstrum do not provide consistent or satisfying results. Therefore, we choose BaNa, YIN, and HPS for detailed comparison using the violin piece with eight different types of noise at 0 dB SNR. In Fig. 3.17 we can see that BaNa has the lowest GPE rate

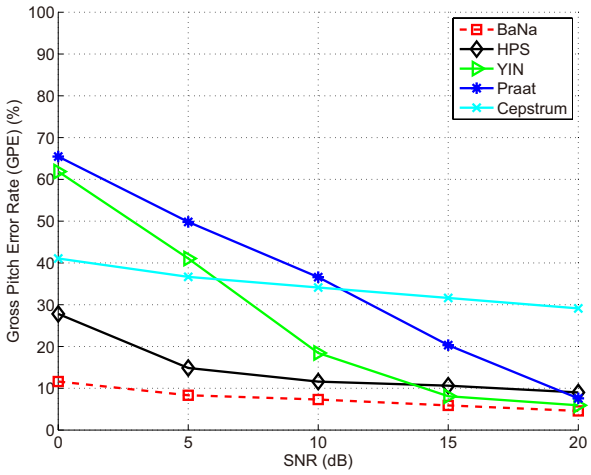


Figure 3.15: GPE rate of the different algorithms for a piece of clarinet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

for seven out of eight types of noise, especially for the speech babble noise.

3.7 Implementation Issues

With an increasing number of speech-related smartphone apps emerging in the market, and due to the fact that speech captured by smartphones are usually affected by different types of noise, it is important to discuss the challenges in implementing the BaNa F_0 detection algorithm on a mobile platform. To explore these issues, we implemented BaNa as an app on an Android platform[1]. Since the F_0 candidates and their confidence scores can be calculated separately for each frame, as explained in Section 3.3.2, we can take advantage of multithreading to speed up the implementation. Single-core and multi-core devices can both benefit from multithreading through an increased utilization of the processor(s). When all threads finish the calculation of F_0 candidates for their own assigned frames,

[1]Code for the Android implementation of BaNa is available at [93].

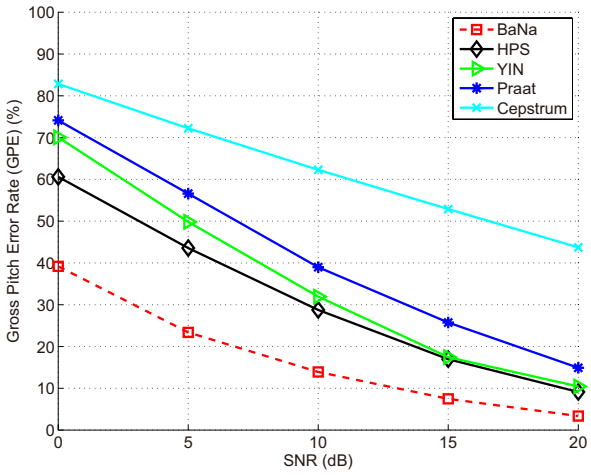


Figure 3.16: GPE rate of the different algorithms for a piece of piano music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

the Viterbi post-processing can go through all the frames to determine F_0 for each frame.

To test the speed of the BaNa F_0 detection implementation, we ran tests with different parameter settings and speech sample lengths on a Google Nexus 7. The specs of the device are: Nvidia Tegra 3 quad-core processor clocked at 1.2GHz, 1GB of RAM. Of course, the speed of the algorithm highly depends on the capabilities of the mobile device. Table 3.3 shows the elapsed time to process a 1.3 s long speech sample with sampling rate of 22,050 Hz. All the parameters for the BaNa algorithm are set to be the same as those in Table 3.2. For a more reliable measurement, the elapsed time for each test is averaged over 10 trials. We can see that the BaNa F_0 detection algorithm runs roughly 8 times faster by using the 2^{13} FFT size than using the 2^{16} FFT size, though using the 2^{13} FFT size still provides a reasonable frequency resolution of $22,050/2^{13} = 2.7$ Hz per sample. Also, we can see that multithreading helps to further reduce the elapsed

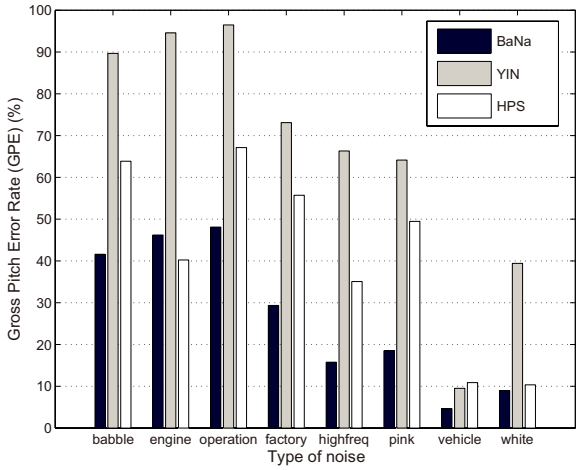


Figure 3.17: GPE rate of BaNa, YIN and HPS for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

time.

We show in Table 3.4 the elapsed time for F_0 detection for speech samples with different lengths. For this test, we choose the setting that provides the fastest speed, i.e., the number of threads is set to 4, and the FFT size is set to 2^{13}. These results show the possibility to turn the BaNa algorithm into a real-time F_0 detector even on mobile devices.

3.8 Conclusions

In this chapter, we presented BaNa, a noise resilient hybrid F_0 detection algorithm for speech and music. BaNa was designed to detect F_0 in a noisy environment, for example on a smartphone. This would enable the wide deployment of speech-based applications, such as the ones that use emotion detection. Evaluations show that BaNa achieves the lowest GPE rate for most cases among the algorithms

Table 3.3: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform with a different number of threads and FFT sizes. The speech file is 1.3 s long.

Number of threads	FFT size			
	2^{16}	2^{15}	2^{14}	2^{13}
1	11.05	5.16	2.52	1.42
2	6.85	3.15	1.49	0.85
3	5.93	2.67	1.28	0.92
4	5.89	2.67	1.25	0.80

Table 3.4: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform for speech samples with different lengths.

Number of threads	FFT size	Length of speech sample (s)				
		2	4	6	8	10
4	2^{13}	0.91	1.61	2.39	3.05	3.82

investigated from the literature including YIN, HPS, Praat, Cepstrum, PEFAC, SAFE and Wu for different types of background noise, and under different SNR levels from -10 dB to 20 dB. Even for the very noisy scenario of 0 dB SNR, the GPE rate of BaNa averaged over all types of noise is only about 20% to 35% for speech for the different databases evaluated. The GPE rate for music at 0 dB SNR is 12% to 39% for different instrument pieces. Additionally, we implemented the BaNa algorithm on an Android platform, and implementation issues such as delay and multithreading are discussed. Tests on a real device show that the implementation is fast enough to provide for real-time F_0 detection applications in the future.

4 Mobile Cloud Computing - A Survey

4.1 Introduction

While energy efficiency of the devices themselves and analysis of data in noisy environments are both important features of energy efficient sensing and computing systems, where to perform the computation of data analysis is also an important design consideration. Computing can be performed locally on the node, or, for more intense applications, computing can be off-loaded to external computing resources, such as cloud-based resources. Recent developments in mobile computing have truly empowered human users, as mobile computing can augment cognitive capabilities dramatically, e.g., through voice recognition, natural language processing, machine learning, augmented reality, and decision-making [1]. With recent advances in mobile devices, coupled with the technological advances in wireless and cloud technologies, computationally intensive applications such as the emotion classification application described in Chapter 1 may run on devices with limited resources such as tablets, netbooks and smartphones using the cloud remotely as an additional computational resource.

Although different definitions exist in the literature [111,112], we define mobile-cloud computing as the co-execution of a mobile application within the expanded

mobile/cloud computational platforms to optimize an objective function. A typical objective function is the application response time, where the goal is to minimize the objective function. Expanding the application computational resources beyond the mobile is necessary for applications where the objective function cannot be minimized sufficiently by the mobile platform alone (e.g., real-time face recognition), as well as for applications that rely on data not stored on the mobile device. In mobile-cloud computing, it is crucial to provide the user seamless, transparent and cost-effective services as mobile devices rent computing, storage, and network resources from the cloud in order to process and store a vast amount of data [113–115].

We define application cost as an example objective function that quantifies the fees charged by Cloud operators, such as Amazon Web Services, during the execution of the application. For example, Amazon charges for compute-usage per hour per CPU instance, which implies increasing application costs as the required amount of computation increases. Similarly, cloud operators charge for the usage of database instances, such as Microsoft SQL Server. Table 4.1 shows some example mobile-cloud applications and their computational/storage demands, as well as their application response-time sensitivity. While applications requiring higher computational and storage resources might cost more during operation in a Cloud platform such as AWS, certain response-time sensitive applications, such as the Battlefield application described in Table 1, might tolerate this increased cost due to their need for low response time. Notice that Cloud operators charge less for compute-resources with lower response time guarantees. Specifically, while AWS charges nothing for Micro instances with no response time guarantees, it charges a small amount for the Small instance, and significantly higher for the Large instance, which is a dedicated CPU instance. By the preparation of this document, the AWS pricing for these instances ranged from $0.10 to $0.40 per core per GHz per hour [113], where the unit price decreased with a higher core-count commit-

ment (i.e., number of cores available to an instance). This implies a rich variety of options when executing mobile-cloud applications. The choice of the Cloud CPU instances depends on the application priorities listed in Table 4.1.

Table 4.1: Cloud-based applications and their resource requirements. Each application has a significantly different response time requirement and resource utilization tolerance to reduce costs while still keeping the functionality within expected bounds.

Application	**Description**	**Database Size**	**Compute Resources**	**Time Sensitivity**
Battlefield	Assist soldiers in the battlefield through real-time object recognition	HIGH	HIGH	HIGH
Natural Language Processing	Perform real-time speaker or speech recognition	LOW	MEDIUM	MEDIUM
Airport	Conduct real-time face recognition of known criminals	HIGH	HIGH	HIGH
Fire Fighting	Assist fire fighters with disaster in real time	MEDIUM	MEDIUM	MEDIUM
Medicine	Accelerate medical research (e.g., recognizing DNA sequences in real time from a microscope while research is in progress)	HIGH	MEDIUM	MEDIUM
Archeology	Recognize archeological structures in real time while researchers are at the archeological site	HIGH	LOW	LOW
Surgery	Recognize diseases (e.g., tumors) in real-time from a cloud-based database while surgery is in progress	HIGH	MEDIUM	HIGH
Amber Alert	Identify criminals by searching through the FBI database for match to a photo taken by cameras	LOW	LOW	MEDIUM
Social Network	Profiling online users by searching through the database for marketing purposes	HIGH	LOW	LOW

The primary focus of this chapter is to elaborate on the techniques that enable these mobile-cloud applications to achieve the goals listed in Table 4.1. Although the demands of these applications will not change from that shown in this table,

achieving certain goals might never become possible by using mobile-only or even a mobile-cloud combination. This is due to the limited computation and storage on a mobile device, which does not permit the processing or storage of large amounts of data locally, as well as the high network latencies connecting the mobile and cloud, placing a lower bound on application response times when utilizing the cloud for processing and storage of large amounts of data. Later in this chapter, we will describe how the required application response times may be achieved by using an edge-server device called a "cloudlet", creating a mobile-cloudlet-cloud platform.

This chapter is organized as follows. In Section 4.2, the technological challenges and the state-of-the-art in computational and storage capabilities of mobile devices and the network latencies are studied. Issues related to energy efficiency and security are also explored, followed by a brief study of the aforementioned intermediate layer cloudlet and its function in the mobile-cloud computing environment. Existing architectural designs as well as performance enhancement techniques proposed in the literature for mobile-cloud as well as mobile-cloudlet-cloud computing are surveyed in Section 4.3 and Section 4.4. Section 4.5 concludes the chapter with discussions on future research areas.

4.2 Technological Challenges in Mobile-Cloud Computing

Running the resource-intensive applications enumerated in Table 4.1 far exceeds the capabilities of today's mobile devices. The constraints on mobile devices in terms of weight, size, battery life, ergonomics, and heat dissipation limit the resources available in mobile hardware, including the processor speed, memory size, and storage capacity. Given these challenges, mobile computing benefits tremen-

dously when combined with cloud computing that can offer virtually limitless computing power and storage space, as well as access to up-to-date databases, only available in the cloud.

There are, however, several technical obstacles to enabling mobile devices to benefit from cloud computing resources, including the compute capability and storage capacity available at the mobile, network connectivity and latency challenges, the need for energy-efficiency at the mobile device, and security concerns. As each one of these constraints affect mobile-cloud computing in a unique way, they will be individually detailed in the following subsections.

4.2.1 Compute Capability and Storage Capacity

Despite an order of magnitude higher computational power of today's mobile devices compared to the ones from just a few years ago, the relative computational power ratio of a non-mobile and a mobile device is likely to stay approximately the same in the foreseeable future. This is due to the architectural and technological state-of-the-art advances being applied to mobile platforms as well as non-mobile platforms simultaneously by different market leaders such as Intel for desktop platforms and ARM for mobile platforms. The most important metric, computer-power-per-Watt (also defined as GFLOPS-per-Watt) has almost reached equal levels in both mobile and desktop platforms. For example, a Tegra3-based mobile phone incorporating an ARM CPU and an Nvidia GPU at the core can deliver approximately 10 GFLOPS/Watt [116]. Alternatively, a desktop platform composed of an INTEL Core i7 CPU and an Nvidia Geforce 600 GPU [117] nearly has the same power efficiency metric, delivering around 10 GFLOPS/Watt compute power. This is due to the significant recent advancements in mobile processors: almost every power efficiency technique employed in desktop CPUs is now being incorporated into mobile CPUs, with the most important being the ability to ar-

chitect the CPU with multiple cores, which is known to have a dramatic energy reduction advantage [118].

The storage technology is slightly different in that the widespread use of Solid State Disks (SSDs) allowed mobile devices to be built with storage capacities that are currently around 64GB to 128GB. This is currently an order of magnitude less than that for desktop platforms, which enjoy inexpensive hard disks in the multiple-TB range. This means that mobile-cloud applications that require significant local data storage in the mobile are not feasible.

4.2.2 Network Connectivity

A primary concern in the use of mobile-cloud computing is the non-negligible latency over the WAN (Wide-Area Network) between the mobile and the cloud, which hurts the user experience in mobile-cloud computing. Interactive applications that constantly engage the users are likely to suffer the most from long delay, jitter and jerky and sluggish processing. Studies [1] show that the quality of client performance becomes highly variable with long latency.

In order to measure latencies over WAN connections, we ran a simple program that sends ping packets from a client computer to servers in cloud datacenters in January and February 2012. The client computer was located in Rochester, New York, in the United States, and we used the five datacenters available in AWS [113], which are all located in geographically different regions, namely in Virginia and Oregon in the United States, Ireland in Europe, Sao Paulo in South America, and Singapore in Asia.

Table 4.2 and Table 4.3 show the mean and standard deviation of these latencies for the AWS datacenters when being accessed by the client computer from wired and wireless networks. This data clearly indicates the challenges in running a mobile-cloud application that uses the AWS datacenters as cloud servers. The

Table 4.2: Average and standard deviation of latencies over wired connections (in ms).

Measuring Time	Weekend				
Datacenter	VA	OR	Ireland	Sao Paolo	Singapore
Mean	122	322	294	389	580
Std Deviation	124	525	201	166	242
Measuring Time	Weekday				
Datacenter	VA	OR	Ireland	Sao Paolo	Singapore
Mean	42	223	196	389	546
Std Deviation	18	41	25	166	34

response time of such an application will be lower-bounded by the mean latency of the communication to the datacenter it is using as cloud servers. Alternatively, the predictability of the response time will be determined by the standard deviation of the latency. While there have been significant improvements in network throughputs over the past decade, allowing users to enjoy such high-speed connections with 50 Mbps downstream bandwidth (e.g., DOCSIS3 cable standard [119]), network latencies have not improved nearly at the same rate.

Although the network latencies might improve in the future, this is expected to be at a much slower pace, potentially keeping the latencies observed in Table 4.2 and Table 4.3 approximately the same in the foreseeable future. For example, an application requiring a 200 ms response time (close to what can be described as *real-time*) is not a candidate to run on cloud servers residing across international boundaries with 300 ms to 1100 ms latencies. This presents a dilemma in mobile applications in that the particular emphasis should be placed on latency, not

Table 4.3: Average and standard deviation of latencies over wireless connections (in ms).

Wireless Connections	Wi-Fi				
Datacenter	VA	OR	Ireland	Sao Paolo	Singapore
Mean	253	389	293	434	697
Std Deviation	470	635	520	704	1278
Wireless Connections	3G				
Datacenter	VA	OR	Ireland	Sao Paolo	Singapore
Mean	930	817	798	872	1061
Std Deviation	595	710	915	1079	2060

the throughput, when developing an application. Alternatively, any intermediate device, such as the cloudlet that will be described later, should be targeted to reduce the negative impact of this high latency.

4.2.3 Power and Energy Consumption

Today's mobile devices incorporate significantly sophisticated power management circuitry [116,120]. This, combined with power consumption demands that change drastically in sudden peaks imply a sophisticated power consumption pattern from the mobile device based on the activities being performed (i.e., talk, compute, or run applications). Power consumption could change between sudden peaks of mW and a few W. While the power consumption is in fact an irrelevant measure in terms of battery life, the energy consumption is the relevant measure in determining the battery life.

To quantify the utility of mobile-cloud computing, one must take into account the energy demands of computation and communication separately. Analyzing

these two activities separately will shed light onto the balance that must be maintained between computation and communication via efficient scheduling. In the following two subsections, we study the power and energy demands of these two activities.

Computation Power and Energy Consumption

Since the computation energy is the only relevant metric for determining battery life, we analyze the amount of energy required to execute an identical task in a desktop and mobile platform. The energy efficiency metric, defined as GFLOPS-per-Watt describes how many Watts of power is consumed to while delivering 1 GFLOPS of computational output. This metric is 10 GLOPFS/Watt in a modern mobile processor such as Tegra 3 [116], while it is almost in the same range for a modern CPU/GPU-based desktop computer [117]. Alternatively, a mobile device operates at around one Watt average power consumption, whereas a desktop platform could reach 200-1000 Watts of power consumption.

Although a given computational task (e.g., face recognition) may consume more power and execute in a shorter amount of time on a non-mobile platform (e.g., PC), it will consume less power on a mobile platform (strictly due to the aforementioned constraint on peak power), and is, therefore, expected to complete in a longer time period. However, due to the fact that approximately the same amount of computational energy is required for the same task, the mobile platform will take two to three orders-of-magnitude longer to execute the same task, since its peak power output is nearly two to three orders-of-magnitude lower.

Communication Power and Energy Consumption

The energy efficiency of Wi-Fi and 3G are significantly different. As the data provided by [121] indicates, 3G requires much higher energy levels due to its

inability to transfer large amounts of data, while Wi-Fi can transfer almost an order-of-magnitude more data within the same power envelope. According to their experiments, 3G connections require 2,762 mJ per 100KB (i.e., 27.62 $\mu J/B$). Alternatively, Wi-Fi requires around 5 to 10 $\mu J/B$, making it more energy efficient. These different energy profiles suggest that, when determining optimum algorithms that partition computation and communication, the energy patterns of both must be considered. As an example, assume that an algorithm has a choice among different computation vs. communication options as presented below:

Case 1: Front-loading In this case, most of the computation is done on the Tegra 3 mobile device that has a 10 GFLOPS compute-capability, and a 10 GFLOPS/W energy efficiency, and 3G is used for communication. For the Case 1 algorithm, 20 GFLOP of computation and 100KB of data transfer are necessary. Total energy consumption is 2,000 mJ (i.e., 20 GFLOPS / (10 GFLOPS/Watt) = 2 Watt * 1 second = 2J = 2,000 mJ) for the computation (based on the aforementioned 10 GFLOPS/W for Tegra 3) and 2,762 mJ for the 3G communication (according to the data from [121]), yielding a total energy demand of 4,762 mJ for the entire task.

Case 2: Back-loading Assume now that the same algorithm can be modified to perform more of the computation in the cloud at the expense of increased data transfer via 3G. In this Case 2, the computation in the mobile will be halved to 10 GFLOP at the expense of doubled data transfer size to 200 KB. This will create a compute-energy demand of 1,000 mJ and a communication demand of 5,524 mJ, resulting in a total energy demand of 6,524 mJ, clearly an unfavorable choice.

Case 3: Back-loading on a faster communications link Back-loading on a faster communications link: If we consider Case 2 on a faster Wi-Fi link (e.g., 576 mJ/100KB as shown in Figure 1), the energy demand for the front-loading

and back-loading cases are (2,000+576=2,576 mJ) and (1,000+1,152=2,152 mJ), respectively, making back-loading a better alternative for faster Wi-Fi links, although the decision is the reverse for 3G links.

4.2.4 Security

When considering outsourcing the computational tasks of a mobile application to the cloud, an important issue arises for certain applications: the security of the data being transmitted/received by the application. Depending on the application, the security of the data carries a varying importance. For example, for online games being played by a few gamers through the mobile device, the security factor is negligible, while for remote health assistance applications, it is of utmost importance.

Due to the emergence of applications using wireless sensors with built-in low-power microcontrollers and the sudden spike in interest for concepts such as Internet-of-Things [122], the security of the data being transported by the application has become one of the most important concepts to consider. Additionally, emerging tele-medicine applications also emphasize the importance of security and data privacy [123] within the mobile-cloud platforms [124–126]. The concern for data security is of great importance for mobile devices with powerful processors with a 1 W power budget (e.g., NVIDIA Tegra3), but it is particularly challenging for embedded processors with only a mW power envelope, such as the Microchip 32-bit microcontroller family [127]. This is due to the fact that the encryption of the data using standard Advanced Encryption Standard (AES) encryption [128,129] is compute-intensive and strains the computational resources of the underlying computational platform unless specialized crypto-accelerators are used. However, as most of today's devices have AES hardware acceleration built-in, the encryption time and energy is typically less than the transmission energy

Table 4.4: The major differences between the cloudlet and the conventional cloud [1].

	Cloudlet	Cloud
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24/7 operator
Environment	"Datacenter in a box" at business premises	Machine room with power conditioning and cooling
Ownership	Decentralized ownership by local business	Centralized ownership by Amazon, Yahoo, etc.
Network	LAN latency/bandwidth	Internet latency/bandwidth
Sharing	Few users at a time	Hundreds to thousands of users at a time

of the data, thereby making encryption widely available with minimal impact on cost and energy.

4.2.5 Cloud and Cloudlet: Addressing these Challenges

With the help of the cloud, mobile devices may be able to offload the computationally-intensive parts of their applications. The enormous resources of the cloud may minimize the time and energy cost of the mobile applications on those computations. However, as described in the Microsoft MAUI project (Cuervo et al., 2010), some applications might never be feasible from mobile devices, due to the high latency mobile-cloud connection. Adding a cloudlet, a local device that provides 100 to 1000 times higher computational power than the mobile device with minimal latencies, creates possibilities for running latency sensitive and computationally-intensive applications from a mobile device [1]. The notion of a cloudlet was introduced as a means to overcome some of the technical obstacles described above. The main idea is to provide the abundant resources needed at mobile devices not from distance clouds, but from a nearby cloudlet.

As Satyanarayanan et al. point out, the key differences between the cloudlet and a conventional cloud are listed in Table 4.4. Note that soft state refers to cache copies of data or code that are available elsewhere (e.g., mobile device or the cloud), whereas hard state refers to sole copy of data or code. Since a cloudlet only contains soft state, the loss of a cloudlet is not catastrophic. Cloudlets allow offloading a portion of the tasks as well as a re-shaping of the network traffic by aggregating network packets. With proper task management algorithms, a cloudlet may be able to leverage the power of distant cloud servers to maximize the performance while minimizing the impact of long network latency.

In the following section, different approaches will be presented that use the cloudlet as a buffering layer to either speed up the computation or to reduce the negative effect of the communication latency to the cloud.

4.3 Architectural Design

As stated in the previous section, in mobile cloud computing, there is a clear need for a mechanism to handle the interoperations between the mobile device and the cloud servers in order to improve the performance. Depending on whether or not cloudlets are used, the current research on the design of mobile-cloud architectures can be categorized as cloudlet architectures and non-cloudlet architectures. Many of the technologies being used in non-cloudlet architectures can also be adapted to cloudlet architectures. In this section, we introduce the state-of-the-art for these mobile-cloud and mobile-cloudlet-cloud architectures.

4.3.1 Platforms Providing Cloud Services

The "cloud" may consist of commercial servers like the Amazon Web Services [113], Microsoft's Windows Azure [114], and the Google Cloud Platform [115], or

it may be created ad hoc from available computing resources, as shown through recent studies [130]. While ad hoc clouds have been conventionally created from high-end servers or desktop platforms, recently mobile platforms have been explored as a source for the computing resources.

For example, Hyrax [131] demonstrated the concept of using smartphones as a cloud of computing resources. Marinelli developed a mobile-cloud computing system named Hyrax by porting Hadoop Apache, an open-source implementation of MapReduce, to Android smartphones. Hyrax allows computing jobs to be executed on networked Android smartphones. However, the performance of Hyrax was poor compared with Hadoop on traditional servers, not only because the smartphones were much slower at that time, but also because Hadoop was not originally designed, nor optimized, for mobile devices.

The GEMCloud [24] we proposed in Chapter 5 is another example of using mobile devices to create an ad hoc cloud of computing resources. By utilizing distributed mobile devices to cooperatively accomplish large parallelizable computational tasks, we envision that such approaches can make use of the massive amount of idle computing power that is potentially available to the public. More importantly, we show that a mobile computing system like GEMCloud has significant advantages in energy efficiency over traditional desktop cloud servers when the overall system is considered, rather than each individual computational device (e.g., CPU and GPU). The details of the design and development of GEMCloud are described in Chapter 5.

Other examples of ad hoc cloud systems are NativeBOINC [132] and BOINC Mobile [133], Android platform (Android, 2012) equivalents of the BOINC volunteer computing platform originally designed for PCs and game console platforms [134]. Since the physical devices that build up the cloud determine the cloud's characteristics such as computing power, energy efficiency and network latency, it is important to profile the cloud servers and take this into account when

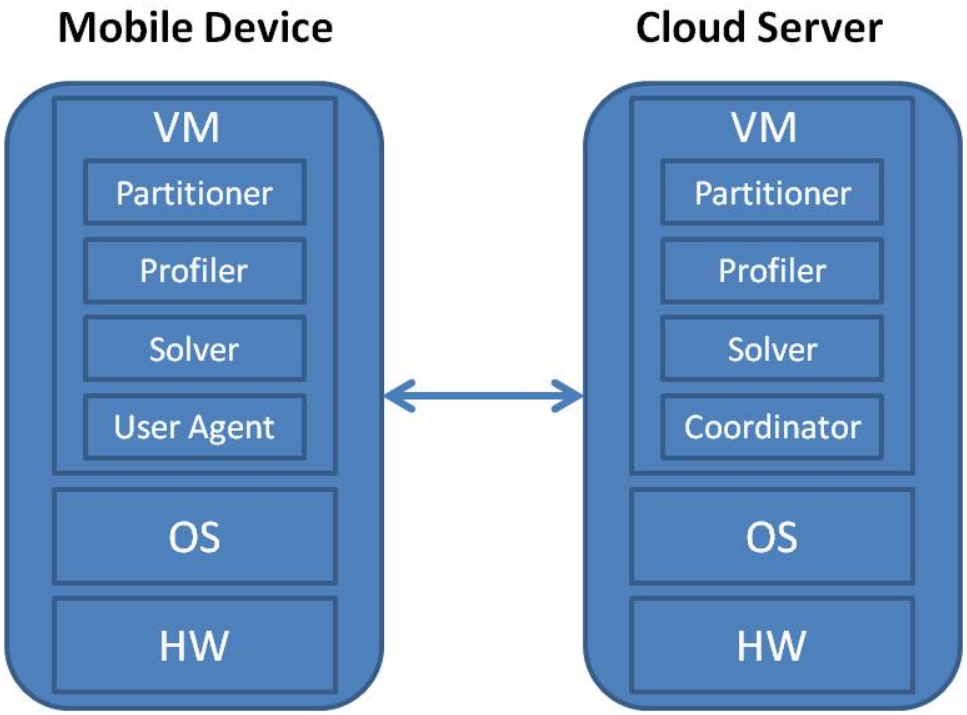


Figure 4.1: A generalized mobile-cloud architecture.

designing the mobile-cloud computing system.

4.3.2 Mobile-Cloud Architectures

Mobile-cloud computing has been investigated since shortly after the concept of cloud computing was introduced in mid-2007, and it has attracted great interest in the research community [111]. Some important implementations of mobile-cloud computing, including MAUI [121], CloneCloud [135, 136] and Virtual Smartphone over IP [137, 138] employ cloud servers to process application partitions offloaded by the mobile device. As discussed previously, the cloud servers may be located in a commercial cloud or an ad hoc cloud.

Although architectural details may vary in different mobile-cloud computing

implementations, some common components are often included on top of the operating system and hardware layers as shown in Fig. 4.1:

- A *Partitioner* that analyzes the application and determines which part(s) of the application can be offloaded to the cloud. Depending on the technique being used, the partitioning granularity may be application-level, thread-level, method-level or even line-of-code-level. For applications that cannot be partitioned, the *Partitioner* is not necessary.

- A *Profiler* that collects the mobile device's system measurements to identify the performance status, resource status and other contextual information. The performance measurements may include network condition (e.g., type of network being used, signal strength, bandwidth, computing power and response latency of various cloud services, etc.), screen brightness, CPU, memory and storage usage. The resource status may include remaining battery, available computing power (derived from CPU, memory usage), and the resources required by the application (or method, thread, depending on the granularity offered by the *Partitioner*). Other contextual status may include location, acceleration, temperature, date and time, etc.

- A *Solver* that gathers information from the *Partitioner* and the *Profiler* to decide how to offload the partitions to the cloud based on an optimization algorithm.

- A *User Agent* on the mobile device and a *Coordinator* on the server that handle the authentication and security. The *Coordinator* may also interact with the server database that stores the mobile device users' profiles (e.g., device specifications, user configurations, subscribed services, contextual data) and activity logs. The *Coordinator* allocates the resources on the cloud server for the mobile users.

In general, MAUI [121], CloneCloud [135, 136] and Virtual Smartphone over IP [137, 138] architectures all have the above components or components with similar functionalities.

4.3.3 Cloudlet Architectures

As described in Section 4.2, cloudlets can be used as intermediaries between mobile devices and cloud servers. One of the first implementations of a cloudlet architecture was a prototype mobile-cloudlet computing system named Kimberley [1], developed by Satyanarayanan et al. The authors envisioned a cloudlet as a "data center in a box" widely dispersed throughout the Internet. Unlike the cloud, the cloudlet is self-managed with decentralized ownership, maintains only soft states, is connected to the mobile over a LAN, not a WAN, and is accessed by only a few users at a time. As a proof-of-concept implementation, the Kimberley system utilizes a local server as a cloudlet to process application partitions. The authors show that in some cases, a local server is able to provide enough computing power to boost the execution speed of a mobile application, while in other cases, the computing resources provided by a local server may not be enough and a cloud has to be used to fulfill the computation and storage requirements of the mobile application. Figure 4.2 elaborates the cloudlet architecture in comparison with the direct mobile-cloud architecture. The major role of a cloudlet in a mobile-cloudlet-cloud network is task management. It may also help the mobile with some intermediate processing.

The MOCHA (MObile Cloud Hybrid Architecture) architecture was created as a solution to massively-parallelizable mobile-cloud applications [6, 139] by Soyata et al. In MOCHA, mobile devices such as smartphones, touchpads, and laptops are connected to the cloud via a cloudlet, a dedicated device designed from commodity hardware supporting multiple network connections such as 3G/4G, Bluetooth,

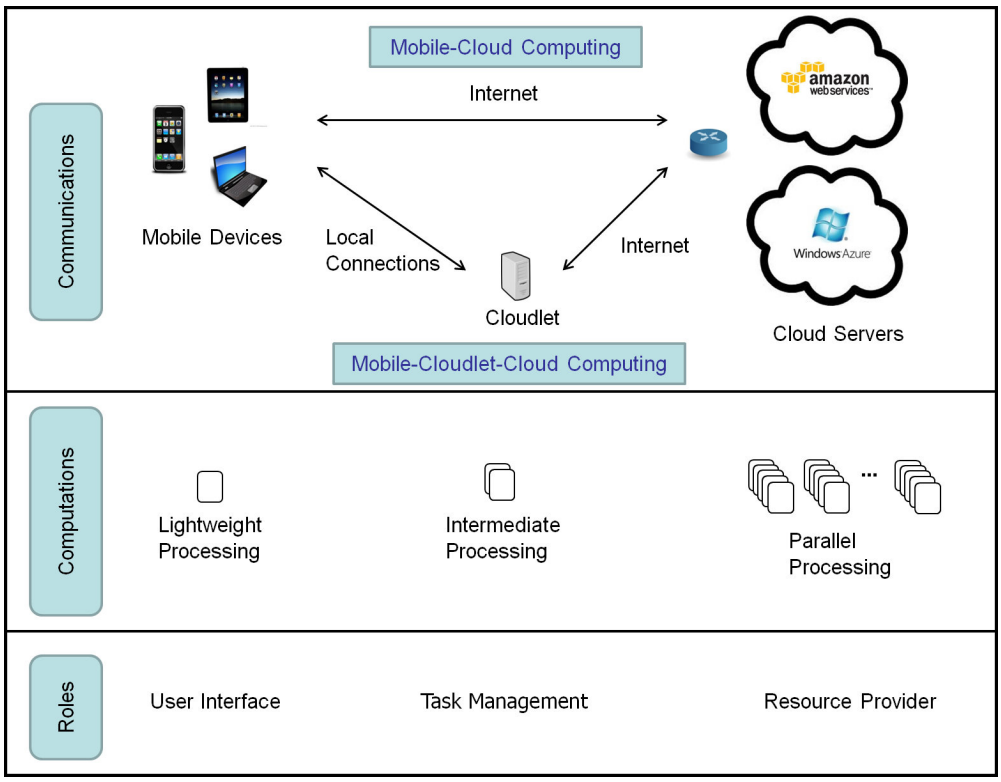


Figure 4.2: The Mobile-Cloud computing and Mobile-Cloudlet-Cloud computing architectures: mobile devices directly interact with a cloud or via the cloudlet and use dynamic partitioning to achieve their quality of service (QoS) goals (e.g., latency, cost).

and WiFi. The cloudlet determines how to partition the computation among itself and multiple servers in the cloud to optimize the overall quality of service (QoS) based on continuously updated statistics of the QoS metrics (e.g., latency, cost) over the different links/routes. The authors demonstrate the concept of MOCHA via a mobile-cloud application demanding real-time response, such as face recognition [6] and simulate the same architecture in a battlefield application where the response time is of primary importance [139].

Similar to the characteristics of cloudlets assumed by the Kimberley [1] and MOCHA [6] architectures, a two-level architecture is introduced by Ha et al. (Ha et al., 2012). This two-level architecture leverages both today's unmodified cloud infrastructure (Level 1) and a second level data center, named 1WiFi, at the edge of the Internet (Level 2), servicing currently-associated mobile devices. The Level 2 data centers are powerful, well-connected and safe cloudlets that only have cached soft state from Level 1 data centers or buffered data from mobile devices. Trust issues and speed of provisioning are the new challenges to this architecture and must be investigated before it can be widely deployed.

In some scenarios such as an office building, multiple cloudlets may be located closely to each other and may be connected in a peer-to-peer fashion. In this case, routing among the cloud servers, the cloudlets and the mobile devices has to be considered. In [140], Fesehaye et al. propose two types of routing schemes, namely distributed and centralized routing. In distributed routing, the routing table is constructed and maintained by the cloudlets. The cloudlets periodically broadcast their presence information to the neighboring nodes and the other cloudlets. When a mobile user hears a broadcast message from a cloudlet, it records the latest cloudlet ID into its cloudlet table. Each mobile user also periodically broadcasts its ID to let the cloudlet in range register this user and forward it to other cloudlets. In centralized routing, the central server is responsible for constructing and maintaining the routing table. The cloudlet periodically

sends the IDs of its mobile users, its own ID and its neighboring cloudlets' IDs to the central server. The central server then computes the routing table for each cloudlet and installs the forwarding tables into the cloudlets.

Simulations were conducted to evaluate the performance of this architecture. In the simulations, distributed routing was chosen as the routing scheme for the cloudlet architecture. The results show that the cloudlet-based approach has lower data transfer delay and higher content delivery throughput than the cloud-based approach. The results were under the assumption that the WiFi transmission range is larger than 250 m. Therefore, the author suggests using the latest technologies such as Flashlinq [141] or by using Wi-Fi repeaters to achieve a desired coverage. Since there are no performance comparisons with the centralized routing, questions still remain as to which routing scheme has better performance.

4.4 Task Management Among Mobile, Cloudlet, and Cloud

The goal of developing a cloudlet-assisted mobile-cloud computing system is to improve the performance (e.g., latency, energy efficiency, monetary cost) on the mobile device. One important approach for improving the performance is to offload partial or full execution of the application to the more resourceful cloudlet or the cloud [142]. Shifting the computation load to a communication load may lead to substantial gains in performance. The computation offloading approaches are based on virtual machine technologies and can be viewed as middleware designs. Besides the middleware that enables the code offloading, task distribution algorithms and control policies are needed to improve the performance to its best. In this section, we will introduce both the middleware designs and the task distribution algorithms that enhance the performance of mobile computing.

Computation Offloading Approaches Despite the advances in mobile device technologies, the processing and storage capabilities of mobile devices are still not comparable to those of servers (or the cloudlet) and will continue to lag in the near future. In order to run computationally-intensive applications, the mobile can offload some of the computation to servers while the mobile device computes only lightweight parts of the application. A virtual machine (VM) can support individual processes or a complete system running on flexible hardware platforms, thereby providing the feasibility to migrate partial or full applications from the mobile device to more powerful cloudlet/cloud servers without major modifications to the application. Therefore, the application processing time can be shortened while the energy consumption on the mobile device is reduced. Yet this approach poses several technical challenges. First, how can we identify and partition the compute-intensive or energy-hungry parts within the mobile application automatically? Second, what strategy should a mobile device employ for partitioning and offloading with the goal of minimizing computation time and maximizing energy savings? Third, how can we implement such a system from a practical point of view?

In this subsection, we provide an overview of the state-of-the-art VM-based techniques for mobile-cloudlet/cloud computing. These include 1) an approach employed by the Kimberley system [1] that demonstrates the feasibility of VM synthesis using VirtualBox, 2) an approach used in MAUI [121] that provides both full and fine-grained remote execution using the .Net framework, 3) the technique in CloneCloud [135, 136] that supports thread granularity partitioning using Dalvik VM, 4) an approach by Chen et al. [137, 138] that enables offloading on non-customized Android devices also using Dalvik VM, and 5) an OSGi approach by Verbelen et al. [143]. Table 4.5 compares these five approaches discussed in this subsection.

Table 4.5: Task Partitioning Approaches Comparison

Publication	Technologies	Platform	Granularity	Application Development Difficulty
Kimberly	VirtualBox	Linux	Application	Low
MAUI	.Net Framework	Windows	Method	Require application developer's annotations
CloneCloud	Dalvik VM	JavaVM supported	Thread	No annotation required
Chen's approach	Dalvik VM	JavaVM supported	Thread	No annotation required
Verbelen's approach	OSGi	JavaVM supported	Component	Require application developer's annotations

Virtual Box in Kimberly

Satyanarayanan et al. implemented a VM for the Kimberley architecture [1] prototype using a technique called dynamic Virtual Machine synthesis that employs transient cloudlet customization. A small VM overlay is delivered by a mobile device to the cloudlet infrastructure, which creates and launches the VM using a base VM plus the delivered VM for the application. The prototype was implemented on a Nokia N810 tablet running Maemo 4.0 Linux, and the cloudlet infrastructure was implemented using Ubuntu Linux. Kimberley uses VirtualBox as the VM manager and a tool called "Kimberlize" to create VM overlays and synthesize those overlays with base VMs to create a launchVM. Both the mobile and the cloudlet run the Kimberley Control Manager (KCM) to support the transient binding between themselves using a TCP tunnel established between these two KCMs.

The authors used VM overlay sizes and the speed of the synthesis process to evaluate the system performance. The VM overlay sizes were 100-200 MB for a collection of Linux applications. These sizes were an order of magnitude smaller than the full VM size (8 GB). The speed of synthesis ranged from 60 to 90 seconds. These results are acceptable for an unoptimized proof-of-concept prototype, and there is plenty of room for improvement through further optimization. For instance, a high-bandwidth short-range wireless network can reduce overlay transmission time, parallelism on the cloudlet can decrease decompression and overlay application times; caching as well as prefetching can be used to eliminate VM synthesis delays. The deployment challenges are also discussed, including 1) the business model (bottom-up versus top-down), 2) the sizing of cloudlets, i.e., how much processing power and storage capacity a cloudlet should provide, and 3) trust and security.

Remote Execution in MAUI

MAUI [121] was originally motivated by the assumption that battery technology will be a major bottleneck for the future growth of smartphones. MAUI consists of three main components. First, program partitioning uses the Microsoft .NET Common Language Runtime (CLR) to enable developers to annotate methods that may be performed remotely, to extract methods that may be performed remotely using reflection (Richter, 2010), and to identify the state of the application using type-safety and reflection. MAUI generates two proxies on both the mobile device and the server that handle control and data transfer to implement decisions on which methods to run remotely and which to run locally. Second, the MAUI profiler and solver will characterize the device and the program, then determine the methods to be executed remotely. On the server side, there is a MAUI coordinator handling the authentications and resource allocations.

The mobile part of MAUI was implemented on an HTC Fuze smartphone running Windows Mobile 6.5 with the .NET Compact Framework v3.5, and the MAUI server was implemented on a desktop with a dual-core 3 GHz CPU and 4 GB RAM running Windows 7 with the .NET Framework v3.5. The main results measure energy consumption and execution time for three applications-face recognition, 400 frames of a video game, and 30 moves in a chess game. The results show that using remote execution on MAUI saves 5-12 times the energy compared to the smartphone only case and reduces the execution time by more than a factor of 6.

CloneCloud Utilizing Dalvik VM

CloneCloud [135, 136] allows a smartphone to partially offload its application to the phone's clone in the cloud. It migrates a modified version of the original application executable to a virtual machine in the cloud. This algorithm allows thread

granularity migration, and therefore the User Interface (UI) or other essential components can remain to be executed at the mobile. Additionally, native methods can execute at both the mobile device and its clones in the cloud/cloudlet. One drawback of the CloneCloud approach is that local threads need to block unless they are independent from the migrated threads.

Chun et al. developed a dynamic profiler to analyze the execution time and energy cost of each method on a mobile device, which are then used by an optimization solver to decide which method(s) should be migrated to the clone. The profiler and optimization solver were implemented on a modified Dalvik VM on Android, and this requirement may limit the scope of its application. CloneCloud is tested on an unlocked HTC G1 Android phone and a server with a 3.0 GHz Xeon CPU running the Android x86 virtual machine via VMware ESX 4.1. Three applications - a virus scanner, image search, and privacy-preserving targeted advertising - were tested on the CloneCloud prototype. The results show that for the tested applications, when connecting to the CloneCloud via Wi-Fi, the execution time is shortened by 2.1x-20x and the energy consumption is reduced by 1.7x-20x. When connecting to the CloneCloud via 3G, the execution time is shortened by 1.2x-16x and the energy consumption is reduced by 0.8x-14x.

Virtual Smartphone Over IP Utilizing Dalvik VM

Chen et al. [137, 138] introduce a framework that allows heavy backend tasks on an Android phone to be offloaded to an Android virtual machine in the cloud. Unlike MAUI or CloneCloud, the authors built an Android OS on an x86 cloud server on which a virtual smartphone is executed. Two frameworks are proposed: the first framework [138] offloads an entire application to the virtual smartphone and controls the application through remote desktop sharing; the second one [137] offloads only the compute-intensive components to the cloud. The former offers heightened security and data leakage prevention as the entire application and

resulting data do not physically reside on the mobile, while the latter offers fast GUI responsiveness and offline execution.

The major advantages of using this approach over MAUI and CloneCloud are 1) no use of additional APIs in the source code is required, and 2) no modifications to the mobile device's OS or root access are required. Note that these features are useful for system deployment. To achieve the above features, the authors replace the AIDL (Android Interface Definition Language) tool with a helper tool so that the compiler automatically creates service wrappers that are offloaded to the cloud by a service offloader. Offloading decisions may be made according to the time and energy consumption required to perform a task. Once offloading is done, the user needs to wait until the task is completed before the service offloader re-evaluates the time and energy metrics.

OSGi Approach

In [143], Verbelen et al. introduced a different definition of a cloudlet. In their cloudlet architecture, the unit of deployment is a component. Components are managed by an Execution Environment that runs on top of an operating system (OS). The OS is installed on a node that is either virtualized or real hardware, and managed by a Node Agent. A cloudlet is a group of nodes (either mobile devices and PCs or elastic cloud servers) that are physically proximate to each other. A Cloudlet Agent optimizes the performance by deploying or configuring the components within the cloudlet.

The proposed cloudlet framework is implemented on top of the OSGi framework [144], allowing components to be installed, started, stopped, updated, and uninstalled without a reboot. The authors use an OSGi bundle named R-OSGi [145] to facilitate the distribution of components across different OSGi instances. In other words, the R-OSGi allows components to be executed on different platforms. The authors implement an augmented reality application to evaluate the

cloudlet framework. Results show that with components being offloaded to a local laptop computer, the application on the mobile device can be improved to satisfy the performance requirements. The experiment results also show that when the cloudlet is running in a distant cloud, the performance decreases to an unsatisfactory level due to the increased latency.

4.4.1 Other Middleware Designs

The above describes approaches that allow mobile devices to offload computational tasks to the cloudlet/cloud. In general, these approaches can be categorized as middleware that lies on top of the operating system and provide services to the applications. More specifically, the offloading approaches described above enable the communication and management of data and code between the client and the cloudlet/cloud. Besides supporting code migration, a middleware framework may also provide generic interfaces to handle the communication and input/output functions, which will facilitate the design of software for these mobile-cloud and mobile-cloudlet-cloud architectures.

One example of such a middleware framework design is given by Flores et al. [146]. In their paper, a generic middleware framework named Mobile Cloud Middleware (MCM) is introduced. MCM enables interoperability between the mobile and the cloud/cloudlet. In MCM, a mobile application first sends an HTTP or XMPP request to MCM, which processes this request and forwards the request to the MCM manager. An interoperability API engine within the manager then decides which API set to use to interact with the cloud/cloudlet. When the process running on the cloud/cloudlet is finished, a notification is sent to the mobile device using the push notification services (C2DM-Cloud to Device Messaging [147] for the Android platform and APNS-Apple Push Notification Service [148] for the iOS platform). This request-notification mechanism is processed asynchronously, so that the mobile device can perform other tasks while waiting for the notification.

For some applications, it is important for the cloud/cloudlet to have the capabilities to dynamically capture and utilize contextual information from mobile devices to improve QoS. Such contextual information may involve user profiles, session quality, network conditions and environmental conditions such as temperature, humidity, and location. In [149], Hoang et al. summarize the functions that context-aware middleware is expected to incorporate, including 1) intelligent monitored data analysis that preprocesses raw sensor data to improve its quality and to update context repositories, 2) network auto-switch that monitors network latency and automatically chooses the best network, and 3) energy consumption management that aims to minimize energy consumption at the device level, the communication level, or the collaborative level. This middleware layer constructs a communication bridge between the data acquisition layers on both the mobile and the cloud service ends.

4.4.2 Task Distribution Algorithms

With the support of middleware, the mobile devices are able to offload their computationally-intensive application components to one or multiple of the resource-rich cloudlets or cloud servers. In order to fully maximize the benefits of utilizing the cloud resources, task distribution algorithms must be developed.

In the MAUI approach, a MAUI profiler is used to estimate the characteristics of the device's energy consumption, the program's runtime and the resource needs, as well as the characteristics of the wireless network such as bandwidth, packet loss rates and delay. Then, the MAUI solver determines which methods can be remotely executed based on the information computed by the MAUI profiler. The solver uses integer linear programming (ILP) to solve an optimization problem whose objective function is to maximize the energy savings given constraints about latency penalty and methods that may be computed remotely. A similar profiler and a similar solver were used by CloneCloud to determine the migration point.

When the network connectivity is intermittent, extra latency will be introduced if the optimization algorithm uses the current communication condition to determine the migration point, such as being used in CloneCloud. In Cirrus Cloud [150], Shi et al. introduce an offloading algorithm that recursively chooses the optimal migration points from the root of the profile tree of an application. At every node within the tree, the algorithm computes the completion time to decide whether to execute the entire subtree locally, migrate it to the cloud entirely or migrate only parts of it. When migrating parts of the subtree, the same algorithm is iteratively applied to all the children of this node. With the computation and future network connectivity accurately known, the algorithm is able to find the optimal partitioning of the application and minimize the execution time. In reality, it is obvious that future network connectivity is not known ahead of time. However, using historical statistics may help to predict the connectivity and therefore achieve a close-to-optimal code migration.

The above approaches consider the code offloading from one mobile client to one server. In the cases when multiple servers may be used, the latency of each individual server must be considered. As indicated by Table 4.2, the response latencies of different cloud servers have significant variations. This diversity of connectivity creates the potential for gains through the smart selection of cloud servers for the offloading of computation. Motivated by this potential, the authors of MOCHA developed two task distribution algorithms, namely the fixed and the greedy algorithms, to optimize the mobile-cloud computing performance in terms of result response time [6].

The fixed algorithm is used to evenly distribute the pending tasks to the cloud servers (and the cloudlet if there is one). On the other hand, the greedy algorithm continuously sends the next pending task to the server (or cloudlet) that is able to return the result in a minimum amount of time. This process is repeated until all the pending tasks are assigned. The authors conduct Monte-Carlo simulations

to analyze the effects of using the fixed algorithm and the greedy algorithm on a mobile-cloud network with a cloudlet or without a cloudlet. In the simulations, a computational job consisting of 5 identical and independent tasks is distributed among available cloud servers (and a cloudlet) with varied processing capabilities and communication latencies. As shown in [6], the greedy algorithm reduces the overall response time by 50% over the fixed approach using the cloudlet, while only a 20% gain is achieved without the cloudlet. The results demonstrate the benefits of using the greedy task distribution algorithm as well as the benefits of using the MOCHA architecture.

While the above approaches all try to maximize the performance of the mobile device side, in [151], Hoang et al. introduce an admission control policy that stands on the cloud server's side. The authors propose an optimization model based on a semi-Markov decision process to maximize the reward (e.g., revenue of service provider) of the resource usage in the cloudlet under resource and bandwidth constraints while meeting the QoS requirements (i.e., mobile users' service requests accept rate). The optimization model is transformed into a linear programming model and can be easily solved by a standard linear program solver. In the paper, the authors consider that the offloaded application partitions will be processed in the cloudlet rather than forwarded to the cloud. Therefore, the bandwidth and resource limitations at the cloud are not included in the model. According to the model, the control policy decides whether the service request from a user should be accepted or blocked. In the performance evaluations, the authors assume a circumstance where two classes of users, i.e., members with higher priority and non-members with low priority, are using the cloudlet. Two services with different bandwidth and resource requirements are considered. The results show that using the proposed control policy, under the bandwidth and resource constraints, the cloudlet is able to satisfy the members' QoS requirements while maintaining high resource utilization rate.

4.5 Conclusions and Future Research Directions

In this chapter, we provided an extensive survey of the state-of-the-art mobile-cloud computing techniques, some of which utilize cloudlets as the middle layer. We provided a summary of the existing architectural designs and compared different approaches that enhance application performance via cloud-based execution. We also highlighted the research and technological challenges in different approaches presented in the literature. While much work has been done to date, mobile-cloud computing is still in its early research stages. Especially, the cloudlet is a new topic in the cloud computing world. Before these mostly theoretical proposals for the cloudlet find their place in practical applications, many research challenges have to be overcome. In this section, we summarize the most important challenges.

- **Cloudlet design.** Two main components of the cloudlet are its hardware architecture and software management mechanism. In the literature, many envision to extend a Wi-Fi access point into a more intelligent machine equipped with cloudlet functionalities [1, 6, 139, 140, 152]. To support this vision, a determination must be made as to what is a reasonable amount of compute power and storage capacity that can be incorporated into the cloudlet without exceeding the power consumption and equipment cost constraints. For example, a cloudlet that costs as much as a desktop PC and consumes as much power is unlikely to be adopted by the masses. Alternatively, a cloudlet that does not have sufficient compute power will not augment the mobile devices' capabilities enough to make an impact on the overall performance. Therefore, ideal cloudlet architecture parameters lie between these two extremes. Such questions are closely related to the deployment strategy that centers on the business model with incentives.

  The primary questions regarding software are 1) support for a variety of

applications, 2) self-managing environments, and 3) efficient resource management. The system software environment should be generic enough so that different kinds of applications can execute without major modifications; the cloudlet resources should be managed automatically with minimal human involvement; and the resource (processing, storage, and networks) usage should be optimized so that cloudlet computation can support as many applications as possible at a given time and the overall execution time can be minimized. We envision cloudlets incorporating modern processors, such as GPUs [117].

- **Task assignment.** Current implementations such as MAUI [121] and CloneCloud [135] have utilized offloading algorithms, where the code in the mobile device runs in a virtual machine (VM) and the execution can be migrated between the mobile and the cloud in real-time. However, there still remain many techniques that can be explored for further performance improvements by migrating the execution across multiple cloud servers, pipelining the transmission of application partitions to hide the transmission delay, and caching the reusable partitions to reduce the transmission load. As discussed previously, multiple cloud service providers or ad hoc cloud servers may be employed for computational or storage resources. This increases the complexity of the task distribution problem. Although the authors of MOCHA consider the computation power and network latency of different cloud servers when assigning the tasks [6, 139], it is necessary to develop more generic task distribution algorithms that take into consideration the resources and the constraints of the mobile devices, cloudlets and the cloud servers. Using a more comprehensive cost model, such as the one shown in Figure 4, is needed to develop better dynamic optimization algorithms to further enhance the performance and robustness. With sufficient computation power, a cloudlet is a proper candidate to optimize the task distribution

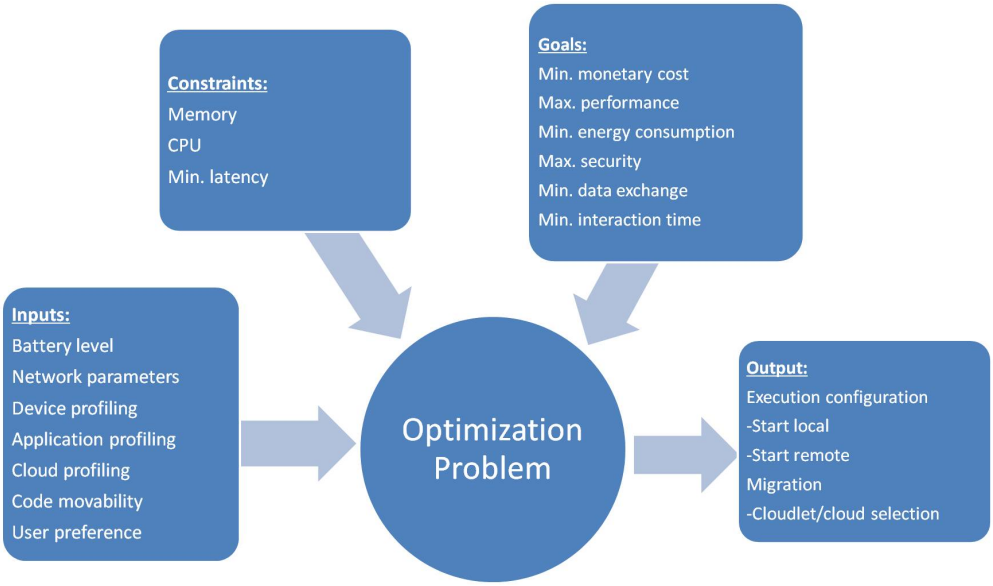


Figure 4.3: The cost model of mobile cloud computing (adapted from [7, 8]).

decision dynamically.

- **Security and privacy.** As many mobile devices and the cloudlet/cloud collaborate and share data, security and privacy is always an important issue. While WPA2 [153] and IPsec [154] provide layer-2 encryption of the data, layer-6 encryption is still a requirement for some applications. For example, layer-6 encryption is critical for pharmaceutical applications such as those involving bioinformatics or computational chemistry that are executed remotely on rented/commercial cloud platforms [113–115]. Homomorphic encryption can allow the computation to be performed without ever decrypting the data, providing additional layers of security. Future work is required to determine how layer-6 encryption, including homomorphic encryption, can be applied when passing data between the mobile, cloudlet and cloud.

- **Energy efficiency.** As more hand-held mobile devices are equipped with sensing capabilities, collaborative sensing applications have become a reality. These applications often require thousands of participating smartphones that do opportunistic sensing with little user involvement. Since this opportunistic sensing may deplete the battery rather rapidly, it is crucial to implement effective resource management strategies to maximize the battery life of these phones. We should consider interactions between mobiles and the cloud as well since heavy communications consume large amount of battery power. We can model this as an optimization problem for optimal resource management and compute the best strategy for a given network topology, battery power, and network conditions.

- **Support for mobile developers.** Developer tools such as software libraries with clearly defined APIs will increase the development productivity of mobile-cloud computing systems. The libraries will also help improve the system performance, efficiency, and compatibility while reducing the chances of faulty design and implementation. These APIs and libraries should be easily extensible, easy-to-use, and transparent to users so that users do not have to have knowledge about implementation details.

5 Energy Savings for Mobile Cloud Computing

5.1 Introduction

Cloud computing provides an approach to accessing shared computing resources as a service. In our sensing and computing system, cloud computing plays an important role in saving energy on a mobile device. Traditionally, the cloud is a group of powerful computers, e.g., servers, workstations, personal computers, etc. However, the traditional cloud computing system usually focuses on performance rather than energy efficiency. As the use of energy resources has raised global concerns, looking for more energy efficient approaches to providing computing power is an urgent task for researchers.

Nowadays, mobile devices such as smartphones and tablets are becoming increasingly powerful and rising quickly in popularity. According to International Data Corporation (IDC)'s statistics [9], 1 billion smartphones were sold worldwide in 2013, more than three times of the 315 million total sales of PCs. From 2012 to 2013, the sales of smartphones still have an annual growth of 39%, with expected continued increases in sales in the future. Tablet device sales are also rising sharply, jumping from 19 million in 2010 to 217 million in 2013, getting closer to the sales of PCs. These comparisons are illustrated in Fig. 5.1. With the contin-

uous growth of annual sales and the evolution of technology, it is reasonable to expect that in the near future there will be enormous amount of computing power available from tablets and smartphones all over the world.

In addition, unlike personal computers, mobile devices are rarely powered off, even when the owners are sleeping, which translates into hours of unutilized computing resources. There is great potential if we can make use of these idle computing resources. However, the approach to utilizing mobile devices for cloud computing has not been researched extensively, leaving the question as to whether a mobile computing system can be powerful and energy efficient at the same time.

In this chapter, we investigate and develop a system named GEMCloud (Green Energy Mobile Cloud) that uses mobile devices to provide distributed computing services to support computationally-complex and parallelizable applications. Besides the implementation, our focus is on the evaluation of the computing capability and the energy efficiency of the system.

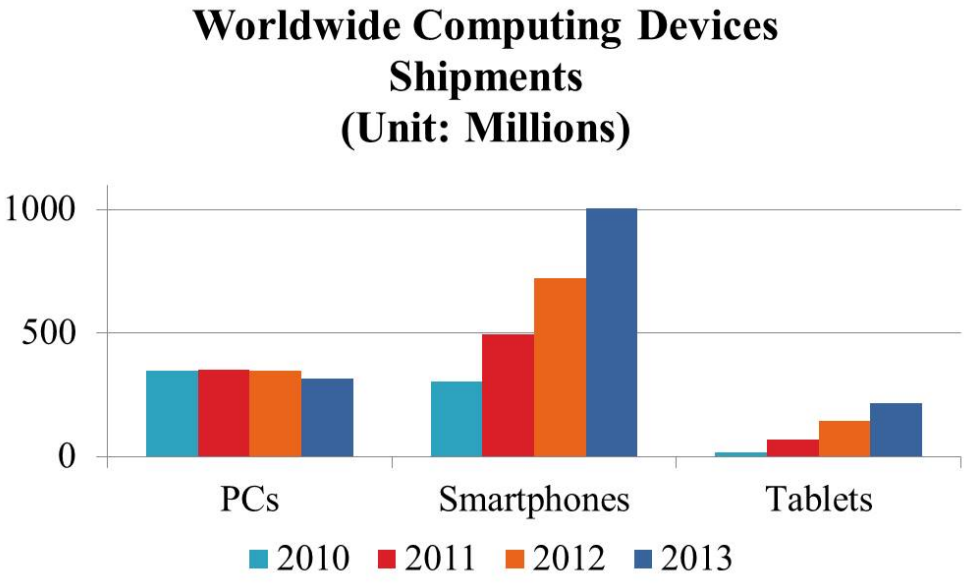


Figure 5.1: Computing device sales comparisons. Data are from [9].

The rest of this chapter is organized as follows. In Section 5.2, we review the current state of the art in the area of mobile computing. Section 5.3 introduces our mobile computing system followed by performance evaluations in Section 5.4. Finally, Section 5.5 concludes the chapter and discusses future directions for this research.

5.2 State of the Art

We propose an energy efficient cloud computing system that provides computational resources from distributed mobile devices to the users. The idea of applying distributed computing and cloud computing to mobile devices has been developed in recent years. However, limited by the traditionally low processing speed and small storage space, the focus of much of this research has been on reducing the computational burden of mobile devices.

One method of reducing the computational burden of mobile devices is to set up an agent between the mobile devices and cloud computing resources to provide mobile devices access to the cloud. For example, the Mobile Cloud Middleware (MCM) [146] introduces a middleware framework that manages the connections and communications between mobile phones and clouds. The MCM also provides an asynchronous server-phone communication mechanism that specifically benefits the mobile phone users.

Another approach to reducing the burden of mobile devices is to offload the computationally-heavy executions to the cloud computing resources. For example, CloneCloud [136] is a system that allows a smartphone device to partially offload its application to the phone's clone (an application-level virtual machine) in the cloud. The authors test CloneCloud on an HTC G1 Android phone. Results show that the CloneCloud approach provides savings in execution time and energy consumption on the mobile devices. A similar idea was also investigated by Cuervo

et al. [121]. Another example from Chen et al. [137] introduces a framework that allows heavy back-end tasks on an Android phone to be offloaded to an Android virtual machine in the cloud.

Both of the above approaches require the computations to be executed in the cloud, which is composed of dedicated computers. However, the mobile devices themselves could be the source of computing power, too. Recent technological advances have greatly improved the performance of smartphones/tablets in terms of CPU/GPU speed, memory size, and storage space. As an example of one of the fastest tablet on the market, the Asus Nexus 7 [155] is equipped with a $1.2GHz$ quad-core CPU with $1GB$ RAM plus $8GB/16GB$ storage. A smartphone or tablet similar to the Asus Nexus 7 can provide a considerable amount of computing power that may be even comparable to the computing power of a desktop computer.

Hyrax [131] has demonstrated the concept of using smartphones as computing resources. The author developed a mobile cloud computing system named Hyrax by porting Hadoop Apache, an open-source implementation of MapReduce, to Android smartphones. Hyrax allows computing jobs to be executed on networked Android smartphones. However, the performance of Hyrax was poor compared with Hadoop on traditional servers, not only because the smartphones were much slower at that time, but also because Hadoop was not originally designed, nor optimized, for mobile devices.

The NativeBOINC for Android project [132] is another example of utilizing mobile devices as computing resources. BOINC (Berkeley Open Infrastructure for Network Computing) [134] is an open-sourced volunteer computing software originally developed for PC users to contribute their computing powers to scientific projects. The NativeBOINC for Android project implemented a BOINC client for Android devices that supports six BOINC projects so far. On the mobile phone client, the user may select several projects to attend and start or stop computing on demand. Eastlack [133] ported BOINC to 4 development boards

with various ARM-based mobile processors and compared their performances with Intel processors. The results indicate that mobile processors have energy efficiency advantages over desktop processors. However, this work does not consider system-level performance comparisons. In this chapter, we develop a working prototype of GEMCloud, a system that exploits the energy efficiency of mobile devices for processing computationally-complex, parallelizable applications.

While the possibility of utilizing mobile devices as computing resources has been demonstrated by researchers, designing and developing a system specific to mobile devices is challenging and must take into account the characteristics of mobile devices in terms of their relatively slow and unstable processing speed, limited battery life, constrained and costly wireless bandwidth and dynamic network topology, among others.

5.3 The GEMCloud System

In this chapter, we design and develop a mobile computing system prototype, namely GEMCloud, that utilizes distributed mobile devices to cooperatively accomplish large parallelizable computational tasks. The main purpose of designing such a system is to find a green approach to making use of the massive amount of idle computing power that is potentially available to the public. In addition, in this chapter we show that a mobile computing system has significant advantages in energy efficiency over traditional desktop computing systems, and, therefore, distributed computing on mobile devices should be explored through prototype implementations.

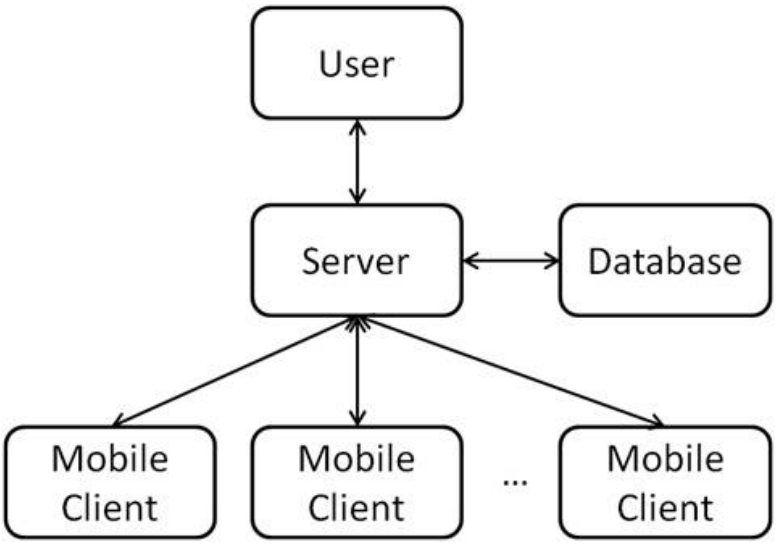


Figure 5.2: The mobile computing system architecture.

5.3.1 System Architecture

Fig. 5.2 shows the system architecture we utilize to create a distributed mobile computing system. The system consists of a network with users who need computing power, a server that is in charge of the organization of the entire network, a database that records the mobile clients' information and task information, and multiple clients of mobile devices that are the computing resource providers.

The role of the server is to organize the entire network and to coordinate mobile clients to perform computational tasks. The server also maintains a database that stores client information such as each client's unique identification, IP address, hardware capabilities, the tasks each client is performing, project data stored on the device, etc. The server may assign the tasks based on the clients' information. For example, the size and number of tasks assigned to a mobile device may be based on the speed and the number of CPU cores on the mobile device.

The clients are mobile devices that connect to the server via the internet and provide computing resources to perform the tasks requested by the user. In our prototype implementation, we use Android [156] smartphones and tablets as the

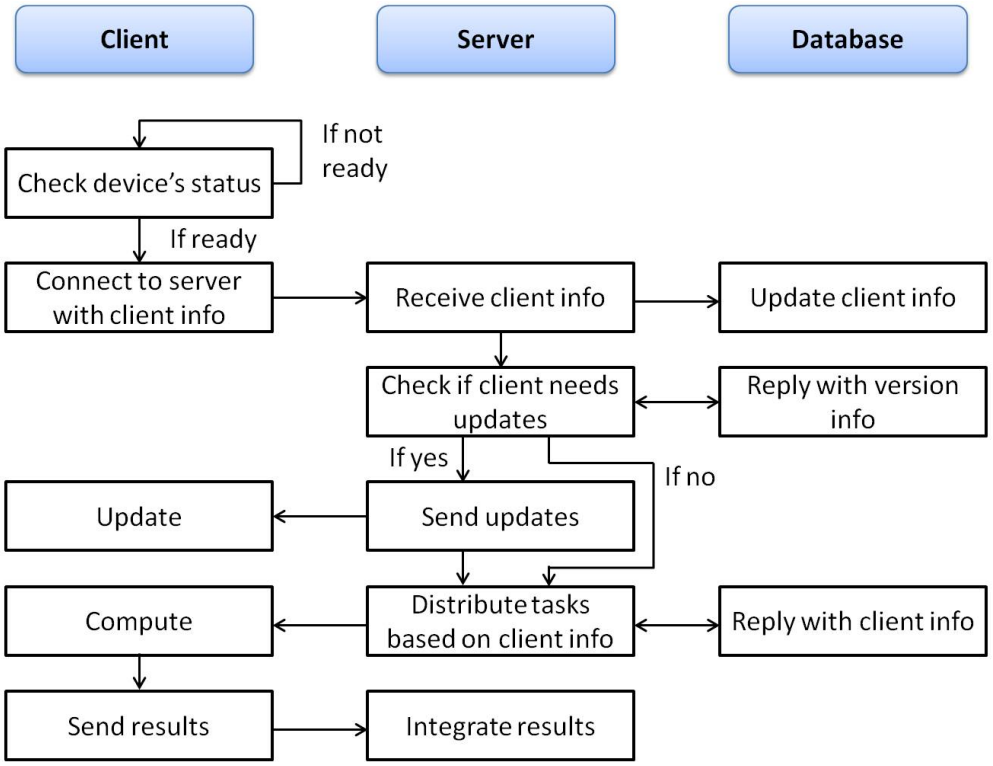


Figure 5.3: The server-client protocol flow chart.

mobile client devices. One of the most important reasons of choosing Android as our development platform is its multi-tasking ability, which is lacking in the iOS platform. Multi-tasking allows an application for mobile devices (i.e., an app) to execute in the background without interfering with the device's other functionalities, which is especially beneficial as multi-core mobile devices are becoming increasingly popular these days. Another important reason to choose the Android platform is that the developer has more control of the mobile device. For example, it is easier to set the percentage of CPU allowed to be used for distributed computing in Android than in iOS.

5.3.2 Server-Client Protocols

In our prototype, the clients follow the server-client protocol flow chart described in Fig. 5.3. Before connecting to the server, the mobile client's application checks the device's status and decides if it should connect to the server and offer its computing resources. For example, if the device is running other applications that consume CPU and memory more than a preset threshold or if the battery is low and not charging, the device will not connect to the server. Such preferences are designed to avoid interfering with the mobile device's normal usage.

After connecting to the server, the client will wait for a response from the server. The server checks the database to determine whether the client's application needs an update. For each mobile client, the server's database stores an entry with a unique client identification, its IP address and other information such as CPU speed, the tasks each client is performing, project data stored on the device, etc. For the prototype we have developed, we are focusing on the evaluation of its computing capability and energy efficiency. Therefore, database management and clients' information that may be used for task distribution are not of concern for now.

Following the updating process, the server assigns tasks to the clients. In our prototype implementation, the server simply assigns the next task on a predetermined list to the next available mobile device. Again, since our current focus is not on task assignment mechanisms, we do not attempt to optimize task distribution in this chapter.

In our prototype, the user's application is a CPU-intensive computing job, which can be split into multiple independent tasks. The user sends a request to the server to complete a job. The server splits the job into small tasks and distributes them to multiple clients. When there are more active clients to provide computing power, the server assigns to each client fewer tasks, which means less time is

Figure 5.4: The prototype screen shot from an Android phone.

required for each client to finish the assigned tasks. Therefore, the turnaround time for the job is reduced when more clients contribute to the computation.

Upon reception of the task assignment, the mobile client performs the computation and sends the results back to the server in a compressed file. The server aggregates all the results and sends them back to the user once all the tasks are completed.

5.4 Performance Evaluations

The main motivation of developing a mobile computing system in this chapter is its potential advantage in energy efficiency. In this section, we evaluate the energy efficiency and computing power of mobile devices and compare them with conventional workstations.

5.4.1 Experimental Setup

In the energy efficiency tests, we evaluate the performance of both the mobile devices and workstations in executing a computationally-complex application. The application used for testing was a protein structure prediction algorithm written in C++. The same algorithm and tasks are used to test on both the mobile devices and the workstations. The mobile devices we tested all use the Android operating system. Since Android applications are written in Java, we use the JNI (Java Native Interface) to execute the native C++ code on the Android devices. The testing code running on the workstations are directly compiled from the C++ code. This may cause the application for the Android platforms to be slower than that for the workstations due to the overhead of Java and JNI integration. However, this is unavoidable when porting C/C++ applications to mobile platforms and thus represents a realistic situation. We used 3 mobile devices and 3 workstations for testing. Their specifications are listed in Table 5.1.

We measure the power consumed by the mobile devices or workstations using the "Watts up? PRO ES" power meter [157]. According to its specifications, the meter has an accuracy of $\pm 1.5\%$ in terms of wattage measurement. We set the recording interval to the lowest value of $1s$. This provides a good sampling rate for our measurements as the test application takes at least $40s$ even on the fastest workstation. We run the same task 10 times on each device and average

Table 5.1: Device specifications (Note: "* 2 (4)" means there are two (four) physical CPUs in the workstation).

Device	CPU	Memory	Release Date
Xiaomi Mi-One (MO)	Qualcomm Snapdragon S3 Dual Core 1.5GHz	1GB	2011
Asus Nexus 7 (N7)	Nvidia Tegra 3 Quad Core 1.2GHz	1GB	2012
Samsung Galaxy S3 (GS3)	Qualcomm Snapdragon S4 Dual Core 1.5GHz	2GB	2012
Workstation1 (WS1)	Intel Xeon E7505 * 2 Single Core 3.06GHz * 2	8GB	2003
Workstation2 (WS2)	Intel Xeon X5355 * 2 Quad Core 2.66GHz * 2	24GB	2006
Workstation3 (WS3)	AMD Opteron 6276 * 4 16-Core 2.33GHz * 4	192GB	2011

the energy consumption readings. For multi-core devices, we run the same task on multiple threads to evaluate the time and energy performance. For mobile devices, we measure the device's energy consumption while the screen is off. For the workstations, we only measure the power consumption of the main unit without the monitor.

5.4.2 Experimental Results

The experimental results are shown in Tables 5.2 - 5.7, corresponding to the Xiaomi Mi-One (MO) Android phone, the Samsung Galaxy S3 (GS3) Android phone, the Asus Nexus 7 (N7) Android tablet and the three Linux workstations (WS1, WS2, WS3).

Our first interest on the mobile devices is the computing power they can provide. As expected, the mobile devices are slower than the workstation competitors. The GS3 is equipped with a dual-core CPU and is able to finish 2 tasks in 217.1s. This means, on average, the GS3 has a computing power of 33.2$tasks/hr$. With the quad-core processor, the N7 has the best computing power among all the mobile devices. It is able to complete 4 tasks within 312.4s, which translates into 46.1$tasks/hr$. This performance is close to the WS1's 51.1$tasks/hr$, showing that mobile devices now may have computing power comparable to some existing desktops and workstations. The WS1 serves as a convenient benchmark for processing speed. We ran the same task on the Amazon Web Services (AWS) and found that each CPU of the WS1 is roughly equivalent to the AWS m1.small instance, which has 1 EC2 Compute Unit. The very high-end WS3 can accomplish 64 tasks within 50.1s. In other words, it is able to complete 4598.8$tasks/hr$, which is about 98.8 times faster than the N7. The times required to complete the same number of tasks for all devices are shown in Fig. 5.5.

When the devices are idle, the GS3 has the lowest power consumption (0.001W) while the MO and N7 consume 0.6W and 0.5W, respectively. As for the workstations we tested, the WS1 has the lowest idle power consumption of 118W, while the other two workstations consume over 350W even when idle. The idle power is an important factor when considering energy efficiency because the power consumed in this period of time is not used for anything productive and is wasted

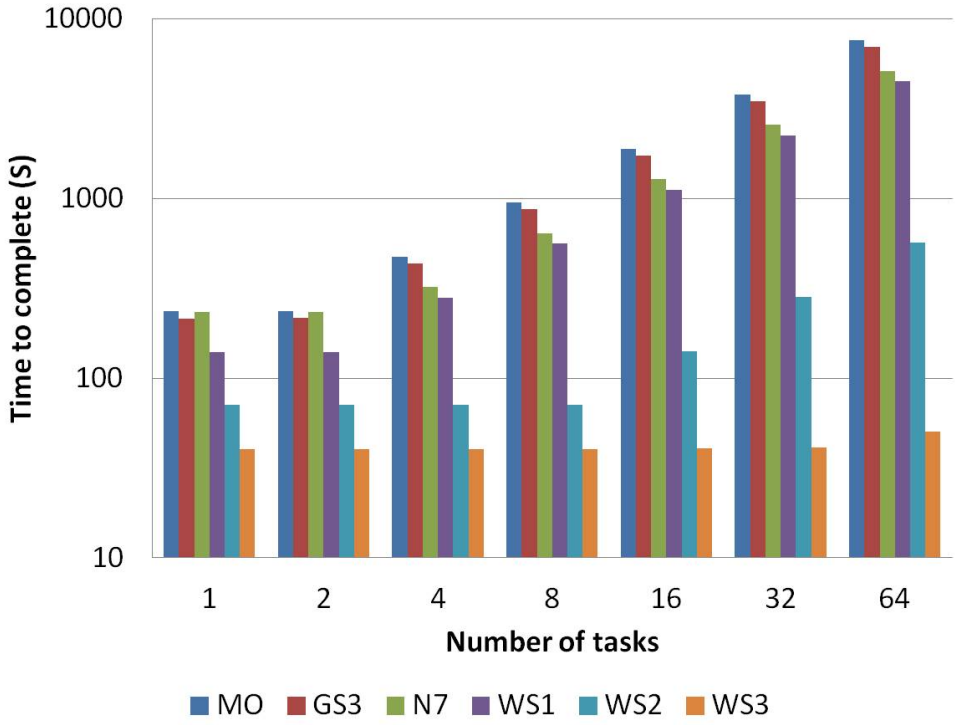


Figure 5.5: Comparison of computing time.

energy. Therefore, in the cases of light utilization, the mobile devices can save a significant amount of energy thanks to their low power consumption when idling.

Due to the fact that mobile device manufacturers build various user interfaces (UIs) on top of the Android platform, the power management strategies vary depending on the device and the manufacturer. For example, the N7 limits the CPU clock while using 4 cores when the screen is turned off. Therefore, we observe a significant increase in the amount of time (25% per core) needed to complete 1 task when all 4 cores are occupied. It is important to consider this power management factor in our future work, as we cannot expect every mobile device to allow applications to run in full speed when the screen is turned off.

As for the energy efficiency, the mobile devices have a clear edge over the workstations. With only 1 task to compute, the GS3 has the best energy efficiency among all the tested devices. It only consumes 288.5J to complete a task, while

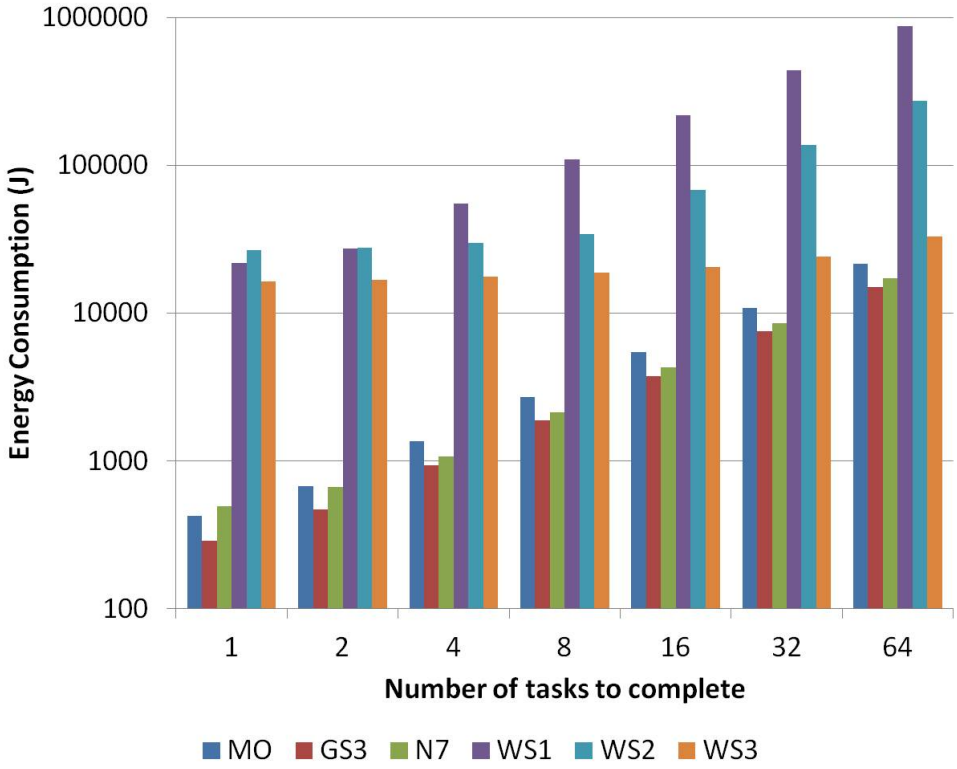


Figure 5.6: Comparison of energy consumption.

the most energy efficient workstation WS3 requires $16,322J$ to process the same task, 57 times the energy required by the GS3. The WS1 and WS2 consume 75.8 and 92.5 times the energy required by the GS3. In other words, the GS3 saves more than 98% of the energy required by the workstations when a single task is computed.

Both the workstations and the mobile devices have their best energy efficiency when their CPUs are fully loaded. Fig. 5.6 shows the amount of energy required for each device to complete a certain number of tasks. All the mobile devices we tested have better energy efficiency than the workstations. More specifically, to complete 64 tasks, the GS3 only requires $15,001.6J$, or $234.4J$ per task, which is the least among all the tested devices. The N7 and MO require $17,110.4J$ and $21,648J$, respectively. As for workstations, the WS3 requires $32,981.2J$ in total,

which is $515.3J$ per task, 2.2 times what the GS3 requires. The WS1 and WS2 consume $877,721.6J$ and $273,075.2J$ respectively, which is 58.5 and 18.2 times what the GS3 consumes. In other words, the GS3 saves 98.3%, 94.5% and 54.5% of the energy required by the WS1, WS2 and WS3, respectively, when the CPUs of each device are at full load.

In cases where the computing job can be split into multiple independent tasks, the energy efficiency is a performance metric more important than the computing speed, as we can increase the system computing speed by recruiting more devices to work cooperatively on the job. However, there is no easy way to improve the energy efficiency. For example, if a job consists of 320 tasks similar to the ones used in our test, a WS3 takes $250.5s$ to complete the job and consumes $164,906J$ in total. If we use the computing resources of 160 GS3, each processing 2 tasks, the total amount of time required to complete the job is $217.1s$ while the energy consumed is only $74,992J$, 45.5% of that consumed when using WS3.

It should be noted that the WS3 represents a class of high-end workstations/servers with significant hardware acquisition cost, which is rarely seen in commercial cloud computing service offerings. The WS1 and WS2, which uses 58.5 and 18.2 times the energy of what the GS3 costs, respectively, are more representative of servers deployed as part of cloud computing infrastructure. Furthermore, the energy efficiency gap widens quickly as the device utilization level decreases, which is due to the fact that, when being idle, workstations consume hundreds of times more energy than do mobile devices. Therefore, using mobile devices as a green computing resource alternative to a conventional server-based cloud is promising and feasible.

Table 5.2: Performance results of the Xiaomi Mi-One (1 CPU, 2 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks
Power (Watt)	0.6	1.8	2.9
Computing Time (Sec)	N/A	235.0	236.1
Total Energy (Joule)	N/A	424.1	676.5
Energy per Task (Joule)	N/A	424.1	338.3

Table 5.3: Performance results of the Samsung Galaxy S3 (1 CPU, 2 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks
Power (Watt)	0.001	1.3	2.2
Computing Time (Sec)	N/A	214.4	217.1
Total Energy (Joule)	N/A	288.5	468.7
Energy Per Task (Joule)	N/A	288.5	234.4

Table 5.4: Performance results of the Asus Nexus 7 (1 CPU, 4 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks	4 Cores 4 Tasks
Power (Watt)	0.5	2.1	2.8	3.3
Computing Time (Sec)	N/A	233.7	234.3	320.7
Total Energy (Joule)	N/A	492.3	661.8	1069.4
Energy Per Task (Joule)	N/A	492.3	330.9	267.4

5.5 Conclusions

In this chapter, we introduce GEMCloud, a mobile cloud computing system that provides computing resources to the user from energy efficient mobile devices. We

Table 5.5: Performance results of the workstation 1 (2 CPUs, 2 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks
Power (Watt)	118.0	156.4	195.8
Computing Time (Sec)	N/A	139.8	140.1
Total Energy (Joule)	N/A	21,864.4	27,428.9
Energy per Task (Joule)	N/A	21,864.4	13,714.4

Table 5.6: Performance results of the workstation 2 (2 CPUs, 8 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks
Power (Watt)	362.6	377.9	393.0
Computing Time (Sec)	N/A	70.6	70.6
Total Energy (Joule)	N/A	26,687.8	27,755.1
Energy per Task (Joule)	N/A	26,687.8	13,877.6
	4 Cores 4 Tasks	8 Cores 8 Tasks	
Power (Watt)	423.5	482.2	
Computing Time (Sec)	70.6	70.8	
Total Energy (Joule)	29,917.4	34,134.3	
Energy per Task (Joule)	7479.4	4266.8	

provide the design of the system and implement a prototype for testing. Our contribution is mainly focused on the evaluation of the energy efficiency of this system by providing comprehensive tests on the mobile devices. We provide performance comparisons among various mobile devices and workstations. The results show that the smartphones and tablets have lower individual computing power but

Table 5.7: Performance results of the workstation 3 (4 CPUs, 64 Cores)

	Idle	1 Core 1 Task	2 Cores 2 Tasks
Power (Watt)	395.6	407.8	420.3
Computing Time (Sec)	N/A	40.0	40.0
Total Energy (Joule)	N/A	16,322.0	16,824.4
Energy Per Task (Joule)	N/A	16,322.0	8412.2
	4 Cores 4 Tasks	8 Cores 8 Tasks	16 Cores 16 Tasks
Power (Watt)	438.7	466.9	506.4
Computing Time (Sec)	40.1	40.2	40.6
Total Energy (Joule)	17,586.8	18,751.1	20,538.4
Energy per Task (Joule)	4396.7	2343.9	1283.7
	32 Cores 32 Tasks	64 Cores 64 Tasks	
Power (Watt)	588.8	658.1	
Computing Time (Sec)	41.0	50.1	
Total Energy (Joule)	24,150.8	32,981.2	
Energy Per Task (Joule)	754.7	515.3	

much higher energy efficiency. The lower computing power can be made up by recruiting more devices, while the energy efficiency is harder to improve given the same type of devices.

6 Volunteer Computing on Mobile Devices

In Chapter5, we have demonstrated that using mobile devices as computing resources in a cloud may provide adequate computing power while achieving higher energy efficiency compared with traditional workstations. To form a mobile cloud, the devices may be from the employees in a corporation, or the devices may be from the crowd. In this chapter, we investigate the latter, using crowd-sourced mobile devices for computing, namely, mobile volunteer computing.

6.1 Introduction

Volunteer computing is a form of distributed computing that utilizes an interconnected network of devices volunteered by public participants to cooperatively complete computationally complex tasks. Traditional cloud computing usually costs either a significant amount of money for the construction and maintenance of the infrastructure or a subscription fee for using commercial cloud services. Volunteer computing provides an alternative inexpensive approach, as the computing resources are usually donated by volunteers. Since many of these devices are underutilized, when properly designed, volunteer computing platforms are able to harvest idle computing power without interfering with volunteers' normal usage of their devices.

Projects in physics, mathematics, molecular biology, medicine, chemistry, astronomy and many other fields have taken advantage of existing volunteer computing platforms such as JXTA [158], XtremeWeb [159], Berkeley Open Infrastructure for Network Computing (BOINC) [134], and GEMCloud [160]. Usually, these projects require a large amount of computing power and are able to be split into smaller tasks in order to be assigned to different volunteered devices. Among the volunteer computing platforms, BOINC is the most successful one with more than 200,000 volunteers and more than 400,000 computers active at the time of writing this article [161]. Approximately 8000 PetaFLOPS of computing power per day is contributed to the scientific projects supported by BOINC.

With the advances in semiconductor technology, mobile devices are becoming more and more powerful. Some of the flagship smartphones such as Samsung Galaxy S5 are equipped with 2.5GHz or higher frequency quad-core CPUs [162]. With 64-bit processors appearing on smartphones [163] and tablets [164], mobile devices are getting closer to PCs in terms of computing capabilities. According to IDC, over 1 billion smartphones and 70 million tablets have been shipped in 2013 [165] [166]. Since smartphones and tablets are usually idle rather than turned off when they are not being used, these devices form a huge pool of idle computing resources that could be potentially harvested for volunteer computing.

While volunteer computing projects have been successful on personal computers, there are only a few projects exploring the utilization of mobile devices as the source of volunteer computing, such as [167] [168] [133] [132]. Currently, there are only a few volunteer computing platforms running on Android devices that can be publicly downloaded through the Google Play Store, including BOINC and two re-packaged versions of BOINC: HTC power to give [169] and Samsung Power Sleep [170], and GEMCloud [160].

Motivated by a desire to better guide future design and development of volunteer computing platforms utilizing mobile devices, we have conducted a pub-

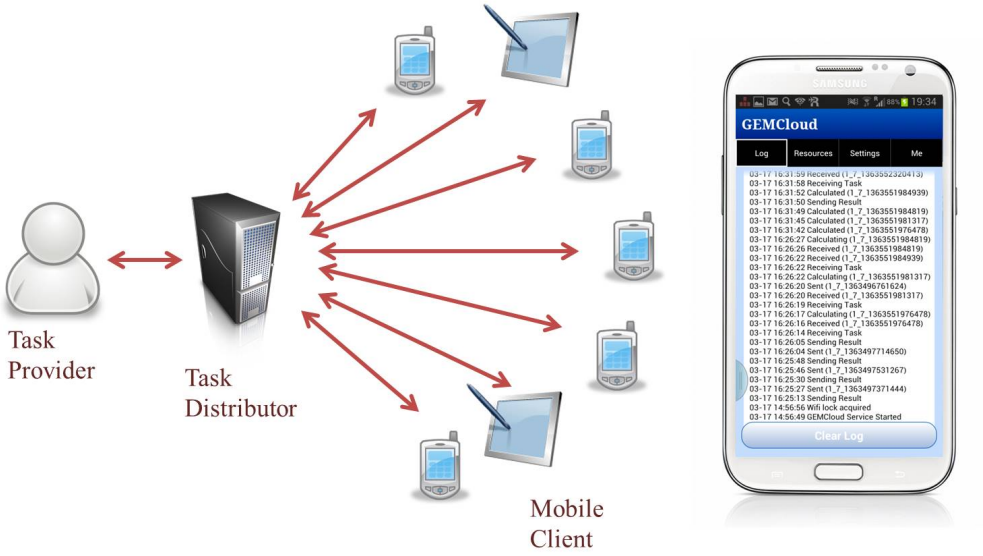


Figure 6.1: The architecture of the GEMCloud platform.

lic study using GEMCloud, a mobile volunteer computing platform described in Chapter 5. The goals of the study included: 1. Analyze the participants' availability for volunteer computing on mobile devices; 2. Analyze the participants' behavior when using different types of devices; and 3. Analyze the factors that impact mobile device users' participation in volunteer computing.

The rest of this chapter is organized as follows. In Section 6.2, we briefly review the design and development of the GEMCloud mobile volunteer computing platform (for a full description of the GEMCloud architecture, see Chapter 5). Section 6.3 describes the set-up of the public study. Section 6.4 provides analysis of the study participants' availability for volunteer computing. Section 6.5 analyzes the behavior of the participants. Several factors in the mobile volunteer computing system that have impact on the volunteers' normal usage of the devices are analyzed in Section 6.6. Finally, Section 6.7 concludes the chapter.

6.2 GEMCloud Platform

We use GEMCloud (described in Chapter 5) as the mobile volunteer computing platform for this study. The GEMCloud system architecture is shown in Fig. 6.1. There are three roles in the system: task provider, task distributor, and task executor. A task provider is the one who needs computing resources for a distributed computing project. The project must be parallelizable, such that it can be divided into smaller pieces of tasks that will be uploaded to the task distributor. Each task should be properly sized so that it can be finished by a mobile device within a reasonable amount of time. A task distributor is a set of programs running on a server that continuously listen for task requests. The tasks will be distributed to mobile clients (task executors) that have announced their availability and requested tasks.

To minimize interference with mobile device users' normal usage of the device, GEMCloud provides preference settings to the users to customize when and how they want GEMCloud to utilize their devices' computing resources. The settings include:

- Whether the GEMCloud app can communicate with the server using cellular data or only WiFi (default: WiFi only).
- Whether GEMCloud can perform computation while the mobile device is operating on battery (and the minimum battery charge level) or only when plugged in to wall power (default: Only when plugged in).
- The number of CPU cores that may be used for computation (default value: 1) and the total CPU capacity that must be retained (default value: 65%)
- The maximum storage that can be used by GEMCloud (default value: 160MB).

- The maximum value of phone temperature (to prevent overheating) (default value: 38°C).
- Motion detection that allows GEMCloud to reduce its CPU usage temporarily when detecting motion of the device, which may indicate that the user has started using the device (default: enabled).

The default settings as shown above only allow the app to run computation while charging and communicate with the server under Wi-Fi connection, which minimizes interference to the normal device usage. However, users with larger battery life devices (e.g., tablets) or those who do not care about the resource usage can dedicate more computing time to the project by modifying the settings to fit their preferences. There is also a resource tab to allow users to monitor GEMCloud's usage of the battery, CPU, memory, and data traffic.

Besides preference settings, another potential concern for volunteers is the energy cost for participating in the volunteer computing and performing the computation. It has been shown that using smartphones and tablets for distributed computing can perform the same computing tasks as on PCs with better energy efficiency compared with high performance workstations in Chapter 5. For each volunteer, the extra electricity for performing the computing costs very little. A common scenario for mobile volunteer computing is when the mobile device user is sleeping, the device is plugged in to the wall charger and performing volunteer computations. For a Samsung Galaxy S3 smartphone, once the phone is charged, the power consumption for performing computation without turning on the screen is 1.3W when using 1 core (at its full capacity) and 2.2W when using 2 cores. Assuming the extra energy cost for doing the computation while charging the battery has negligible difference compared with when the battery is fully charged and still plugged in, the total energy used for 8 hours of volunteer computing per night is 10.4Wh for 1 core and 17.6Wh for 2 cores. According to the U.S. Energy Informa-

tion Administration, the national average electricity price in the United State for residential usage in 2013 was 12.12 cents/kwh [171]. Using this rate as a guide, the total electricity cost for participating in volunteer computing on a Galaxy S3 smartphone is only 0.13 cents/night when using 1 core and 0.21 cents/night when using 2 cores. Therefore, the monetary cost for each volunteer to participate in the mobile volunteer computing project is very low and should not be a factor that discourages volunteers.

6.3 Volunteer Computing on Mobile Devices Study Details

As we have discussed in Section 6.1, the massive number of underutilized mobile devices can potentially provide a significant amount of computing power for volunteer computing projects. However, it is not known whether people will embrace this concept and join volunteer computing projects in their daily lives. It is also crucial for volunteer computing project designers to understand how the volunteers may behave and what is important to retain the volunteers' loyalty. To evaluate the feasibility of using mobile devices for volunteer computing, and to analyze the behavior of the volunteers and the factors that impact their behavior while participating in the volunteer computing project, we conducted a public study using the GEMCloud platform.

6.3.1 Study Set-up

Using flyers as our main approach for soliciting volunteers, we attracted volunteers from among the students, staff and faculty at the University of Rochester to run the GEMCloud app on their Android mobile devices. Others who learned about the study from family or friends and did not have any affiliation with the

University of Rochester also participated in the study. For each participant, we asked them to respond to a pre-study survey when the participant joined the study and a post-study survey after participating in the study for one month. We awarded a $20 Amazon gift card to each participant who finished both surveys and successfully installed the GEMCloud app and created an account using the same email address as the two surveys. The prize was provided to award the participants for their time and effort on filling out the surveys and providing their feedback. To associate the survey data with the volunteer computing results, we asked each participant to create an account within the app using the same email address as that used to complete the surveys. However, there is no requirement for a participant to finish any computing tasks in order to be awarded the gift card. Hence, we expect to collect computing results based on participants' volunteering activities.

6.3.2 Task Distribution

The computing tasks used for the study are protein structure prediction tasks. The algorithm to perform the computation is installed along with the GEMCloud app. There are three different types of tasks. According to their sizes, we name them as small, medium and large tasks. As a reference, a Samsung Galaxy S3 can finish a small task in 170 s, a medium task in 280 s and a large task in 469 s. Each type of task requires a set of files that will be transmitted from the server to the mobile device the first time the mobile device receives the type of task. These files do not need to be transmitted again if the mobile device receives the same type of task again. For each type of task, a task description file that includes a sequence of atoms will be sent to the mobile devices. For this study, when a mobile device requests a task, a randomly selected type of task (small, medium or large) was sent.

6.3.3 Data Collection

We gathered information from the devices of the participants (e.g., processor capabilities, type of wireless connections used) as well as requesting participants to complete surveys that informed us of their expectations (pre-survey) and experiences (post-survey) with having GEMCloud on their Android mobile devices. For all the participants, there were at least 30 days between when they filled out the pre-survey and when they received the post-survey, and we continued collecting data from the participants even after they completed the post-survey. The first participant joined the study on Oct. 31, 2013. The last date that a participant filled out our post-study survey was on Mar. 17, 2014. We set the last day of data collection to be Apr. 17, 2014, the day our last participant had been participating for a month after completing the post-survey. Therefore, although each participant joined the study on a different day, we have collected data from each participant for at least 60 days. In the later analysis, when we analyze the individual behavior of the participants, we use the first 60 days of data for each participant in order to have comparable data for each participant.

6.3.4 Participant Demographics

As of Apr. 17, 2014, 132 participants finished the pre-study and post-study surveys as well as installed the app since the study started on Oct. 31, 2013. The majority of the participants were students, who account for 60% of the total participants. University employees account for another 11%. 28% of all the participants were not affiliated with the University of Rochester. In terms of the age distribution of the participants, 86% of the participants were between 18 to 29 years old. The participants for this study were generally a representation of the university community. The details of the participant demographics are shown in

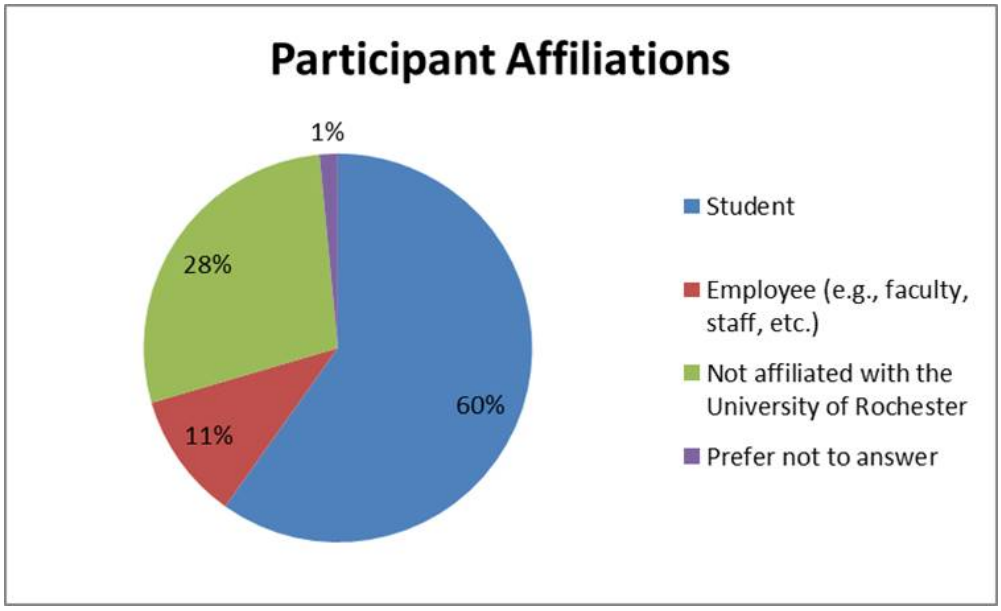


Figure 6.2: Participant affiliations.

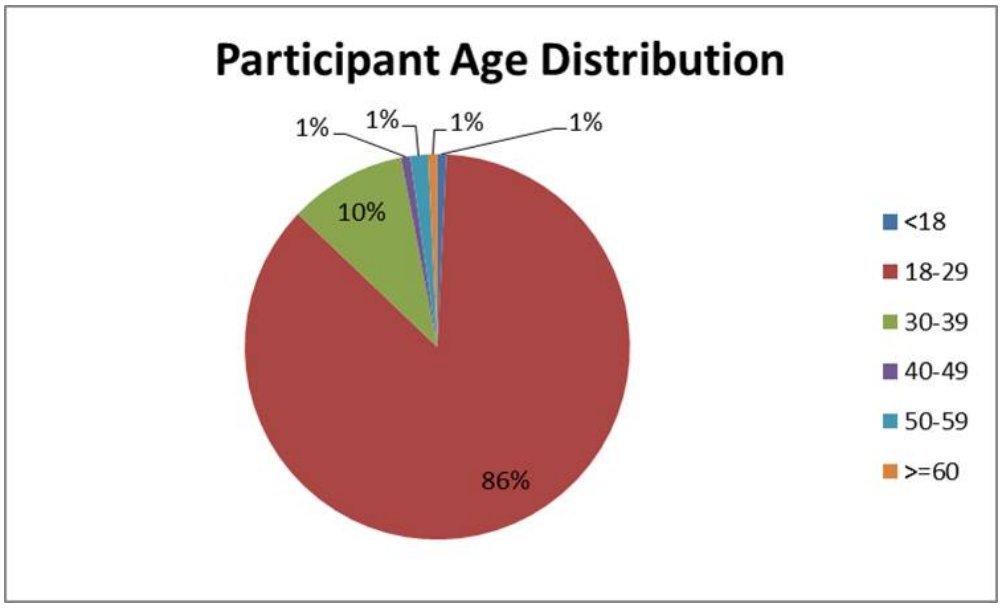


Figure 6.3: Participant age distribution.

Fig. 6.2 and Fig. 6.3.

6.4 Participants' Availability

One of our goals for conducting this study was to verify the feasibility of using mobile devices for volunteer computing. From the survey, only 8 out of the 132 participants (6%) indicated they were not willing to contribute the idle time of their devices for volunteer computing to support scientific discovery and medical research. As shown in Fig. 6.4, among the 132 participants, 120 or 91% of the total participants finished at least 1 task during their first 60 days of participation in the study. 94 or 74% of the participants finished more than 100 tasks. 12 participants, or 9% of the total, however, did not complete any tasks. This means they uninstalled the app before completing any tasks. In total, 322,689 tasks including 189,814 small tasks, 95,095 medium tasks and 37,780 large tasks have been completed by the participants. These equal to a total of 68 days of full running on an Amazon EC2 large instance. Given the duration of this study and the number of participants, the completed results indicate a significant amount of computing power provided by each participant.

6.4.1 Geographical Distribution of Participants

We only advertised the study on the University of Rochester campus through flyers. However, through social networks and word of mouth, participants from all over the world learned about the study and installed the GEMCloud app and contributed their computing resources. Based on the IP address from which each task result was sent, we were able to approximately localize where the result was from by using the GeoIP web service provided by MaxMind Inc. [172]. Fig. 6.5 shows a Google map with all the task results pin pointed on their approximate

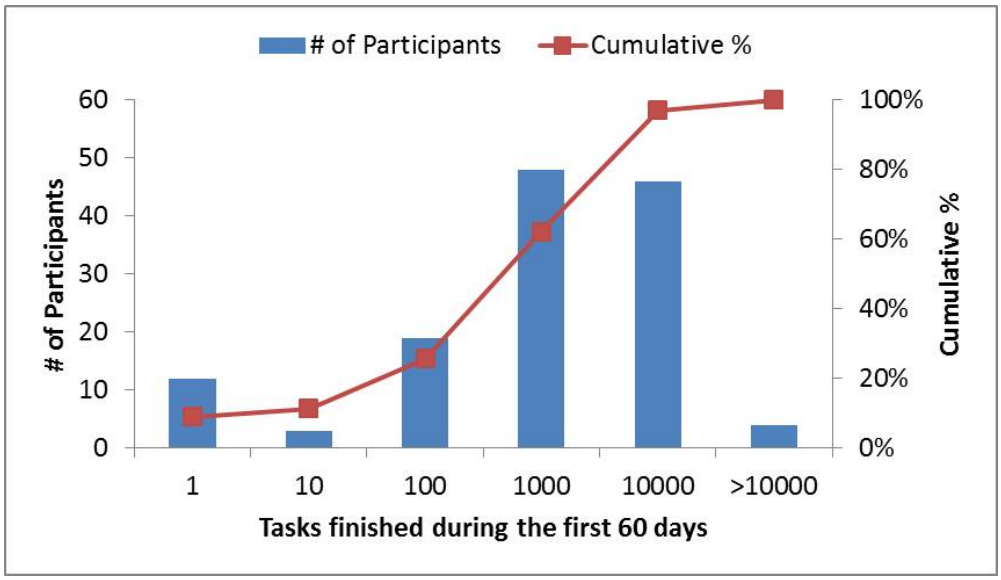


Figure 6.4: Histogram of the number of tasks finished by each volunteer during the first 60 days.

locations. Task results from similar locations are grouped together. As of the last day of data collection, we received results from 13 different countries all over the world. It should be noted that the task results here also include those that were not generated by the study participants, i.e., from participants who did not complete the surveys.

6.4.2 Active Participants During the Study

Fig. 6.6 shows the numbers of daily active participants since the study began on Oct. 30, 2013 until Apr. 17, 2014, a month after the last participant received the prize. We define a daily active participant as a participant who successfully returns at least one task result back to the server on that day. As we can see from Fig. 6.6, after 27 days, the number of daily active participants reached its peak value of 53 active participants per day. Since day 36, the number of daily active participants started to decrease. As shown in Fig. 6.7, the number of task

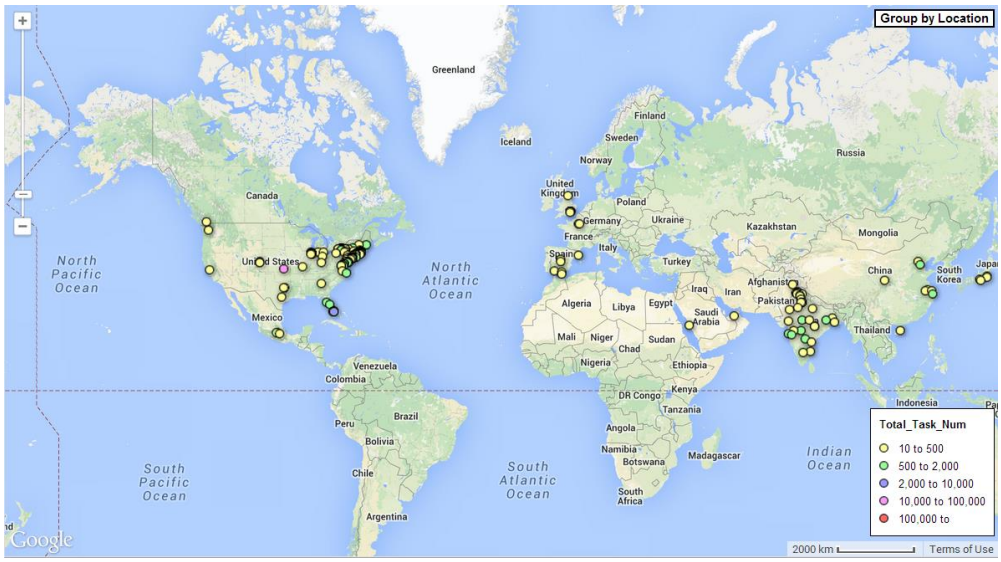


Figure 6.5: Map showing where volunteer computing users are located.

results all participants returned each day is in general consistent with the number of active participants per day. The drop in the number of active participants is after the first group of participants received their prizes, an indication of the role of the prize as one of the motivations for participants volunteering their devices for scientific computing. In Section 6.6, we will provide detailed analysis of the effect of prizes in our volunteer computing study.

6.4.3 Active Duration of Participants

We are interested in how long each participant may engage in a volunteer computing project similar to GEMCloud. Therefore, we analyze the active duration of each participant starting from the day he/she joined our study. As Fig. 6.8 indicates, on the first day after each participant joined the study, there are 120 active participants. This number drops to 82 after 5 days of participation, since there is no requirement for participants to continuously uses the GEMCloud app. After that, the number of active participants decreases at a slower rate. After 30

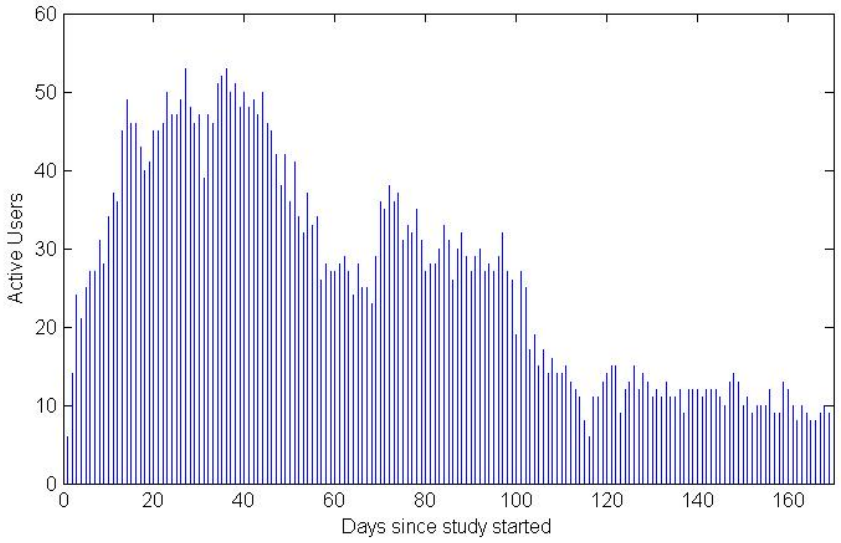


Figure 6.6: Number of active participants since the study started.

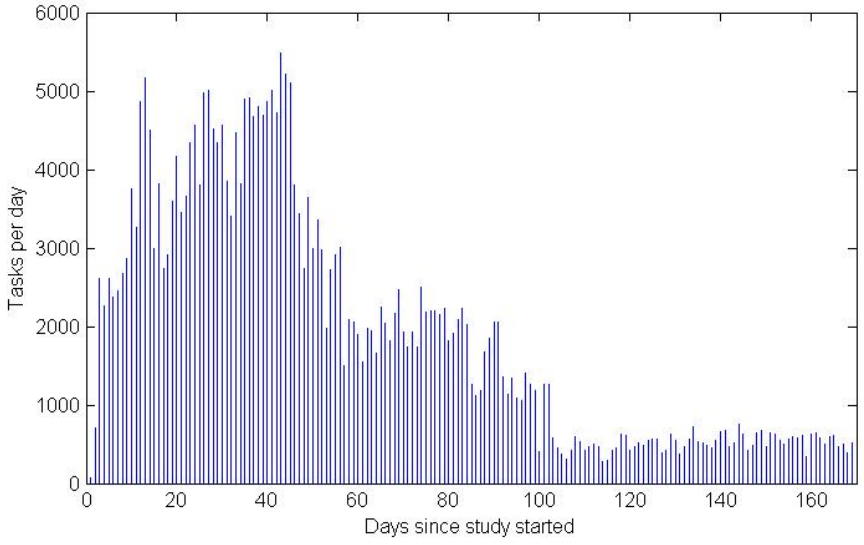


Figure 6.7: Number of daily task results returned since the study started.

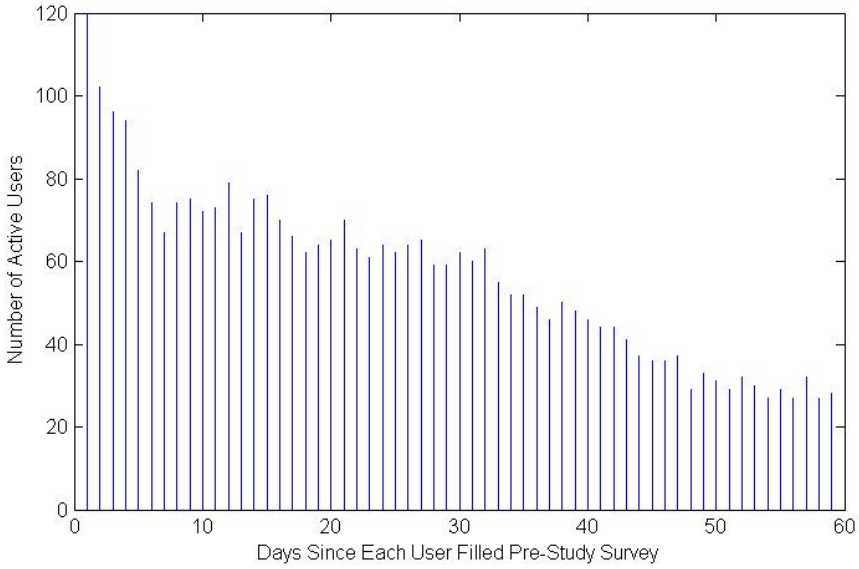


Figure 6.8: Number of active participants since joining the study.

days, there are still more than 50% of the participants who remain active.

Since every participant may join the study on a different day, the total participation duration of each volunteer is different. In order to compare the behavior of each participant, we analyze the active ratio of the participants during the first 60 days of their participation. Fig. 6.9 shows a histogram of this active ratio, which is defined as the number of days of active participation out of the first 60 days of participation. As we can see, 35 participants or 26.5% of all 132 participants have an active ratio lower than 20% during their first 60 days of participation. On the other end, 13 participants or 9.9% of all 132 participants were active for more than 90% of the days during their first 60 days of participation. 52 participants or 39.4% of all participants, have an active ratio of more than 50%, showing again that a significant portion of the participants stay active for at least a month during their participation.

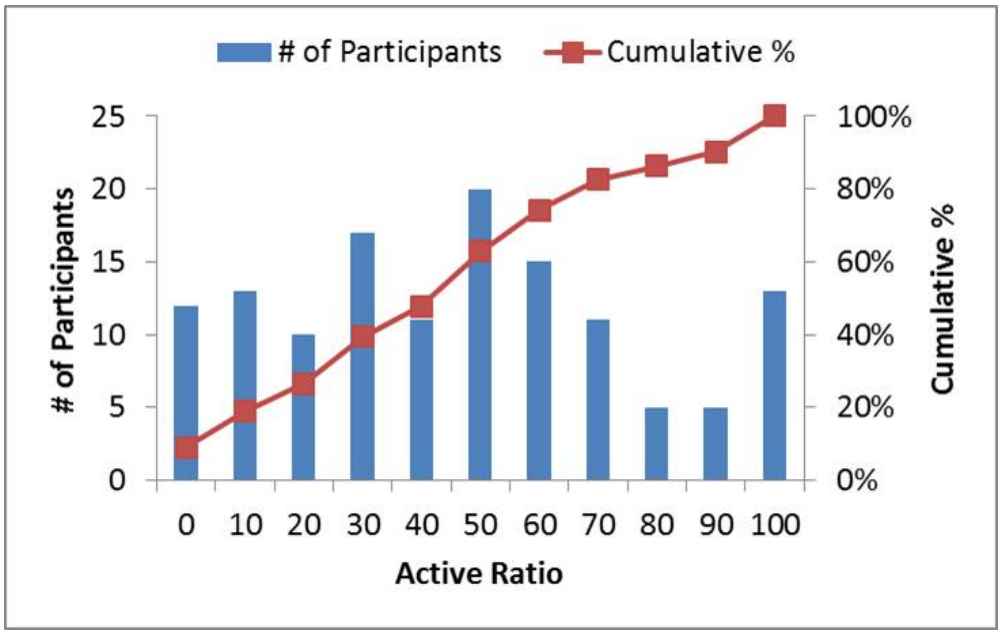


Figure 6.9: Active ratio for the first 60 days.

6.5 Analysis of Participants' Behavior

According to our surveys, among all 120 participants who have successfully returned at least one task result, 112 of them claim to have at least 1 Android smartphone and 40 of them claim to have at least 1 Android tablet. Based on the task results we collected, there were 134 devices used by these participants to provide computing resources, 111 of them are smartphones and 23 are tablets.

In general, mobile devices are different than desktop devices such as PCs or workstations, as mobile devices are almost always on and are powered by batteries when they are carried around, and will be charged periodically or at their owners' convenience. Since mobile devices have limited battery lifetime, it is usually not desirable to the device's owner to sacrifice the device's battery life for volunteer computing. The GEMCloud app is designed with preference settings allowing the user to decide when GEMCloud can perform computing tasks. The default settings only allow GEMCloud to request and process tasks when the device is

charging and connected to WiFi so that running GEMCloud will not utilize battery energy and will not utilize mobile data. On the other hand, the user may change the default settings to allow the app to run when the device is using cellular data or when the device is operating on battery power. Obviously, allowing the device to run volunteer computing projects on battery and use mobile data will improve the availability of the device. We are interested to see how many participants were willing to use their device for volunteer computing on battery or with cellular data.

6.5.1 Comparing Smartphone and Tablets Use

Smartphones and tablets have both different hardware and different uses/purposes. For example, smartphones usually have cellular connectivity and have battery lifetimes of less than a day, while tablets usually only have Wi-Fi connectivity and may last longer than one day on a fully-charged battery. The participants may use a smartphone or tablet in our study, and the type of device affects the usage behavior. Our survey data show different usage styles among the smartphone users and the tablet users. Specifically, from the survey data shown in Table 6.1, Table 6.2 and Table 6.3, the smartphone users tend to use the devices longer everyday and, due to the more limited battery life, charge them more frequently. This is very important to volunteer computing, as in general, volunteer computing is aimed at harvesting free computing cycles without interfering with the mobile device users' normal activities.

Table 6.1: Survey Stats: Daily usage time of smartphones and tablets

Daily Usage	Smartphones		Tablets	
	# of participants	% of participants	# of participants	% of participants
<1 hour	6	5%	11	28%
1-3 hours	39	35%	17	43%
4-6 hours	39	35%	11	28%
7-10 hours	19	17%	0	0%
>10 hours	9	8%	1	3%

6.5.2 Usage Patterns for Smartphone and Tablet Devices

We collected the model name of each device along with the returned task results. Based on the model of the device, we can classify the type of that device, i.e., smartphone or tablet. We analyze the numbers of tasks completed per minute according to the time of day by smartphone users and tablet users. As shown in Fig. 6.10 and Fig. 6.11, the behaviors of smartphone users and tablet users are different. The number of tasks completed by smartphone users reaches its peak during the night (03:00 - 07:00) and falls to its minimum during the afternoon (14:00 - 20:00). Since most participants set the GEMCloud app to run computations only when the device is charging and connected to WiFi, it is expected that participants charge their devices during the night and use the devices off battery throughout the day. On the other hand, for tablet users, the tablet devices don't always need to be charged on a regular basis as the battery lives for tablets are usually longer than a day. Therefore, we do not see any peak hour in the tablet users' data.

Table 6.2: Survey Stats: Charging frequency of smartphones and tablets

Charging Frequency	Smartphones		Tablets	
	# of participants	% of participants	# of participants	% of participants
Multiple times per day	29	26%	4	10%
Once per day	76	68%	15	38%
Once every 2 days	7	6%	10	25%
Twice a week	0	0%	7	18%
<= Once a week	0	0%	4	10%

6.5.3 GEMCloud Preference Settings

People use smartphones and tablets for their everyday purposes. The intention of volunteer computing on mobile devices is to make use of these mobile devices' free computing cycles without interfering with their original usage. To accomplish this goal, we set the default setting of the GEMCloud app to only allow task processing when the device is charging and connected to WiFi, so that the GEMCloud app will not consume any battery energy or cost any cellular data to its user. We also allow the user to change the settings according to their preferences. We collect the preference setting data so that we can understand more about what settings meet the most users' needs. Since one participant may use multiple devices for volunteer computing, we analyze preference settings per device rather than per participant. In total, we have collected preference data from 134 unique devices from 120 participants.

The first thing we are interested to see is how frequently GEMCloud users change their devices' preference settings. As shown in Fig. 6.12, 47% of all the active devices have preference setting changes on the first day the app is installed.

Table 6.3: Survey Stats: Charging time of smartphones and tablets

Charging Time	Smartphones		Tablets	
	# of participants	% of participants	# of participants	% of participants
Morning	25	22%	9	23%
Afternoon	25	22%	9	23%
Evening	42	38%	20	50%
Overnight	87	78%	25	63%

After the first day, the percentage of devices that change the preference settings drops to less than 10% per day.

The survey responses indicate that before starting the study, 96% of the participants would be willing to contribute the idle time of their devices when the devices are being charged, while 71% of the participants would be willing to use their devices for volunteer computing while operating off battery, including 26% that specified the battery usage limits. After 30 days, the number of participants who are willing to use their devices for volunteer computing when the devices are being charged reduces to 93%, and those who are willing to use their devices when they are operating off battery reduces to 61%, including 29% of whom specified the battery usage limit.

Fig. 6.13 shows the number of devices that allow GEMCloud to run tasks when they are on battery and the number of devices that allow GEMCloud to run tasks only when charging. At the beginning of joining the study, 34% of smartphones and 35% of tablets allow the GEMCloud app to run tasks on battery. 66% of smartphone and 65% of tablets only allow the app to run tasks when the device is charging. As time goes on, the percentage of active devices that run tasks on

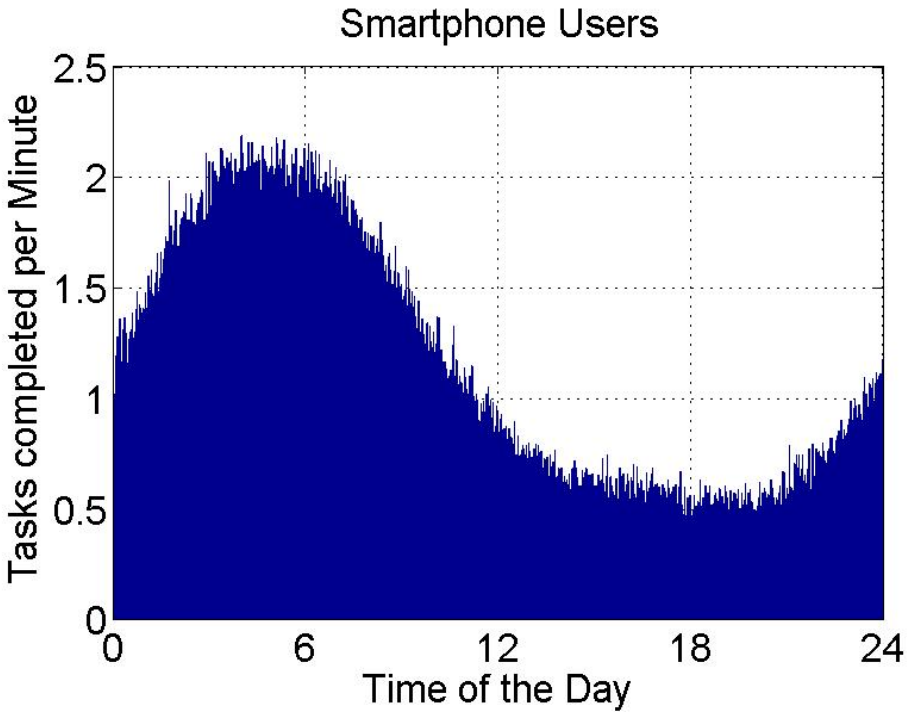


Figure 6.10: Smartphone users usage pattern.

battery gradually reduces to 15% for smartphones and 20% for tablets on the 60th day since the device joins the study. These results indicate that although in the survey, 61% of participants specified their willingness to contribute their devices' idle computing cycles when they are on battery, however, in practice, using mobile devices for volunteer computing on battery is not desirable for most participants.

As for network connectivity, before joining the study, 94% of participants were willing to contribute the idle time of their devices when connected via WiFi, while 56% of participants were willing to contribute when their devices are connected via cellular network, including 18% who specified the cellular data usage limit. After 30 days of participation, the same 94% of participants remain willing to contribute their devices' idling time when connected via WiFi, while the number of participants who are willing to contribute when their devices are connected via cellular networks drops slightly to 51%, which includes 13% who specified the

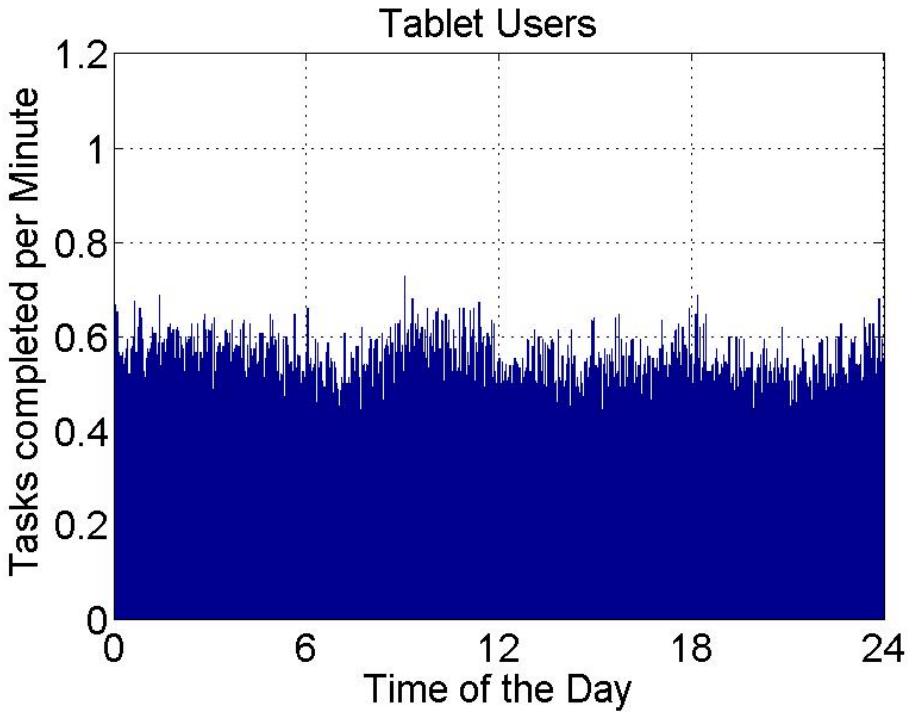


Figure 6.11: Tablet users usage pattern.

cellular data usage limit.

According to the collected preference data, at the beginning of their participation, 84% of active devices 82% of smartphones and 96% of tablets allow GEMCloud to run tasks only when the device is connected to WiFi, while only 8% of smartphones and 4% of tablets allow GEMCloud to run tasks when the devices are connected through cellular networks. As time goes on, the percentage of active devices that allow GEMCloud to run when connected to WiFi rise slightly to 89% of smartphones and 100% of tablets. These results indicate that, in practice, most participants prefer not to use cellular data of their mobile devices for volunteer computing.

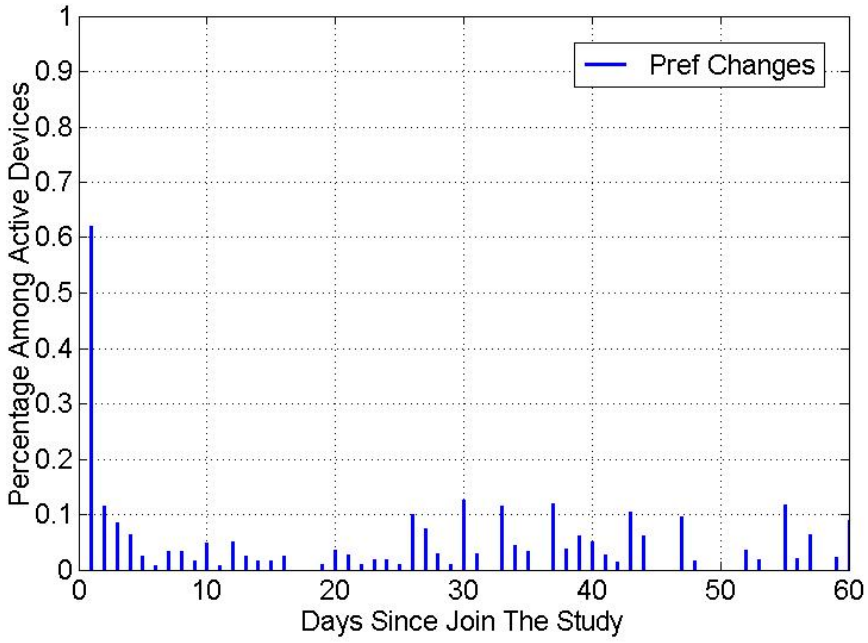


Figure 6.12: Number of preference changes per device.

6.6 Factors that Impact a Volunteer's Behavior

One of the goals of our study is to analyze the factors that impact users' behaviors while participating in the volunteer computing projects. In this study, we identify three important factors that influenced the behavior of participants.

6.6.1 Prizes

The first factor is the prizes. Although in the consent form that participants needed to sign before participating in our study we clearly stated that the volunteers do not need to complete any task in order to be qualified for the prizes, we still observed a large difference in the number of active users and completed tasks before and after the prize was awarded. As we can see from Fig. 6.15 and Fig. 6.16, where the data are aligned according to the day each volunteer receives the prize, clearly there is a drop in the number of active users and daily completed

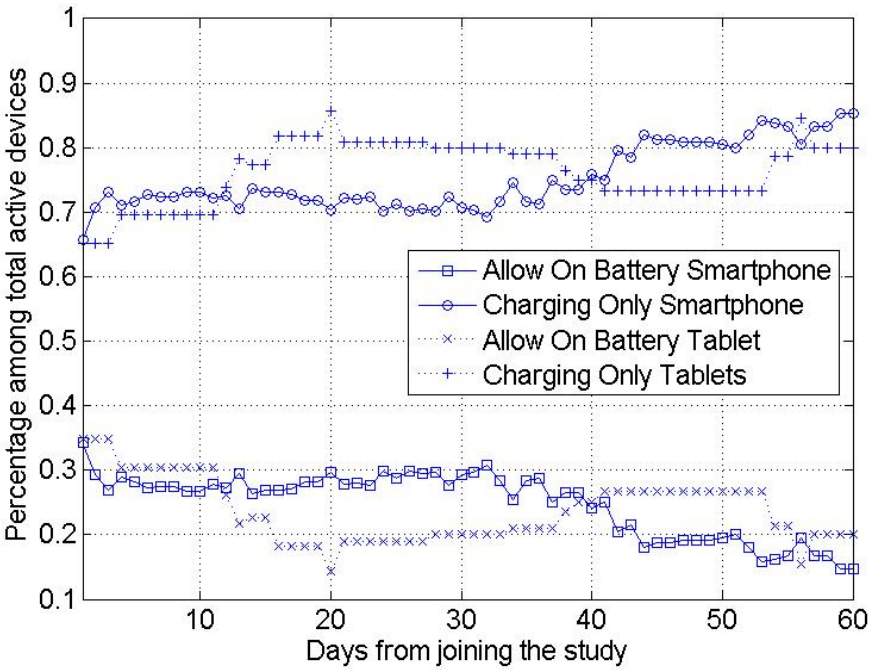


Figure 6.13: Number of devices that allow computing on battery vs. charging only.

tasks on the day that the volunteers receive the prizes.

6.6.2 Negative Impacts to Volunteers' Devices

In the post-survey, we asked the volunteers if they have noticed that the GEM-Cloud app brought any negative impact to their normal usage of the device. As we may expect, if the volunteer allows GEMCloud to run tasks when the device is on battery, the battery life will be noticeably shortened. Aside from battery drain, there are other negative impacts that may be noticeable to volunteers, including the following.

- When a device is continuously running CPU heavy tasks, even while it is

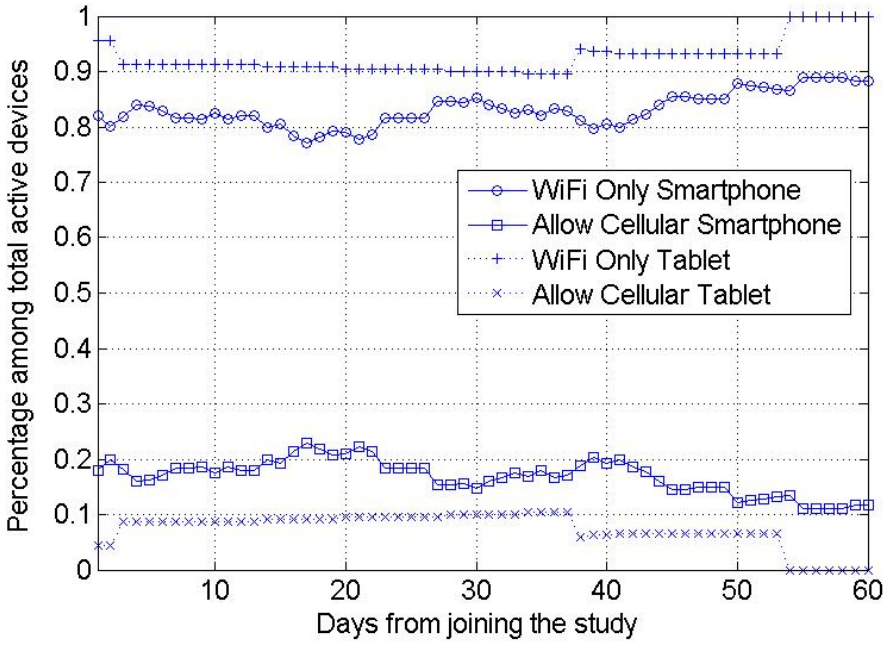


Figure 6.14: Number of devices that allow computing on cellular data vs. Wi-Fi only.

charging, its temperature may rise higher than its normal charging temperature when no CPU heavy task is running continuously. Therefore, the time for the device to be fully charged may take longer.

- The device may become sluggish if the volunteer uses it while it is running CPU heavy tasks, especially when using all of its CPU cores. GEMCloud has a mechanism to allow GEMCloud to adjust its computing load adaptively (e.g., reduce the tasks' CPU load by pausing the task thread(s) for a short period of time) in order to guarantee the percentage of idle CPU according to the user's preference settings. However, this may not totally avoid some users finding their device sluggish when GEMCloud was running tasks.

- In the early version of GEMCloud, when a device's battery condition was changed from charging to discharging, GEMCloud would continue to finish

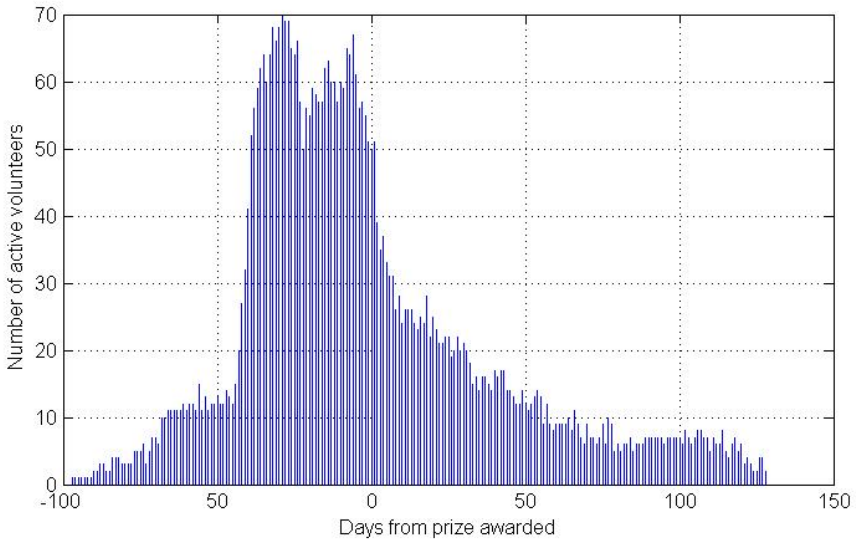


Figure 6.15: Number of daily active volunteers according to the day the users received the prizes.

the current task regardless of the preference setting being that GEMCloud is permitted to work on battery or not. This will cost a small amount of battery energy depending on the device's specifications.

To some volunteers, some of the above negative impacts may be noticeable. We are interested to see whether any negative impacts were perceived by the volunteers and how the negative impacts influence the behavior of those volunteers.

We identify 46 volunteers who only used GEMCloud when their devices were charging and 74 volunteers who have completed tasks on battery. The volunteers who have never completed any tasks on battery finished 1840 tasks on average. The volunteers who have completed tasks on battery finished 2266 tasks on average, 23% more than the former. This is possibly because compared with volunteers who only use their devices for volunteer computing while charging, volunteers who are willing to use their devices for volunteer computing on battery have stronger motivation and therefore are able to contribute more. Allowing volunteer com-

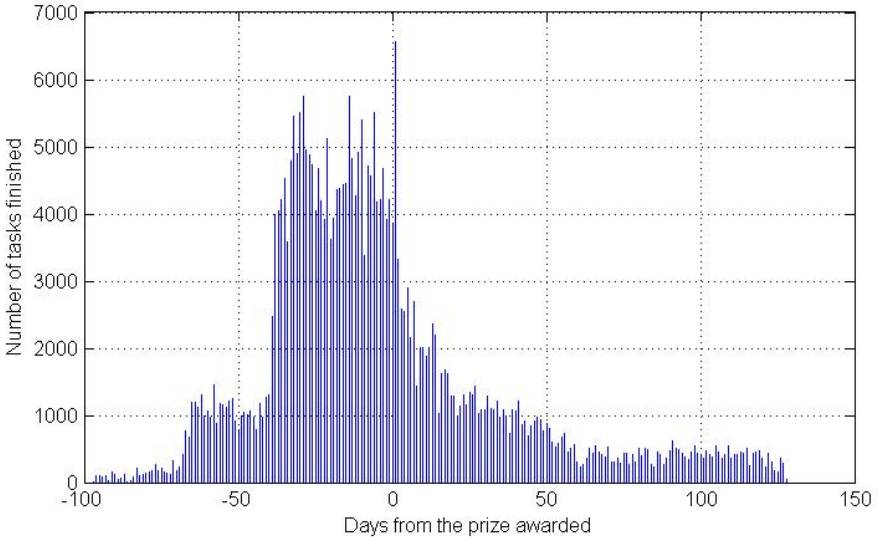


Figure 6.16: Number of tasks finished each day according to the day they received the prizes.

puting on battery may also lead to longer duration per day for each device to contribute.

Within these 46 volunteers who have never completed any tasks on battery, volunteers who did not notice any negative effect have many more active days (29 vs. 20 days) and have completed many more tasks (2150 vs. 960 tasks) than volunteers who noticed some negative impacts, as shown in Table 6.4. This indicates that the volunteers who are not willing to use their devices' battery for volunteer computing projects probably care more about their mobile devices' performance and therefore negative impacts to the device play an important role in determining the amount of the volunteers' contributions.

For the remaining 74 volunteers, each of them has finished at least 1 task while their devices were operating on battery. Since the default settings do not allow GEMCloud to run tasks when the device is operating on battery, this means that these 74 volunteers have changed that setting at some point in the study and are possibly aware of the battery drain resulting from GEMCloud operating on

Table 6.4: Volunteer performance with regard to negative impact

	Only when charging		On battery	
Negative impact	No	Yes	No	Yes
Number of volunteers	34	12	50	24
Avg. active days	29	20	28	31
Avg. computing hours	269	128	221	427
Avg. completed tasks	2150	960	1845	3144
Hours per day	9.3	6.6	7.9	13.8
Tasks per day	74	49	66	102
Tasks per hour	8.0	7.5	8.4	7.4

battery. Within these 74 volunteers, 24 of them noticed that GEMCloud had some negative impacts to their devices and 50 of them did not notice GEMCloud had any negative impacts to their devices. The volunteers who noticed negative impacts from GEMCloud to their devices had an average of 31 active days during their first 60 days after joining the study in comparison to 28 active days for volunteers who did not notice any negative impacts. The users who noticed negative impacts completed 3144 tasks on average, 70% more than the users who did not notice any negative impacts, who finished 1845 tasks on average. It is possible that to some volunteers who are willing to use their devices' battery for volunteer computing projects, negative impacts to their devices' performance are not their concern or they have higher tolerance for the negative impacts. Therefore, although some of them noticed the negative impacts, these volunteers were still willing to contribute.

6.6.3 User-app Interaction Frequency

In the post-study survey, we also asked each volunteer how many times on average he/she interacted with the GEMCloud app. In the app, once the volunteer has set up an account and modified the preference settings, they may leave the app running in the background and the app will only perform computations when it is allowed. This ensures that the app does not require any user intervention. However, on the other hand, the volunteer may check GEMCloud's activity log, the number of tasks that have been completed and his/her ranking among all the volunteers. The volunteer may also change the preference settings, as well as turn the app on or off according to their need at the moment. These user-app interactions may be related to the volunteer's involvement with the volunteer computing project. The more the volunteer interacts with GEMCloud may indicate higher user involvement, which may lead to longer participation. We correlate the survey responses with task results in order to understand the relationship between user-app interaction and the user's contribution for volunteer computing.

Table 6.5: The impact of user-app interaction frequency

User-app interaction frequency	< Once per day	> Once per day
Number of volunteers	39	81
Average active days	25	29
Average computing hours	195	301
Computing hours per day	7.7	10.3
Average completed tasks	1459	2413
Tasks Per Day	57	83

The results, shown in Table 6.5, clearly demonstrate that volunteers who interact with the GEMCloud app more than once per day have 15% more active days and finished 65% more tasks within the first 60 days after joining the study com-

pared with volunteers who interact with the app less than once per day. This indicates a connection between high user-app interaction frequency and high volunteer computing performance in terms of more active days and more tasks completed. The reason could be that people who interact with the app more frequently are more involved with the app and therefore stay in the volunteer computing study longer and donate more computing power. Or it could be that people who truly want to contribute to volunteer computing projects have more user-app interaction based on their interest in the project. If the former is true, designing the volunteer computing app to allow volunteers to have more user-app interactions may prolong volunteers' participation and the number of finished tasks. If the latter is true, then improving the user experience will benefit those who truly want to contribute to volunteer computing projects and therefore may prolong their duration of participation and the number of completed tasks.

6.7 Conclusions

Volunteer computing on mobile devices not only provides inexpensive computing resources to solve complex computing projects in addition to volunteer computing on PCs, but also saves energy, which helps to reduce greenhouse gas emissions due to the generation of electricity from fossil fuels and therefore could potentially benefits the environment [160].

We have shown that people, at least from within the university community, are willing to participate in volunteer computing projects and provide a significant amount of computing power from their mobile devices. Participants' behaviors with using the volunteer computing app vary based on the device type and their personal preferences. The user experience with the volunteer computing app may have a significant impact on volunteers' participation and should be one of the most important things to consider when designing and developing volunteer com-

puting platforms for mobile devices. In addition, monetary prizes can be used for attracting more volunteers. This could introduce a different model compared with traditional volunteer computing.

7 Conclusions and Future Directions

7.1 Conclusions

This dissertation envisions an energy-efficient sensing and computing system. Health-monitoring and emotion-recognition applications were introduced as examples to demonstrate the potential practical value of such a system. To address the challenges in the design and development of such systems, we propose the WISP-Mote passive radio wake-up sensor device to reduce the energy consumption of wireless sensors, the BaNa pitch detection algorithm to improve the pitch data accuracy for emotion classification, and the GEMCloud platform to provide energy efficient cloud computing resources. To evaluate the feasibility of using mobile devices for cloud computing purposes and to guide future design of a mobile cloud computing system, we also conduct a public study using the GEMCloud platform.

The main contributions of this dissertation can be summarized as follows:

1. We develop WISP-Mote, a passive radio wake-up sensor node. WISP-Mote can be put into sleep mode for ultra low energy consumption and can be awakened remotely by a wake-up transmitter only when necessary. The performance evaluation of a network of WISP-Mote sensor nodes is provided.

Comparison of a WISP-Mote network and a network with traditional sensor nodes with different duty cycles shows using WISP-Mote can provide at least 4 times better energy efficiency with other performance metrics similar or better than a 0.5% duty-cycling network. Simulations of two example applications are also provided to show the benefits of using WISP-Mote in specific scenarios.

2. We develop BaNa, a noise resilient hybrid F_0 detection algorithm for speech and music. Evaluations on both speech and music data show that BaNa achieves the lowest GPE rate for most cases among the algorithms investigated for different types of background noise, and under different SNR levels from -10 dB to 20 dB. An implementation of the BaNa algorithm on Android is also provided. The tests of the BaNa Android app show that it is possible to use BaNa for real-time pitch detection on a mobile platform.

3. An extensive survey of the state-of-the-art mobile-cloud computing techniques is provided with summarization of the existing architectural designs. Different approaches that enhance application performance via cloud-based execution are compared. Research and technological challenges in different approaches are also highlighted.

4. We design and develop a mobile volunteer computing platform named GEMCloud. We evaluated the energy efficiency of using GEMCloud by providing comprehensive tests on various mobile devices and compared with the results of traditional workstations. The test results showed that using mobile devices as computing resources results in higher energy efficiency while providing similar computing power if enough devices can be harvested.

5. A public study on mobile volunteer computing using the GEMCloud platform is conducted. We showed the feasibility of using mobile devices for volunteer computing and studied volunteers' behavior based on the device type

and their personal preferences. Our study also showed that user experience is crucial for retaining the user loyalty and should be carefully considered when designing a mobile volunteer computing system. Monetary prizes can be used for attracting volunteers and in a different model of crowd-sourced computing system.

7.2 Future Directions

While several important technical challenges have been addressed in the dissertation, there are a few research directions that may lead to additional benefits for energy-efficient sensing and computing systems.

7.2.1 Passive Wake-up Radio

The range of the WISP-Mote wake-up radio is short, which limits its potential applications. Multiple energy harvesters or different types of energy harvesting techniques can be used to increase the wake-up range for wake-up radio sensor nodes [173, 174]. In addition, as mentioned in Chapter 2, a directional antenna and use of beam forming may also improve the range of passive wake-up radios. On the protocol side, in this dissertation, we introduced a simple but effective CSMA-based MAC protocol. According to the application scenario, a different MAC protocol may be needed in order to provide better performance.

7.2.2 Mobile Volunteer Computing

In a volunteer computing system, due to the heterogeneity of all the volunteered devices, to maintain a high quality of service is challenging. Depending on the requirements of the task provider, the task results may be needed in a timely manner. When assigning tasks to devices, the task scheduler needs to consider the

computing power and availability of the devices and the latency of the networks. With this information, a cost model may be built and used to determine the optimal strategy to assign tasks to volunteer computing devices. In cases when a task result may contain errors, a reliability control mechanism is needed. One solution would be sending redundant tasks to multiple devices and collecting at least two identical results before the system stops sending redundant tasks. The number of copies of each task to be sent is an important parameter that needs to be tuned ahead of time or on the fly in order to balance redundancy and reliability.

In cases where multiple client devices connect to the server via one device, e.g., a local cloudlet providing Wi-Fi connection, data processing techniques can be applied. One example is to use the cloudlet to buffer the returned results, preprocess them and send them to the GEMCloud server as a bundle. The total amount of time and energy cost to transmit all the results to the server may be reduced by using this approach.

In addition, the security of the system must be investigated to protect the privacy of mobile device owners and to guard against unauthorized access to the computing data and results. Unlike traditional cloud systems, as a resource provider, each individual mobile device in a volunteer computing system is not shared by multiple users. Only the app has control of the computing and communications. Therefore, similar to other available commercial apps, the volunteer computing app developer must convince the users that the app is not harmful. Making sure the app only has necessary accesses to the devices and making all accesses as transparent as possible will be helpful. In terms of the security of communications, strong encryptions and well-designed communication protocols are important to guarantee the security of data transfer and must be considered while designing the system.

Bibliography

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, pp. 14 –23, oct.-dec. 2009.

[2] "Emotional prosody speech and transcripts database from Linguistic Data Consortium (LDC)." `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2002S28`.

[3] "Background noise samples used to construct the AURORA noisy speech database." `http://www.ee.columbia.edu/~dpwe/sounds/noise/`.

[4] P. C. Bagshaw, S. M. Hiller, and M. A. Jack, "Enhanced pitch tracking and the processing of F0 contours for computer aided intonation teaching," in *Proceedings of Eurospeech*, pp. 1003–1006, 1993.

[5] F. Plante, G. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *Proceedings of Eurospeech*, pp. 837–840, 1995.

[6] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-Vision: Real-Time face recognition using a Mobile-Cloudlet-Cloud acceleration architecture," in *Proceedings of the 17th IEEE Symposium on Computers and Communications (IEEE ISCC 2012)*, (Cappadocia, Turkey), pp. 59–66, Jul 2012.

[7] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[8] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: a comparison of application models," *arXiv preprint arXiv:1107.4940*, 2011.

[9] "International Data Corporation (IDC)." `http://www.idc.com/`.

[10] "Motorola Backflip Specifications." `http://www.gsmarena.com/motorola_backflip-3079.php`.

[11] "Samsung Galaxy Nexus Specifications." `http://www.gsmarena.com/samsung_galaxy_nexus_i9250-4219.php`.

[12] "HTC One X Plus Specifications." `http://www.gsmarena.com/htc_one_x+-4976.php`.

[13] "Verizon Network Coverage." `http://network4g.verizonwireless.com/#/coverage`.

[14] "AT&T Network Coverage." `http://www.att.com/network/`.

[15] "PCMag.com tests of network speeds of major wireless carriers." `http://www.pcmag.com/article2/0,2817,2405658,00.asp`.

[16] H. Ba, N. Yang, I. Demirkol, and W. Heinzelman, "Bana: A hybrid approach for noise resilient pitch detection," in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pp. 369–372, 2012.

[17] H. Ba, L. Chen, W. Heinzelman, Z. Ignjatovic, and M. Sturge-Apple, "A Method For Signal Detection and Quantification of Heart Rate Data in Human Research: Insights From Engineering and Psychology."

SRCD Themed Meetings 2012. Link:http://www.ece.rochester.edu/~ba/publications/SRCD2012_He%20Ba.pdf.

[18] H. Ba, I. Demirkol, and W. Heinzelman, "Feasibility and benefits of passive RFID wake-up radios for wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference*, pp. 1 –5, Dec. 2010.

[19] H. Ba, I. Demirkol, and W. Heinzelman, "Passive wake-up radios: From devices to applications," *Elsevier Ad Hoc Networks*, 2013.

[20] H. Ba, J. Parvin, L. Soto, I. Demirkol, and W. Heinzelman, "Passive rfid-based wake-up radios for wireless sensor network," in *Wirelessly Powered Sensor Networks and Computational RFID* (J. R. Smith, ed.), pp. 113–129, Springer, 2013.

[21] R. Muraleedharan, I. Demirkol, O. Yang, H. Ba, S. Ray, and W. Heinzelman, "Sleeping techniques for reducing energy dissipation," in *The Art of Wireless Sensor Networks* (H. M. Ammari, ed.), Springer, 2013.

[22] N. Yang, H. Ba, W. Cai, I. Demirkol, and W. Heinzelman, "Bana: A noise resilient fundamental frequency detection algorithm for speech and music," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, pp. 1833–1848, Dec 2014.

[23] T. Soyata, H. Ba, W. Heinzelman, M. Kwon, and J. Shi, "Accelerating mobile cloud computing: A survey," in *Communication Infrastructures for Cloud Computing* (H. T. Mouftah and B. Kantarci, eds.), Hershey, PA, USA: IGI Global, 2013.

[24] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, "Mobile computing - a green computing resource," in *2013 IEEE Wireless Communications and Networking Conference (WCNC): SERVICES & APPLICATIONS (IEEE*

WCNC 2013 - SERVICES & APPLICATIONS), (Shanghai, P.R. China), pp. 4418–4423, Apr. 2013.

[25] Texas Instruments, "Application report for texas instruments cc430 wake-on-radio functionality." `http://www.ti.com/lit/an/slaa459a/slaa459a.pdf`, 2012. accessed: 07/2013.

[26] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the 2003 IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 30 – 41, May 2003.

[27] A. Sample, D. Yeager, P. Powledge, A. Mamishev, and J. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, pp. 2608 –2615, Nov. 2008.

[28] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 364 – 369, April 2005.

[29] I. Demirkol, C. Ersoy, and E. Onur, "Wake-up receivers for wireless sensor networks: benefits and challenges," *Wireless Communications, IEEE*, vol. 16, no. 4, pp. 88–96, 2009.

[30] B. Otis, Y. Chee, and J. Rabaey, "A 400 μW-RX, 1.6 mW-TX super-regenerative transceiver for wireless sensor networks," in *2005 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 396 –606 Vol. 1, Feb. 2005.

[31] N. Pletcher, S. Gambini, and J. Rabaey, "A 65 μW, 1.9 GHz RF to digital baseband wakeup receiver for wireless sensor nodes," in *IEEE Custom Integrated Circuits Conference, (CICC)*, pp. 539–542, Sept. 2007.

[32] P. Le-Huy and S. Roy, "Low-power 2.4 GHz wake-up radio for wireless sensor networks," in *IEEE International Conference on Wireless and Mobile Computing*, pp. 13 –18, Oct. 2008.

[33] S. von der Mark, R. Kamp, M. Huber, and G. Boeck, "Three stage wakeup scheme for sensor networks," in *SBMO/IEEE MTT-S International Conference on Microwave and Optoelectronics*, pp. 205 – 208, July 2005.

[34] J. Ansari, D. Pankin, and P. Mahonen, "Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications," in *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1 –5, Sept. 2008.

[35] B. V. d. Doorn, W. Kavelaars, and K. Langendoen, "A prototype low-cost wakeup radio for the 868 MHz band," *International Journal on Sensor Networks*, vol. 5, pp. 22–32, Feb. 2009.

[36] Austria Microsystems, "AS3933 3-D low frequency rf wake-up receiver." `http://www.austriamicrosystems.com/Wake-up-receiver/AS3933`, 2010.

[37] L. Gu and J. A. Stankovic, "Radio-triggered wake-up for wireless sensor networks," *Real-Time Systems Journal*, vol. 29, pp. 157–182, March 2005.

[38] A. G. Ruzzelli, R. Jurdak, and G. M.P. O'Hare, "On the RFID wake-up impulse for multi-hop sensor networks," in *1st ACM Workshop on Convergence of RFID and Wireless Sensor Networks and their Applications (SenseID)*, ACM, Nov. 2007.

[39] R. Jurdak, A. Ruzzelli, and G. O'Hare, "Multi-hop RFID wake-up radio: Design, evaluation and energy tradeoffs," in *Proceedings of 17th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1 –8, Aug. 2008.

[40] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, (New York, NY, USA), pp. 95–107, ACM, 2004.

[41] "IEEE 802.15.4MAC/Phy standard for low-rate wireless personal area networks(LR-WPANs)." `http://www.ieee802.org/15/pub/TG4.html`.

[42] "Impinj RFID reader." `http://www.impinj.com/products/rfid-reader.aspx`.

[43] EPCglobal, "UHF class-1 generation-2 standard." `http://www.epcglobalinc.org/standards/uhfc1g2/\uhfc1g2_1_2_0-standard-20080511.pdf`.

[44] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, pp. 25 – 33, Jan.-March 2006.

[45] "Tmote Sky datasheet." `http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf`.

[46] M. Holland, R. Aures, and W. Heinzelman, "Experimental investigation of radio performance in wireless sensor networks," in *2nd IEEE Workshop on Wireless Mesh Networks*, pp. 140 –150, Sept. 2006.

[47] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ASPLOS-X, (New York, NY, USA), pp. 96–107, ACM, 2002.

[48] R. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh, "Citysense: An urban-scale wireless sensor network and testbed,"

in *IEEE Conference on Technologies for Homeland Security*, pp. 583 –588, May 2008.

[49] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 882 –901, Feb. 2009.

[50] F. Gil-Castineira, F. Gonzalez-Castano, R. Duro, and F. Lopez-Pena, "Urban pollution monitoring through opportunistic mobile sensor networks based on public transport," in *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pp. 70 –74, July 2008.

[51] Omni-ID, "Omni-ID RFID tags." `http://www.omni-id.com/`.

[52] T. Vogt and E. Andre, "Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 474–477, 2005.

[53] B. Cardozo and R. Ritsma, "On the perception of imperfect periodicity," *Audio and Electroacoustics, IEEE Trans. on*, vol. 16, no. 2, pp. 159 – 164, 1968.

[54] J. H. Jeon, W. Wang, and Y. Liu, "N-best rescoring based on pitch-accent patterns," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 732–741, 2011.

[55] C. Wang, *Prosodic Modeling for Improved Speech Recognition and Understanding*. PhD thesis, Massachusetts Institute of Technology, 2001.

[56] Z.-H. Ling, Z.-G. Wang, and L.-R. Dai, "Statistical modeling of syllable-level F0 features for hmm-based unit selection speech synthesis," in *Proceedings of*

International Symposium on Chinese Spoken Language Processing, pp. 144–147, 2010.

[57] S. Sakai and J. Glass, "Fundamental frequency modeling for corpus-based speech synthesis based on a statistical learning technique," in *Proc. of IEEE ASRU*, pp. 712–717, 2003.

[58] J. Woodruff and D. L. Wang, "Binaural detection, localization, and segregation in reverberant environments based on joint pitch and azimuth cues," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 806–815, 2013.

[59] O. W. Kwon, K. Chan, J. Hao, and T. W. Lee, "Emotion recognition by speech signals," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[60] F. Al Machot, A. H. Mosa, K. Dabbour, A. Fasih, C. Schwarzlmuller, M. Ali, and K. Kyamakya, "A novel real-time emotion detection system from audio streams based on bayesian quadratic discriminate classifier for adas," in *Nonlinear Dynamics and Synchronization 16th Intl Symposium on Theoretical Electrical Engineering*, 2011.

[61] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "EmotionSense: a mobile phones based adaptive platform for experimental social psychology research," in *Proceedings of the 12th int. conference on Ubiquitous computing*, pp. 281–290, 2010.

[62] K. Chang, D. Fisher, and J. Canny, "AMMON: A Speech Analysis Library for Analyzing Affect, Stress, and Mental Health on Mobile Phones," in *2nd Intl Workshop on Sensing Applications on Mobile Phones*, 2011.

[63] Y. Yang, C. Fairbairn, and J. F. Cohn, "Detecting depression severity from vocal prosody," *Affective Computing, IEEE Trans. on*, vol. 4, no. 2, pp. 142–150, 2013.

[64] J. P. Bello, G. Monti, and M. Sandler, "Techniques for automatic music transcription," in *Intl Symposium on Music Information Retrieval*, pp. 23–25, 2000.

[65] M. Antonelli, A. Rizzi, and G. Del Vescovo, "A query by humming system for music information retrieval," in *Intelligent Systems Design and Applications (ISDA), 10th Intl Conference on*, pp. 586 – 591, 2010.

[66] S. Kim, E. Unal, and S. Narayanan, "Music fingerprint extraction for classical music cover song identification," in *Multimedia and Expo, 2008 IEEE Intl. Conference on*, pp. 1261 –1264, 2008.

[67] P. Cariani, "Neural Representation of Musical Pitch - MIT OpenCourseWare." `http://ocw.mit.edu/courses/health-sciences-and-technology/hst-725-music-perception-and-cognition-spring-2009/lecture-notes/MITHST_725S09_lec04_pitch.pdf`, 2009.

[68] A. M. Noll, "Cepstrum pitch determination," *Journal of the Acoustical Society of America*, vol. 41, pp. 293–309, 1967.

[69] M. R. Schroeder, "Period histogram and product spectrum: New methods for fundamental frequency measurement," *Journal of the Acoustical Society of America*, vol. 43, pp. 829–834, 1968.

[70] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the Institute of Phonetic Sciences 17*, pp. 97–110, 1993.

[71] H. Ba, N. Yang, I. Demirkol, and W. Heinzelman, "BaNa: A hybrid approach for noise resilient pitch detection," in *IEEE Workshop on Statistical Signal Processing*, pp. 369 –372, 2012.

[72] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley, "Average magnitude difference function pitch extractor," *Audio, Speech, and Language Processing, IEEE Trans. on*, pp. 353 –362, 1974.

[73] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 111, pp. 1917–1930, 2002.

[74] J. Liu, T. F. Zheng, J. Deng, and W. Wu, "Real-time pitch tracking based on combined SMDSF," in *Proc. of Interspeech*, pp. 301–304, 2005.

[75] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," *Speech Coding and Synthesis*, 1995.

[76] L. R. Rabiner and R. W. Schafer, *Theory and Application of Digital Speech Processing*. Pearson, 2011.

[77] T. W. Parsons, "Separation of speech from interfering speech by means of harmonic selection," *Journal of the Acoustical Society of America*, vol. 60, pp. 911–918, 1976.

[78] X. Chen and R. Liu, "Multiple pitch estimation based on modified harmonic product spectrum," in *Proceedings of Intl Conference on Information Technology and Software Engineering*, pp. 271–279, 2013.

[79] S. Gonzalez and M. Brookes, "A pitch estimation filter robust to high levels of noise (PEFAC)," in *Proc. European Signal Processing Conf., Barcelona, Spain*, 2011.

[80] Z. Jin and D. Wang, "HMM-based multipitch tracking for noisy and reverberant speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 5, pp. 1091–1102, 2011.

[81] F. Huang and T. Lee, "Pitch estimation in noisy speech using accumulated peak spectrum and sparse estimation technique," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, no. 1, pp. 99 –109, 2013.

[82] M. Wu, D. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Speech and Audio Processing, IEEE Trans. on*, vol. 11, no. 3, pp. 229–241, 2003.

[83] W. Chu and A. Alwan, "SAFE: A statistical approach to F0 estimation under clean and noisy conditions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 20, no. 3, pp. 933 –944, 2012.

[84] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. of the 13th Int. Conference on Digital Audio Effects*, 2010.

[85] P. De La Cuadra, A. Master, and C. Sapp, "Efficient pitch detection techniques for interactive music," in *Proceedings of the Int. Computer Music Conference, La Habana*, 2001.

[86] Thomas O'Haver, Command-line findpeaks MATLAB function. `http://terpconnect.umd.edu/~toh/spectrum`.

[87] P. van Alphen and D. van Bergem, "Markov models and their application in speech recognition," in *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, pp. 1–26, 1989.

[88] D. Iskra, B. Grosskopf, K. Marasek, H. van den Huevel, F. Diehl, and A. Kiessling, "SPEECON - speech databases for consumer devices:

Database specification and validation," in *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pp. 329–333, 2002.

[89] B. Kotnik, H. Höge, and Z. Kacic, "Evaluation of pitch detection algorithms in adverse conditions," in *Proceedings of the Third International Conference on Speech Prosody*, pp. 149–152, 2006.

[90] I. Luengo, I. Saratxaga, E. Navas, I. Hernáez, J. Sanchez, and I. naki Sainz, "Evaluation of pitch detection algorithms under real conditions," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1057–1060, 2007.

[91] G. Seshadri and B. Yegnanarayana, "Performance of an event-based instantaneous fundamental frequency estimator for distant speech signals," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, pp. 1853–1864, 2011.

[92] "CMU Arctic Database." `http://www.festvox.org/cmu_arctic/`.

[93] "Generated noisy speech data and BaNa source code, WCNG website." `http://www.ece.rochester.edu/projects/wcng/project_bridge.html`.

[94] A. P. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "NOISEX-92 study on the effect of additive noise on automatic speech recognition." `http://spib.ece.rice.edu/spib/data/signals/noise/`, 1992.

[95] L. R. Rabiner, M. J. Cheng, A. E. Osenberg, and C. A. McGonegal, "A comparative performance study of several pitch detection algorithms," *Acoustics, Speech, and Signal Processing, IEEE Trans. on*, vol. 24, pp. 399 – 418, October 1976.

[96] M. Wohlmayr, M. Stark, and F. Pernkopf, "A probabilistic interaction model for multipitch tracking with factorial hidden Markov models," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 4, pp. 799–810, 2011.

[97] O. Babacan, T. Drugman, N. d'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 7815–7819, 2013.

[98] "Download page for the SAFE toolkit." `http://www.ee.ucla.edu/~weichu/safe/`.

[99] X. Huang, A. Acero, and H.-W. Hon, *Spoken language processing*, vol. 15. Prentice Hall PTR New Jersey, 2001.

[100] D. Pearce, H.-G. Hirsch, and E. E. D. Gmbh, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, pp. 29–32, 2000.

[101] "Source code for the YIN algorithm." `http://audition.ens.fr/adc/`.

[102] "Source code for the Praat algorithm." `http://www.fon.hum.uva.nl/praat/`.

[103] "Source code for the PEFAC algorithm included in the VOICEBOX toolkit." `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`.

[104] "Source code for the Wu algorithm." `http://www.cse.ohio-state.edu/pnl/software.html`.

[105] M. G. Christensen and A. Jakobsson, *Multi-Pitch Estimation*. Morgan & Claypool Publishers, 2009.

[106] S. A. Raczynski, E. Vincent, and S. Sagayama, "Separation of speech from interfering speech by means of harmonic selection," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 1830–1840, 2013.

[107] M. Wu, D. L. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 11, pp. 229–241, 2003.

[108] J. Wu, E. Vincent, S. A. Raczynski, T. Nishimoto, N. Ono, and S. Sagayama, "Polyphonic pitch estimation and instrument identification by joint modeling of sustained and attack sounds," *IEEE Journal of Selected Topics in Signal Process.*, vol. 5, pp. 1124–1132, 2011.

[109] "Freesound website for short pieces of music download." `http://www.freesound.org/`.

[110] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 18, no. 8, pp. 2121–2133, 2010.

[111] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, pp. n/a–n/a, 2011.

[112] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Gener. Comput. Syst.*, vol. 29, pp. 84–106, Jan. 2013.

[113] "Amazon Web Services." `http://aws.amazon.com`.

[114] "Windows Azure: Microsoft Cloud Platform." `http://www.microsoft.com/windowazure`.

[115] "Google Cloud Platform." `https://cloud.google.com/`.

[116] "Nvidia Tegra 3." http://www.nvidia.com/object/tegra-3-processor.html.

[117] "GeForce 600 Series." http://en.wikipedia.org/wiki/GeForce_600_Series.

[118] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: Avoiding the power wall with low-leakage, STT-MRAM based computing," in *Proceedings of the International Symposium on Computer Architecture*, vol. 38, (Saint-Malo, France), pp. 371–382, Jun 2010.

[119] "DOCSIS." http://en.wikipedia.org/wiki/DOCSIS.

[120] "Qualcomm Snapdragon." http://www.qualcomm.com/snapdragon.

[121] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, (New York, NY, USA), pp. 49–62, ACM, 2010.

[122] "Internet of things." http://en.wikipedia.org/wiki/Internet_of_Things.

[123] "HIPAA." http://www.hhs.gov/ocr/privacy/index.html.

[124] C. S. Pattichis, E. Kyriacou, S. Voskarides, M. S. Pattichis, R. Istepanian, C. N. Schizas, and C. S. Pattichis, "Wireless telemedicine systems: An overview," *IEEE Antennas & Propagation Magazine*, vol. 44, pp. 143–153, 2002.

[125] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mob. Netw. Appl.*, vol. 12, pp. 113–127, Mar. 2007.

[126] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *Netwrk. Mag. of Global Internetwkg.*, vol. 22, pp. 26–33, July 2008.

[127] "Microchip's 32-bit Microcontrollers." `http://www.microchip.com/pagehandler/en-us/family/32bit/`.

[128] Wikipedia, "Advanced encryption standard," 2012. `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`.

[129] NIST, "Advanced encryption standard (aes)," 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[130] M. Ali, "Green cloud on the horizon," in *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, (Berlin, Heidelberg), pp. 451–459, Springer-Verlag, 2009.

[131] E. E. Marinelli, "Hyrax: Cloud computing on mobile devices using mapreduce," Master's thesis, Carnegie Mellon University, 2009.

[132] "Native Boinc for Android." `http://nativeboinc.org/site/uncat/start`.

[133] J. R. Eastlack, "Extending volunteer computing to mobile devices," Master's thesis, New Mexico State University, 2011.

[134] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, (Washington, DC, USA), pp. 4–10, IEEE Computer Society, 2004.

[135] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proceedings of the 12th conference on Hot topics*

in operating systems, HotOS'09, (Berkeley, CA, USA), pp. 8–8, USENIX Association, 2009.

[136] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, EuroSys '11, (New York, NY, USA), pp. 301–314, ACM, 2011.

[137] E. Chen, S. Ogata, and K. Horikawa, "Offloading android applications to the cloud without customizing android," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pp. 788 –793, march 2012.

[138] E. Y. Chen and M. Itoh, "Virtual smartphone over ip," in *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pp. 1 –6, june 2010.

[139] T. Soyata, R. Muraleedharan, S. Ames, J. H. Langdon, C. Funai, M. Kwon, and W. B. Heinzelman, "Combat: mobile cloud-based compute/communications infrastructure for battlefield applications," in *Proceedings of SPIE*, vol. 8403, May 2012.

[140] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of cloudlets on interactive mobile cloud applications," in *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*, pp. 123 – 132, sept. 2012.

[141] M. Corson, R. Laroia, J. Li, V. Park, T. Richardson, and G. Tsirtsis, "Toward proximity-aware internetworking," *Wireless Communications, IEEE*, vol. 17, pp. 26 –33, december 2010.

[142] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, pp. 129–140, Feb. 2013.

[143] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, MCS '12, (New York, NY, USA), pp. 29–36, ACM, 2012.

[144] T. O. Alliance, "Osgi - the dynamic module system for java," 2012. `http://www.osgi.org/`.

[145] J. S. Rellermeyer, G. Alonso, and T. Roscoe, "R-osgi: distributed applications through software modularization," in *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, Middleware '07, (New York, NY, USA), pp. 1–20, Springer-Verlag New York, Inc., 2007.

[146] H. Flores, S. N. Srirama, and C. Paniagua, "A generic middleware framework for handling process intensive hybrid cloud services from mobiles," in *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '11, (New York, NY, USA), pp. 87–94, ACM, 2011.

[147] "Android Cloud to Device Messaging Framework." `https://developers.google.com/android/c2dm/`.

[148] "Apple Push Notification Service." `http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html`.

[149] D. B. Hoang and L. Chen, "Mobile cloud for assistive healthcare (mocash)," in *Proceedings of the 2010 IEEE Asia-Pacific Services Computing Confer-*

ence, APSCC '10, (Washington, DC, USA), pp. 325–332, IEEE Computer Society, 2010.

[150] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: the challenge of intermittent connectivity," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, MCC '12, (New York, NY, USA), pp. 23–28, ACM, 2012.

[151] D. T. Hoang, D. Niyato, and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pp. 3145 –3149, april 2012.

[152] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies, and M. Satyanarayanan, "The impact of mobile multimedia applications on data center consolidation," *Technical Report CMU-CS-12-143, School of Computer Science, Carnegie Mellon University*, October 2012.

[153] W.-F. Alliance, "WPA2." `http://www.wi-fi.org/knowledge-center/glossary/wpa2%E2%84%A2`.

[154] B. Technologies, "Security Architecture for the Internet Protocol," 2005. `http://tools.ietf.org/pdf/rfc4301.pdf`.

[155] "Asus Nexus 7 Android tablet." `http://www.asus.com/Tablet/Nexus/Nexus_7/`.

[156] "Android." `http://www.android.com/`.

[157] "Watts up? PRO ES Watt meter." `https://www.wattsupmeters.com/secure/products.php?pn=0`.

[158] L. Gong, "Jxta: a network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95, 2001.

[159] G. Fedak, C. Germain, V. Neri, and F. Cappello, "Xtremweb: a generic global computing system," in *Proc. of IEEE/ACM CCGrid*, 2001.

[160] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, "Mobile computing - a green computing resource," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pp. 4451–4456, April 2013.

[161] "BOINC website." `http://boinc.berkeley.edu/index.php`.

[162] "Samsung Galaxy S5." `http://www.samsung.com/us/galaxy-s-5-the-next-big-thing-is-here/`.

[163] "Apple iPhone 6." `https://www.apple.com/iphone-6/`.

[164] "Nexus 9 Android tablet." `http://www.google.com/nexus/9/`.

[165] "IDC Press Release - worldwide smartphone shipments, Jan. 27, 2014." `http://www.idc.com/getdoc.jsp?containerId=prUS24645514`.

[166] "IDC Press Release - worldwide tablet shipments, Jan. 29, 2014." `http://www.idc.com/getdoc.jsp?containerId=prUS24650614`.

[167] M. Arslan, I. Singh, S. Singh, H. Madhyastha, K. Sundaresan, and S. Krishnamurthy, "Cwc: A distributed computing infrastructure using smartphones," *Mobile Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[168] M. Black and W. Edgar, "Exploring mobile devices as grid resources: Using an x86 virtual machine to run boinc on an iphone," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pp. 9–16, Oct 2009.

[169] "HTC Power To Give web page." `http://www.htc.com/us/go/power-to-give/`.

[170] "Samsung Power Sleep web page." http://www.samsung.com/at/microsite/powersleep/.

[171] "Average Retail Price of Electricity to Ultimate Customers." http://www.eia.gov/electricity/monthly/pdf/epm.pdf.

[172] "Maxmind GeoIP web services." http://dev.maxmind.com/.

[173] L. Chen, H. Ba, W. Heinzelman, and A. Cote, "RFID range extension with low-power wireless edge devices," in *2013 International Conference on Computing, Networking and Communications, Wireless Ad Hoc and Sensor Networks Symposium (ICNC'13 - WAHS)*, (San Diego, USA), pp. 524–528, Jan. 2013.

[174] L. Chen, S. Cool, H. Ba, W. Heinzelman, I. Demirkol, U. Muncuk, K. Chowdhury, and S. Basagni, "Range extension of passive wake-up radio systems through energy harvesting," in *IEEE ICC 2013 - Ad-hoc and Sensor Networking Symposium (ICC'13 AHSN)*, (Budapest, Hungary), pp. 1534–1539, June 2013.