ENERGY MEASUREMENTS AND ANALYSIS TO UNDERSTAND COMPUTING SYSTEMS AND NETWORKS

A Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Dong Han

December 2014

ENERGY MEASUREMENTS AND ANALYSIS TO UNDERSTAND COMPUTING SYSTEMS AND NETWORKS

Dong Han

APPROVED:

Omprakash Gnawali, Chairman
Dept. of Computer Science

Edgar Gabriel
Dept. of Computer Science

Jaspal Subhlok
Dept. of Computer Science

Ricardo Vilalta
Dept. of Computer Science

Abhishek Sharma
NEC Labs America, Inc.

Dean, College of Natural Sciences and Mathematics

To my Mom, Dad, Wife and my Son.

Acknowledgments

First and foremost, I am deeply indebted to my advisor Professor Omprakash Gnawali, for making my last four years of graduate school one of the best periods of my life. Dr.Gnawali is one of the smartest people I know, he even let me steal an enormous amount of his time, and always there when I need a hand supporting me, helping me. Dr.Gnawali has been supportive and has provided insightful discussions about the research, he has also given me the freedom to pursue various projects without objection. I appreciate all his contributions of time, ideas, and funding to make my PhD experience productive and stimulating. He is my primary resource for getting my science questions answered and was instrumental in helping me crank out this dissertation. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the PhD pursuit. I also have to thank the members of my PhD committee, Professor Jaspal Subhlok, Professor Edgar Gabriel, Professor Ricardo Vilalta, and Dr.Abhishek Sharma for their helpful dissertation advice and suggestions in general, especially thanks Dr.Sharma for guiding me in my last project. My study was partially supported by the National Science Foundation under grant no. IIS-1111507, and a generous gift from Cisco.

I will forever be thankful to my previous advisors, Professor XiaoJing Yuan and Professor Rong Zheng. Dr.Yuan has been helpful in providing advice many times during my first a few years during my PhD study. Dr.Zheng is an excellent advisor who encouraged and expected me to think more independently about our experiments and results. I am also thankful for the excellent example Dr.Zheng has provided as a scientist.

I am lucky to meet all the present and previous members (co-founders) of the Networked Systems Laboratory: Anirup Dutta, Qiang Li, Varun Prakash, Shengrong Yin, Hessam Mohammadmoradi, Milad Heydariaan and Srijani Mukherjee. I have had great fun working with all of them and learning from their different research styles. Some of my best times in last a few years were spent with them in our lab and on our hiking trips.

My time at UH was made enjoyable in large part due to the many friends that became a part of my life. I am grateful for the time spent with my friends, and for many other people and memories. I also thank all of them for providing support and friendship that I needed.

Many friends have helped me stay sane through these difficult years. I would like to acknowledge Rui Jiang, Dr.ChengWei Huang, Dr.Wei Sun, Dr.Zhen Zheng, WenBo Liu, Dr.GuanBo Zheng, DongYu Ang, Lei Pan, ZiSong Jin, Ying Jiang. I greatly value their friendship and I deeply appreciate their belief in me. Particularly, I would like to acknowledge Dr.ChengWei Huang for the many valuable discussions.

Lastly, I would like to thank my family for all their love and encouragement. For my parents who raised me up with a love of science and supported me in all my pursuits. They has been non-judgmental of me and instrumental in instilling confidence. I love them so much, and I would not have made it this far without them. And most of all for my loving, supportive, encouraging, and patient wife Quan Wang whose faithful support during the final stages of this PhD is so appreciated. I truly thank Quan for sticking by my side, even when I was irritable and depressed. I feel that what we both learned a lot about life and strengthened our commitment

and determination to each other and to live life to the fullest. Thank you. Finally, I sincerely appreciate my son, Aiden, has endured with me shoulder to shoulder during the last one month before my defense.

ENERGY MEASUREMENTS AND ANALYSIS TO UNDERSTAND COMPUTING SYSTEMS AND NETWORKS

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Dong Han

December 2014

Abstract

In this work, we design techniques to use energy instrumentation to study the health and workloads of a computing system. Analysis of energy consumption with the goal of understanding the computing system in an uncontrolled environment is an open research area. The main challenge is to infer the system state only from discrete time-series energy data.

We have analyzed power-consumption data on computing systems. Our focus is on how to distinguish various events and how to reveal the health of the system. In addition to studying the data collected in a laboratory environment, we have analyzed 3-years of continuous energy measurements of a large enterprise computing environment. We can infer system health, failures, activities, and trends from energy data.

We have investigated power-consumption data of networking systems, especially the low-power wireless networks. We designed two novel features called High-Power-Length-Counter and High-Power-Overlap-Counter. We evaluated our approaches on three real-world testbeds and various network scenarios. We found that these features reveal network protocols, application workloads, and routing topology from energy data alone. This information was not possible to reveal only from energy data prior to this work.

The contributions of this work are: (a) Techniques to analyze and reveal health information of computing system. The energy profiling during boot up, idle and failure exposes operating states of the system. (b) Design of two novel features that use fine-grained energy-instrumentation data on networking systems, to identify routing protocol, infer network topology, and determine application workloads. Our proposed

features can achieve 97% accuracy when used to identify the routing protocols, and infer the network topology with 98% accuracy. (c) Identification of sources of waste in computing systems. We found that, at least 60% of energy consumed per day was wasted when the collection of computers we studied were left in idle state in a computer lab environment.

Contents

List of Figures

Chapter 1

Introduction

1.1 Energy-Instrumentation Research

Recently, researchers from various background have been attracted to the study of energy consumption of computers. Computers run on electricity. The cost of electricity is not small, and sometimes it is a big burden on the environment. And sometimes electrical power supply systems can be at risk, when our computers work improperly. Therefore, understanding the normal and abnormal power consumption behavior of our computer system is important for both environmental protection and safety. Furthermore, by analyzing the energy consumption of the computing system, we may find specific patterns when software fault occurs and thus improve the reliability of the computing system.

Currently, tremendous amounts of energy are wasted in computing system[65].

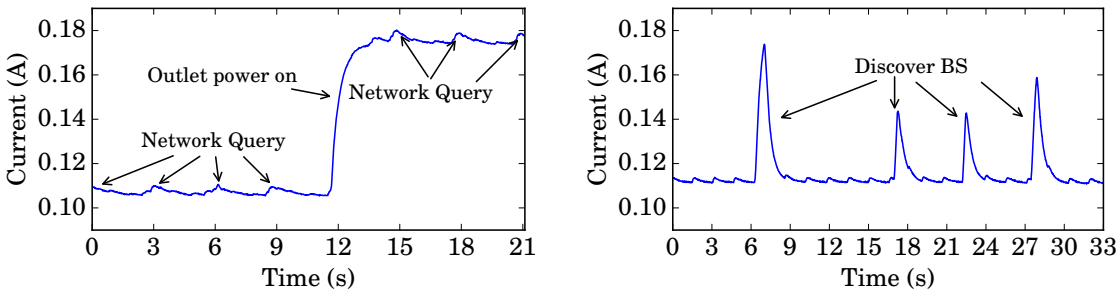
Saving energy on computers may contribute to protecting the environment from global warming. New algorithms and mechanics are needed for making the current computing systems run more efficiently. There has been recent interest in understanding power draw of a collection of machines, e.g., in data centers [22, 51, 20, 12] or computers in a building [45, 28, 62]. These studies provide measurement-based models of computer power use in buildings. These models can be used to test new approaches to make computing infrastructures in buildings more energy-efficient. While these datasets are extremely useful to the community, most of their work were concentrating on understand the power consumption based on given types of workload. Thus, studying the power profile from another direction, e.g., revealing the workloads from the power consumption can be a valuable asset for the computing research community.

Energy instrumentation and its analysis could also serve as a tool to gain a better understanding of the network, either as a complementary instrumentation or when other types of instrumentation are not available or not convenient. In the first order, a network that is expected to use a few mW of power but is found to be using a few tens of mW of power, we could infer that there is something unusual and worthy of further diagnosis. Even when we do not have access to the source code (for example, nodes that we might purchase commercially), we could use energy instrumentation to validate that the activities performed by the nodes are consistent with their expected behavior. Thus, energy instrumentation can be used to validate sensor network operations when other instrumentation is costly or unavailable, such as reverse engineering on closed source devices.

We motivate these possibilities with two examples from real-world devices. Monster PowerControl is a commercially available smart plug to which appliances can be connected. The outlets can be controlled using a smartphone. The system has a base station connected to a home Internet router and other user devices communicate with the base station on the 900MHz band. We measure the power used by the plugs to reveal four properties of the system without any source code.

First, we can tell the power state (On/Off) of plug outlet from the energy draw. As shown in Figure 1.1(a), the plug itself consumes 0.10A - 0.11A when the outlet is in power off state. After we turn on the outlet at 12s, the current increases to 0.17A - 0.18A even without any appliance plugged into the smart plug. Second, the current draw gives hints about the episodic communication between the plug and base station. Figure 1.1(a) has arrows corresponding to the radio communication events (verified using the timing of the link LED). Third, we can verify that the devices query the base station for new commands rather than the base station pushing messages to the devices: the episodic crests continue even when we turn off the base station. Fourth, the current draw can also give hints about the wireless connectivity between the user devices and the base station. We moved the user devices farther from the base station until it got disconnected. Figure 1.1(b) shows the spikes in current that appear when the base station is not reachable indicating the node scanning for the base station.

We may not be able to derive such detailed insights from power profiling of all smart devices. Belkin WeMo switch, for instance, directly connects to a household WiFi router. In this device, we do not observe a periodic query pattern but we



(a) Filtered energy draw of smart plug (b) Energy draw after its lost connection

Figure 1.1: Current draw of smart plug can reveal its working states

found that whenever we send a command from a smartphone, there is a spike in current on the device suggesting a proposal in which the Cloud pushses commands to the device. Thus, depending on the device and the system, different amounts of information about the smart devices and their systems, protocols, and applications may be revealed by power profiling.

1.2 Addressed Research Questions

On the computing system and networks, it is hard to study and understand the detailed working states on the system, including the software and hardware activities. Furthermore, it is difficult to accurately predict the system failure in advance, to prevent any data and economic loss. The existing solution by using software monitoring is able to monitor the software activities on the computing system, but there is not enough information to monitor every hardware components in the system. On the networking system, besides online debug each device in the network, there is

hardly a method to efficiently reveal the current working states each single device. In this dissertation, we consider, broadly speaking, how the health information of the computing system can be reflected on the energy consumption data.

The motivation of this work is to provide a non-intrusive method to study what is happening on the system, by analyzing the power consumption data. Every electronic system consumes energy, so the power consumption data is always available, even during the period when the system fails to run. Energy instrumentation and its analysis could also serve as a tool to gain a better understanding of the system, either as a complementary instrumentation or when other types of instrumentation are not available or not convenient.

The immediate implication of this work is that demonstration of the energy instrumentation can not only be used to classify the workloads on the system, but also can be used to detect various events happening on the system, such as system boot up, failure and workload change.

The long term implications of this work are to use this work as a starting point, to build a comprehensive diagnostic system by monitoring the energy consumption on the computing systems. Like software monitoring, energy monitoring can automatically monitor the activities and workloads on the system. Furthermore, the usage of the energy also can be used to diagnose the hardware performance, even to predict and prevent the system failure by recognizing the energy patterns.

In particular, we address how to understand the fine-grained energy consumption data on computing systems, and consider how to extract the features from the energy

data to reveal the detailed workloads on a networking systems. To that end, we address the following research topics in this dissertation:

- **How to design and implement an instrumentation system on computing systems as a foundation for energy study.**

  Instruments should be deployed in the way that power draw can be measured for a long period of time. We investigated the possibility of using smart plugs, hall-sensor and commercialized power strip to record the energy consumption behavior of various systems. A general-purpose power meter should be able to gather data for inferring various system properties and should be programmerable to fit various real-world situations.

- **How to detect computing activities solely from power activities, and based on which how to avoid energy waste and reveal the performance changes in computing systems.**

  Previous studies on system recognition adopt various ways to infer the properties of a system. Many system test methods require intrusive sensors. From our point of view, the energy consumption behavior alone may reveal certain important characteristic of computer systems. For instance, power draw data collected from smart plugs may provide us useful information to infer the system status, especially under abnormal system behavior. When we successfully gather enough data and correctly infer system behavior, we are able to explore the possibilities of predicting such system behavior. The most common and the most important system status is perhaps the computer failure case, such as blue

screen for Windows and kernel panic for Linux. By predicting the computer failure we may advance our research in next step to be more practical in real world applications. We may be able to improve the reliability and efficiency of massive computing services.

- **How to extend the energy analysis on computing systems to the network system, in which protocols can be recognized, its workload can be evaluated, and the detail routing topology can be disclosed.**

  Based on the further exploration of energy instruments we are able to raise the same question for network systems, e.g., wireless sensor network. The energy efficiency is a key character for wireless sensor network. By analyzing the power draw of each nodes in the network we may improve the design of energy efficient sensor network. Furthermore, the system status may also be inferred for the network. For instance, protocols as well as certain network activities can be inferred. However there are few existing methods on this topic. We proposed to explore the possibilities from the machine learning and signal processing perspectives. We hope to be able to evaluate, validate, predict various information for the network level system, so that we may deploy robustness, efficiency, environment friendly for future networking community.

1.3 Research Contributions

The key research contributions of this dissertation as summarized as follows:

Understanding Desktop Energy Footprint in an Academic Computer Lab

We design and deploy power-usage and user-activity instrumentation in an academic computer lab, and we present a large dataset describing computing power data in a shared homogeneous computing environment. Our measurement-study logged power use and user activity on 22 computers in one of the computer labs for over 30 days. We collected 59.6 million power readings and 220.3 million user activity readings. In addition to understanding the relation between user activities and power draw, in particular, we identify at least 60% of energy consumed per day was wasted when the computers were left in idle state in a computer lab environment

Energy Measurements and Analysis to Understand Computing Systems

We implement the energy instrument system to collect and analyze power consumption data on computing systems. We present energy patterns to reflect the health information of computing system, in particular during the system boot up, idle and failure periods. Algorithms are designed to identify these events from continuous energy data. Our proposed algorithms detected 270 Pre-OS boot up events among an 18-month power trace, and accurately detected the 8 out of 8 failures from a 19-month power trace. Analysis over 30 months of data reveals the energy consumption changes over time, and the temperature value affects the health of the computing systems.

Revealing Protocol Information and Activity from Energy Instrumentation in Wireless Network

We study the design of classification features based on energy data with the goal of revealing protocol, network, and application information. Our experiments on three real-world testbeds increases our confidence in the accuracy of the results, which include evaluating the proposed features across multiple protocols, network topologies, and application workloads. The classification evaluations with those features can identify the routing protocol with more than 90% accuracy and application workloads with 85% accuracy. We also demonstrate how to reveal the routing topology in the network, including the next hop for each node, with just energy instrumentation, with 98% accuracy.

Chapter 2

Related Works

There is a large amount of work in energy measurement and analysis. In this chapter, we describe the existing work related to energy instrumentation on computers, servers and networks, from the perspective of the device level and the system level.

2.1 Energy Profiling on Computing Systems

In this section we give an overview of research related to wireless power meter, measurement of energy consumption, and machine idle proportion in enterprise data centers.

2.1.1 Power-Meter Design

Since power consumption has become a significant concern in the development of all kinds of computing laboratories and data centers, much progress and various measurement methods have been designed to measure the power used by the systems, such as the work of Cornil et al. [11], who use the Fluke ampere meter to measure the current from the power supply. Serra et al. [53] use an AD7757 IC chip based on the shunt resistor from Analog Devices for measuring electric power. Lifton et al. [43], introduced the MIT Plug sensor network, which embodied the idea of designing sensor nodes to seamlessly become a part of their environment. Koller et al. proposed an application-aware power meter for data centers, which addressed the challenges of estimating the power drawn by the whole center.

Unlike tradional power meters, where the results can only be displayed on local LEDs or saved as data onto flash drives and read later, wireless power meters can transmit the power readings to a remote database to be processed later. There are several wireless powers meter that have been designed in recent years, such as ACme [33] or commercial meters such as *The UFO Power Center*. Jiang, et al. [34] deployed ACme meters for high-fidelity monitoring of electrical usage in a building. Krioukov, et al. [38] presented a personalized smartphone application designed to control the lighting, heating and cooling in user's vicinity, by using ACme for the sensing and actuation. PowerNet [36] is a platform from Stanford University used for collecting, viewing, and analyzing plug-level power data collected. The goal of PowerNet is aiming at answering questions about total power usage, variation, and efficiency. In our work, we use PowerNet meters to collect power readings.

2.1.2 Computer Energy Measurement Studies

Many measurement studies have contributed to understanding energy consumption in commercial buildings and households. [35] reported a hybrid sensor network based on PowerNet for monitoring the power and utilization of computing systems. Their 3-month monitoring and measurements revealed the IT-related power waste and savings opportunities. Dawson-Haggerty et al. [14] developed a stratified sampling method for surveying energy use and conducted a year-long, 455-meter deployment of wireless plug-load electric meters in a large commercial building. They found that the interior of a commercial building is a dynamic environment and confirmed the value of point-to-point routing in a real sensor network deployment. ViridiScope [37] is an indirect power monitoring system. It estimates an appliance's power consumption by placing a magnetic sensor near a power cord, based on the fact that the appliance emits measureable magnetic signals when it consumes energy. The eMeter system [46] provides device level energy consumption in a household that is based on a single sensor. This system can provide real time energy usage to a user's smart phone, which makes it possible to get consumers to save energy. Hnat, et al. [31] studied a large-scale residential sensing system for monitoring people's energy consumption in their homes. In this project, the team experienced significant connectivity and access challenges in the home-environment. In this measurement study, we collect and analyze power readings from an academic computer lab environment and put the results in the context of user behavior.

2.1.3 Large-Scale Energy Instrumentation

Several previous studies have addressed the energy consumption problem in computing. They may be categorized into two groups, i.e., i) measurements and analysis on a large scale of energy costs in computer systems, ii) green algorithms that might contribute to the design of future computer systems.

Yamini et al. [65] studied power management in cloud computing, and they proposed to use a green algorithm to enable lower energy usage in private and public clouds. Many researches have been focused on computers and data centers, while a few others have further considered all computing related resources. Wang et al., [61] studied the energy consumption, and they carried out measurements across five major cities. Their analysis showed that during the four years of experimental time, the IT-related electricity consumption was a major challenge in environment protection.

Studies on power consumption of computers may help us to optimize the design of computer systems and architectures. Krithika et al. [39] studied the power consumption of processors, including Intel and AMD xenon processors. Their findings are helpful for computer processor designers. Gong et al., [24] proposed fuzzy comprehensive evaluation criteria for future cloud system design.

Barroso et al. [5] studied CPU utilization of more than 5,000 servers during a six-month period. Their results showed those servers mostly operated between 10% and 50% of their maximum utilization levels, which follows the distributed systems design principles. Energy-proportional computers would significantly increase the energy efficiency since computers spend most of their time at moderate utilization

range.

However, very few studies have addressed the detection of computer failures and their relation to energy waste. Computer failures may cause unexpected damage in real-time computing systems. For instance, in a nuclear plant, modern command, control and communication system consisted of hundreds of computers. The risk of computer failure in such systems is very high. Alemzadeh et al. [4] revealed the high risk caused by computer failures in medical devices and their impacts on patients. Chen et al. [9] studied software service failure from system input and internal measurements. If we can predict computer failure both hardware and software, we may reduce the risk and improve computing system reliability.

2.2 Energy Study on Networking Systems

In this section, we will give an overview of research related to energy measurement, profiling, and their applications on wireless sensor networks.

2.2.1 Device-Level Energy Measurements

Hardware instrumentation: Energy consumption is a significant concern in the design and development of WSN; hence much progress and various measurement methods have been designed to measure the energy used by the nodes. In earlier days, researchers used an oscilloscope to measure energy consumed by the motes [7].

FlockLab [44] has power meters attached to motes so the researchers could understand the energy footprint of their protocols and applications. Epic [21] motes are probably the most well-known example of such nodes. Epic is an open mote platform for application-driven design. It integrated an iCount-based energy metering system which is based on a linear model on the time each unit was on. LEAP2 [55] provides unprecedented capabilities for directly observing energy usage for wireless sensor nodes in real-time with microsecond-scale time resolution that enables power profiling for each hardware subsystem. Researchers demonstrated node states can be accurately inferred from power traces [63]. In addition to inferring the working states on individual nodes, in this work, we are able to infer additional information, such as the actual routing topology.

Software instrumentation: people use number of transmissions as proxy for energy. There are other proxies people have used. PowerTOSSIM [54] employs a novel code-transformation technique to estimate the number of CPU cycles executed by each node, to estimate the power consumption for each node. Researchers proposed a software based on-line code-level energy estimation model in which the mechanism uses the current draw of each component during different periods and aggregates them together to produce the total energy consumption [18]. PowerBooter [67] is an automated power model construction technique to monitor power consumption on smart phone by built-in battery voltage sensors. Our techniques can use data from hardware or software instrumentation.

2.2.2 Applications of Energy Measurement

While the primary reason for energy measurement is to understand the energy used by a sensor network system, researchers have found other use for energy data. Power Trace Testing is presented in [64], which designed a method to automatically investigate the correctness of a WSN system by utilizing non-intrusive power measurement. In the testcase the system was able to detect an unexpected use of hardware component, which is not as scheduled. Dunkels et al. [17] used power-state tracking to estimate the wireless network power consumption on the network-level. Their approach can even break down the power consumption into individual activities on each node which enables the power profiling of the pre-activity energy cost. The design of our work, by using the feature name HPLC, is able to detect the unexpected application layer parameter changes such as transmission interval or packet size only from the energy instrumentation.

2.2.3 Revealing Privacy and Security Information

Researchers proposed a technique that used link-layer header data to infer network topology, de-anonymize servers present in an anonymized network, to break their anonymization[50]. Researchers demonstrated that even without priori-knowledge of household activities, it is still possible to extract complex usage patterns and privacy information from the household smart meter [47]. In this work, we are also able to reveal several unexpected pieces of information, such as instant routing topology and location of sink node. All of the information revealed in the work is only based on the

energy data, unlike approaches which access the routing information from decoding the packet headers.

2.2.4 Energy Consumption Evaluation of Protocols

There is also a large field of work on study the energy efficiency on the network protocols. Kumar et al. [40] studied the power consumption of four routing protocols through simulations. The simulation results showed the Flooding routing protocol was the worst in terms of power efficiency due to its relatively simple design while the nodes running protocol LEACH used the lowest average energy compared to the other protocols.

Siva and Daniel [49] proposed a centralized routing protocol called BCDCP. They compared the average energy dissipation values of this protocol to those of other clustering-based protocols by simulations. In their simulation they assumed the energy for transmitting and receiving a k-bit data message is determined by the distance between two nodes, and they applied the same parameters of energy consumption rate of radio operations used in [30]. The evaluation parts show BCDCP outperformed the other protocols in terms of system lifetime and energy consumption reduction on each node.

While some papers focused on measuring the power consumption on different nodes, others tried to summarize the relationship between power consumption and performance of different routing protocols. To the best of the author's knowledge, there have been few reports in the literature to date about the possibility to retrieve

and understand the patterns from energy consumption data or radio activities on the wireless sensor network.

Chapter 3

Understanding Desktop Energy Footprint

Energy-efficient computing has emerged as a major area of research and engineering in the recent years. As a result, hardware as well as software has become more energy efficient over the years. This progress is partly made possible by careful study of energy consumption of various components within a computing system. Identification of energy hotspots in hardware and software components helps us focus our effort in the areas that are likely to maximize the impact on computer power draw. Partial of this chapter has been published on [26].

3.1 Motivation and Overview

In order to prepare the knowledge of design a comprehensive energy instrumentation system, and enhance data analysis skills, we conduct a measurement study of power use in an academic shared computing lab environment. In academic buildings, these shared computer labs contribute for a sizable fraction of total energy use. Two factors differentiate this setting from the settings profiled in prior studies. First, the computer labs in a university setting are largely homogeneous: the labs have one or a small number of desktop models. This homogeneity makes this environment more similar to data centers than a typical population of computers in an academic research building. Second, the computers are shared across a number of users. The computers in a shared lab are not personal computers used by a single user. During the course of a day tens of students might log in and use a given computer. Thus, understanding power use of a single computer requires accounting for different users and their different computing requirements.

Recent studies have shown that the vast majority of workstations and desktop computers remain idle most of the time, which the average CPU idleness at 97.9% in classroom, while the average unused memory is 42.1% [16]. Heap et al. [29] performed 15-minute periodic resource monitor studies on Windows and Unix servers. The study found that Windows servers are idle for approximately 95% of the day respectively, while Unix servers had an average of 85% CPU idleness. Another study showed the average idle time for desktop machines was up to 80% of the day [3].

While some papers focus on measuring the power consumption for different machines, and others tried to summarize the relationship between power consumption and performance, however, to the best of the authors' knowledge, there have been few reports in the literature to date about the relationship between user behavior and energy consumptions, on the same type of computers.

In this work, we design an instrumentation for the computer lab. The instrumentation consists of two sets of sensing systems. First, there are wireless power meters that continuously monitor power draw of each desktop and transmit to a server using wireless network. We use power meters [36] to build our wireless energy sensing system. Second, a small service installed on each desktop PC monitors major user events and logs this information in a database. Using data collected with this infrastructure, we can develop a detailed understanding of power draw and the user activities on the computers that drive power use on the computers.

We collected 280 million samples of power and user activity readings in this work. Analyzing this data, we found a considerable heterogeneity in power use despite identical hardware and software configurations on the computers. This difference in power use across the machines is the result of several factors which we quantify in this chapter. Different users present a different type of computing load on the computers. They might use computers for different lengths of time. Finally, although the computers are expected to be identical, errors or mis-configurations cause these computers to become different and hence might result in different energy use. Our results also show that the computers were only used for a small fraction of uptime, which means the majority of energy used in the computer lab was wasted.

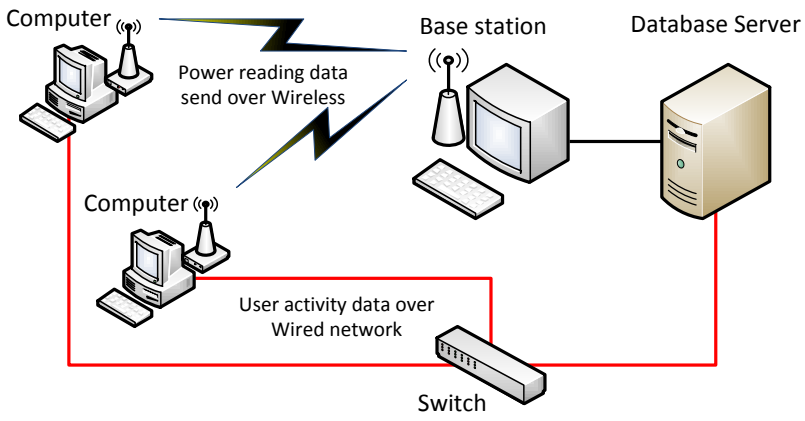


Figure 3.1: Architecture of computer energy and user activity instrumentation system

3.2 Instrumentation Design

We instrumented a computer lab to collect two sets of information: power draw of computers and user activity on the computers. The architecture of sensing system is shown in figure 3.1. The power readings and user activity logs were sent to a database server by wireless and wired network. Undergraduate and graduate students visit this lab for academic purpose, such as to finish programming homework or to access remote server by using ssh client. There are no scheduled classes in the lab. There are 22 desktops made by Gateway located in 4 rows, equipped with Intel Core2 duo 1.88G CPU and 2048 MB memory, running Windows XP. These desktops have CPU and monitor in a single package. There is no policy to restrict students from using any specific computer, but students are required to login with their personal student ID before using it. And this log in, log out activity was automatically recorded in

database.

3.2.1 Wireless Energy Meter

We use PowerNet [36] nodes to measure the power draw of a computer. The computer power cable is plugged to a PowerNet node and the PowerNet node is plugged to an AC outlet. These meters have energy metering ICs and MSP430 microcontroller for sensor control and data processing. The meter can sample power draw at up to 14 KHz. The power meters also have a IEEE 802.15.4 radio chip CC2420 running at 2.4 GHz unlicensed spectrum. We use RPL [56] running on TinyOS [41] for collecting power measurements.

We programmed PowerNet nodes to sample current at 10 Hz. The nodes pack 20 readings into a single packet and send it to the base station. Each reading is 2 bytes. 20 readings and metadata results in a 58 byte application payload. Metadata include a local sequence number, time stamp, and node ID. We increment the local sequence number after sending each packet. We used a local timer value as time stamp. Although this time is not globally synchronized, it is sufficient to study the time gap between the packets.

3.2.2 Process Monitor

We wrote a C++ application to measure CPU usage and installed as a Windows Service on all computers in the lab. Every second, the process monitor calculates each processs CPU usage by using Windows Management Instrumentation (WMI)

API [1]. The process monitor then transmits the list of processes and their CPU utilization to a database server over wired Ethernet. We installed the process monitor as a windows service so that it starts automatically during bootup and continuously collects information regarding processes running in the computer even when no one is logged in to the computer. The process monitor itself uses in average 0.45% (with a peak of 0.8%) of CPU resource, which we can safely ignore from our calculation.

3.2.3 User Authentication Monitor

We use a C# application to monitor user authentication. It records user log-in and log-out activities, user ID, machine name, and then saves this information on a database server.

3.2.4 Meter Calibration

We calibrated all the power meters before deployment. We performed six point calibration with resistive loads from 40 to 260 watts. This range is within the power draw range of a desktop, typically between 80 and 180 watts. The ground truth in Watts of resistive loads was calculated by using its instantaneous current multiplied by the potential difference across this component, which was measured by a high-precision multi-meter.

We first connect all of the resistive loads through one power meter. We then turn on the first load, wait for a few seconds until the temperature is stabilized, then take 50 raw readings using the meter and calculate the average. Then, we repeat the same

process by turning the load one by one. After we have 6 average raw readings at 6 different loads, we use Polynomial curve fitting function in Matlab to calculate the coefficients of raw readings of degree 1 that fits the ground truth. After calibration, all of the 24 lines overlap as expected because they are all measuring the current through the same resistive loads. After calibration, the Mean Square Error (MSE) in power readings across all of the meters was less than 0.2.

3.2.5 Data collection rate

Initially, the process monitor reported all the processes and their utilization to the database server. After analyzing utilization data for two weeks, we found that 85% of the readings had a utilization of 0%. We optimized network transmission and storage for these processes using a simple compression, which is tag a list of processes with a single utilization number of 0% rather than (process, utilization) tuple for each process. This optimization reduced the number of rows in the database by around 90%.

3.3 Evaluation

Our analysis of data collected in this study reveals up to 13.42 times difference in energy footprints of different computers in the lab. Furthermore, we found up even though computers have identical manufacturer specifications, their power draws can be significantly different. In this section, we elaborate on these findings.

3.3.1 Power across time

We first study the temporal trends in power draw of the computers in the lab. We found that the power draw can change by as much as 197.31% over the course of a day. When idle, most computers required around 70 watts. The maximum power requirement was 138.12 watts. The power changes depending on the load on the computer was due to user activities. Figure 3.2 shows the power draw of each machine for 12 days. Different machines show different temporal patterns. In this figure, day 6 and day 7 are weekends. No student was allowed to use the computers in the lab during that period, which is why the power was stable. Figure 3.3 plots the distribution of power draw across time for each computer. It shows that all computers spend at least 29% of the time drawing power less than 70 watts.

Aggregating the power draw from all the computers, we find that the total power drawn by the computers changes across time as shown in figure 3.4. It shows a minimum of 400 watts and a maximum of 1580 watts as total power used by computers in the lab.

3.3.2 Power across activities

One of the reasons for different power draw across time is the changing user activities. The activity most relevant to understanding power draw is a student logging into the computer, launching applications, and after some period, logging off. In figure 3.5, we plot the CDF of average power for sessions during which users were logged in and were not logged in. The power is generally higher when the user is logged in compared

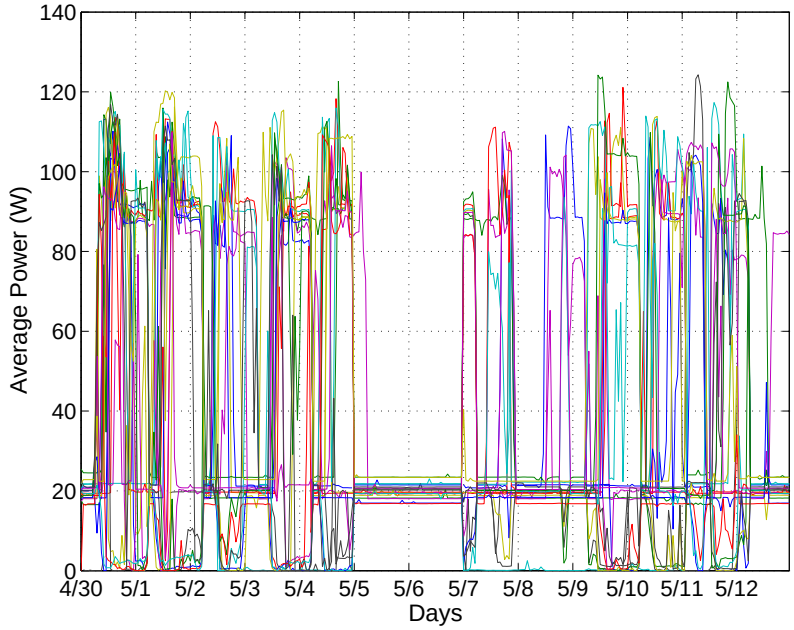


Figure 3.2: Average power draw of each computer across time

to when the user is not logged in, approximately 80-140 watts compared to 20-80 watts on most of the machines. The reason it consumed more energy when user was logged in, is not only that more CPU operations were performed, but also hard disk read-write operations, graphics card calculations, and network transmissions which expend more energy. Some of the blue lines stay around 20 watts for large fraction of time. That means those machines were left in power-off status. For a small fraction of the dataset, the computers use more power when a user is not logged in compared to when a user is logged in. This is because the Windows was performing system updates or scheduled virus scan. Our test indicates this model of machines consume approximately 15 - 22 watts power even when power is turned off; that was mainly because the internal capacities are still charging.

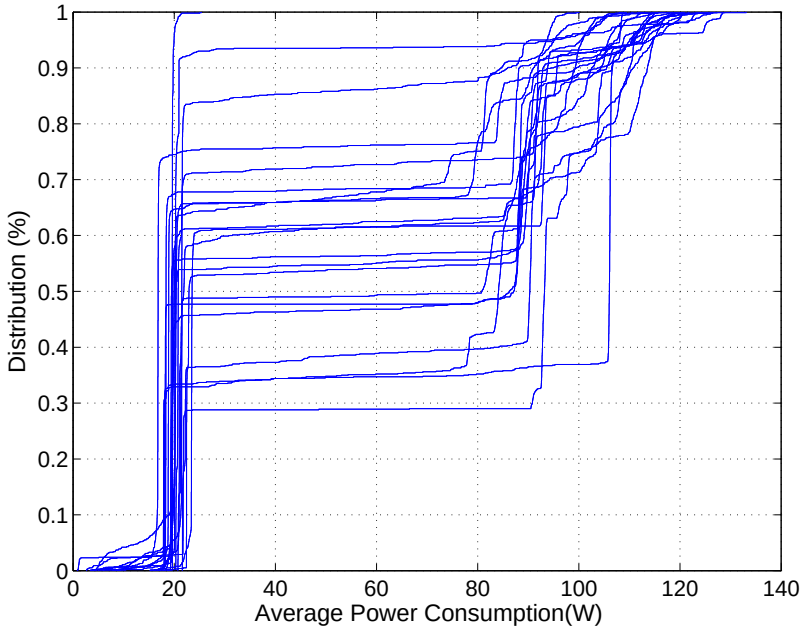


Figure 3.3: CDF of power draw of each computer

3.3.3 Power across machines

Although all the computers in the lab have identical manufacturer specification and software installation, each computer is slightly different due to manufacturing difference, hardware abuse and errors over time, and unintentional errors and updates on the software. For example, some machines do not have McAfee VirusScan installed, while others do. It is believed that such discrepancy is due to software configuration errors accumulating over time.

To understand the difference between the machines, we study two sets of trace. We first plot the distribution of power draw when no user is logged in and CPU utilization is 0-1% in figure 3.6. The figure shows even when all the computers are

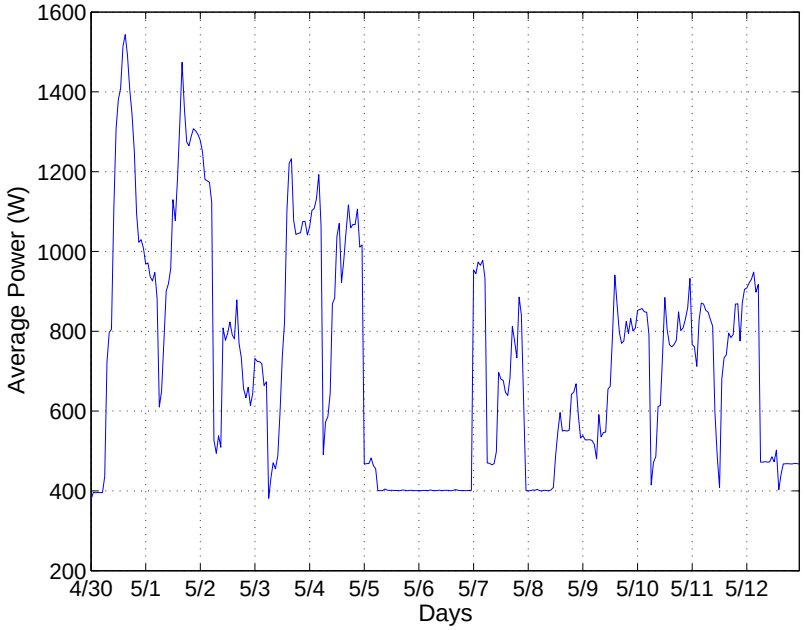


Figure 3.4: Total power used by all the computers

idle, the power draw across the machines is in 85-115 watts range. In the same figure, we also plot the power draw when a user is logged in and the CPU utilization is 99-100%. When the machines are fully utilized, the power draw across the machines are in the 108-136 watts range. Thus, our results show that even though computers of similar manufacturer specification and software installation are subjected to similar CPU loads, the power draw can be significantly different.

3.3.4 Power across students

In figure 3.7,we plot the total computing energy used by 30 students who logged in the most number of times during the two weeks. Each line represents the total energy

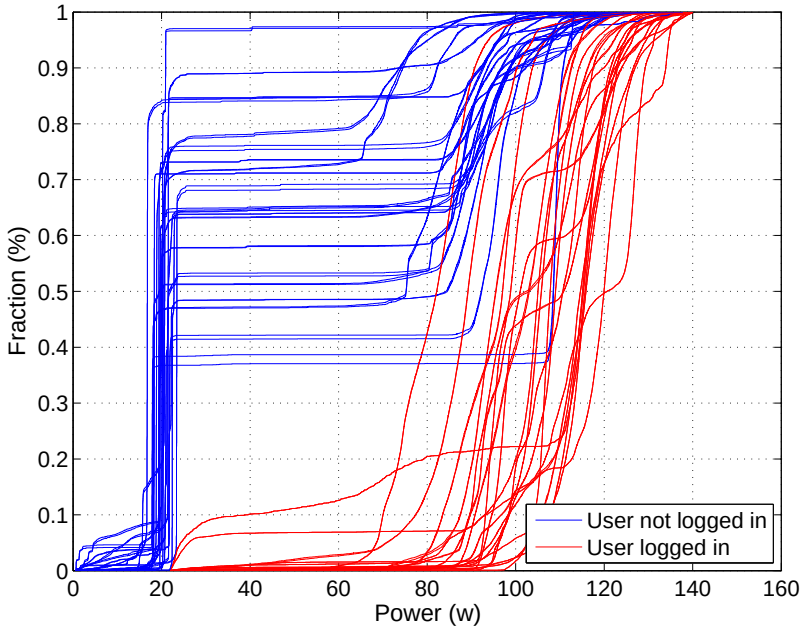


Figure 3.5: CDF of power draw of each computer when a user is logged in and not logged in

used by a student. The dots shows the value of energy consumed during that log-in period. All of these numbers were captured during the same period as the figures showed in pervious sections. We can see that around 80% of students used less than 1,000 kilojoules, while only 0.5% of students used more than 3,000 kilojoules, which is mostly caused by students having been logged in for a significantly longer period, compared to other students.

From figure 3.8 we can tell each student has her unique average power usage, but 70% of these values are in the range of 100-120 watts. In some cases the average power is below 100 watts mainly because after the user logged in, the user left the computer idle or performed some simple tasks like writing emails. The user activity

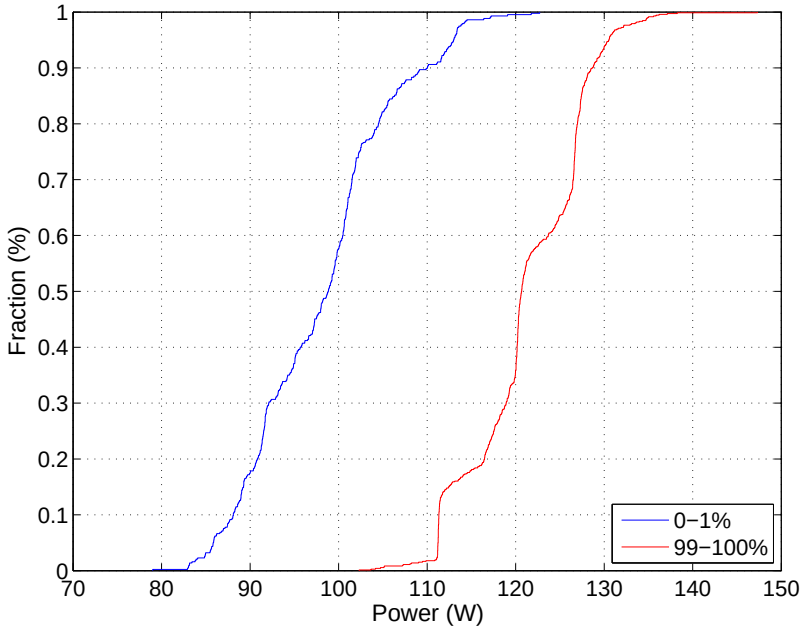


Figure 3.6: CDF of power draw when CPU utilization is 0-1% and 99-100%

log indicates that no high workload task was performed on that computer during that period.

3.4 Discussion

Instrumenting all the computers in a lab is a capital and time intensive process. It is natural to ask if we can use CPU utilization as a proxy for power use. To answer this question, we try to interpret the power draw shown in Figure 3.9 in context of corresponding CPU utilization distribution. Each line in the figure represents individual machine. It is not surprising to see that the fraction of CPU utilization while user is not logged in stays below 30%-40% on most machines, which was caused

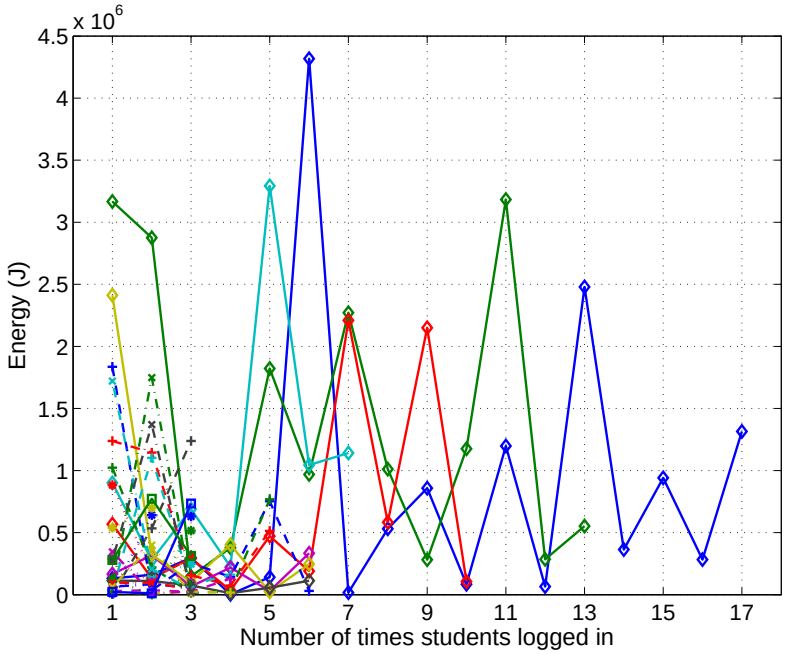


Figure 3.7: Energy consumed by each student

by some background processes like real-time back up and viruses scans. On the other hand, when a user is logged in, the CPU utilization is distributed in the range of 40% - 90%. While users are using the machine, they will consume more CPU resource, since every user task requires additional CPU operations.

Considering these results, using CPU utilization as a proxy for power draw seems promising. However, using CPU utilization to estimate power draw at a finer granularity is not as promising.

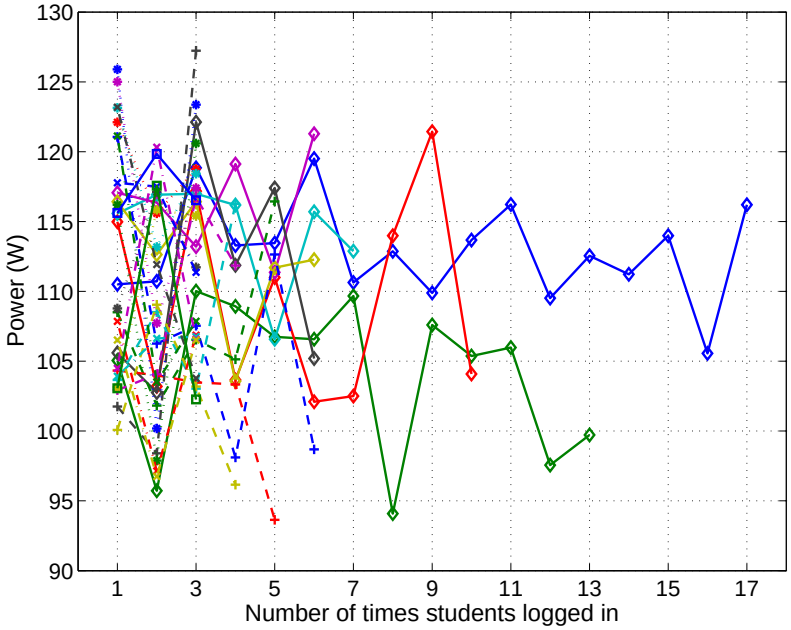


Figure 3.8: Average power used by each student

3.5 Summary

In this work, we described the sensing system that instruments the energy consumption and user activity in an academic computer lab. We measured the power consumption, CPU utilization, and user activity across a homogeneous set of desktop computers. We studied the factors that drive the heterogeneity in energy use across the desktops and lessons learn from the real-world measurement on 22 machines over one month. We found that the energy consumption of each computer is highly related to individual user behavior, and the 60% of energy consumed every day was during the computer was on and no one was logged in.

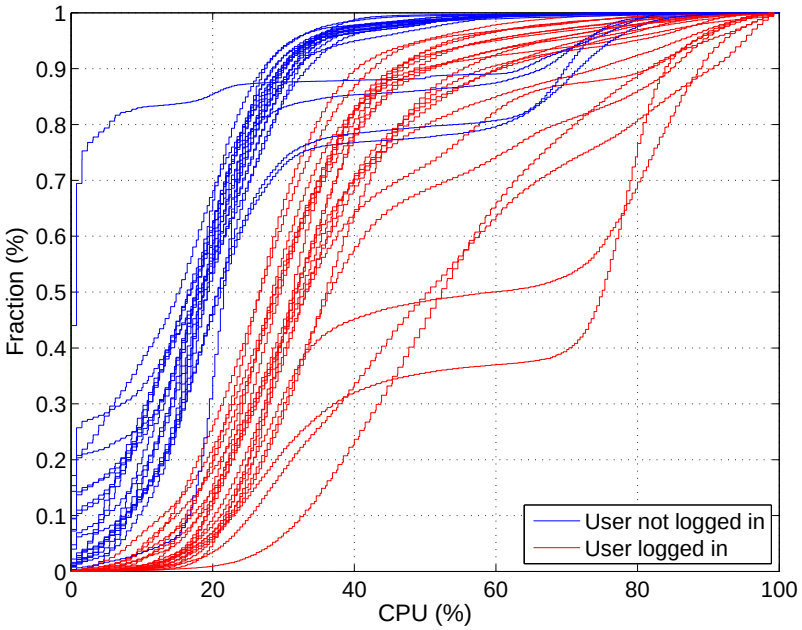


Figure 3.9: CDF of CPU utilization on all computers

Chapter 4

Understanding Computing Systems from Energy Instrumentation

Nowadays, computers are one of the most critical devices used in almost every field. For those fields that heavily rely on computer systems, such as finance, communication, public service and military, the cost of system failure is tremendously large. The prediction of such failures is not straightforward. In this chapter, we propose novel techniques for computer failure detection from energy consumption data. The power measurements data may provide us very useful information in some failure scenarios. However, there are two limits of what we can detect with the power instrumentation. The instrumented values are the aggregated power consumption on the whole device. The detected value cannot reflect the changes in drawing power

by individual component of the device. At current stage of this work, our proposed system can only detect the interference-free events, i.e., during given period, only one event is happening on the computing system, there are no other applications or events actively running at the same time. In our experiments, the power draw shows distinguishable patterns when the computer launch different applications at a time, even use same application to open different files. However, at current stage, our proposed energy instrumentation is not able to distinguish multiple events concurrent running on same computer.

We first study the energy consumption on computer systems, including desktop computers and server computers. We collected 30GB of energy consumption data over 37 months. This data set contains the energy consumption measurements containing everyday use of computers in lab. The measurements of energy consumption show that the average energy consumption during boot up periods has increased, and the maximum power draw also increased.

We further studied the potential relationship between computer failure and its power consumption pattern. One common failure, the Blue Screen of Death is considered. Its power draw follows the same pattern, regardless of the specific reasons that caused the computer failure. Power consumption calculated from the presence of Blue Screen to reboot is constant.

The contributions of this work are three-fold, i) construct a large data-set for studying computer power consumption and its relation with common computer failures, ii) extraction and analysis of power features that correlated to boot up, idle and failure, iii) a novel method to model and detect blue screen failure and Pre-OS

boot up from energy consumption data.

4.1 Definitions

In this section, we define the terms computer boot up, computer idle, and computer failure that we used in this work. The reason to study these three events is because these event can reflect the power consumption properties of the computer without the interference from the operation system and the user activities.

4.1.1 Computer Boot Up

In this chapter, we try to answer this specific question: Does the same computer consume the same amount of energy for boot up over a long period of time? To answer this question, we introduce the definition of the boot up process on a computer, then define a term called Pre-OS boot up that we use in section 4.3.2.

Computer system booting is the process of loading an Operation System (OS) into a computer's main memory, in order to prepare the computer for a user to use. The basic system boot process on a computer system given in [19] is comprised of the following major steps:

1. Turn on the computer's power button.
2. The computer initializes the CPU and reset registers.
3. The CPU loads BIOS (Basic Input/Output System) which holds its startup

instructions.

4. The BIOS runs POST (Power-On Self Test) and other necessary hardware checks.
5. CPU initialization is completed if all tests are passed.
6. BIOS finds the storage drive where the OS is installed.
7. The BIOS copies the OS files into memory and executes them.
8. The OS takes over the control of the boot process.
9. The OS checks hardware inventory and loads hardware drivers.
10. The OS is ready for the user to use.

Because the manufacture of OS publishes security updates and even distribution upgrades over a long period of time, which potentially will cause the time length to copy OS files, the OS execution time changes. In this study we focus on the boot up process until step 6, in order to remove the influence caused by the OS changes. We define the *Pre-OS boot up* on a computer as the process that starts when the computer's power button was pressed, until the moment when BIOS is ready to load OS files. This process is stable over a long-term period unless the BIOS was updated. We study the Pre-OS boot up changes during a 18-month period, and the results are presented in section 4.3.2.

4.1.2 Idle

Periodic verification that the computer is in an idle state is a common setting on various Operating Systems. If the computer is verified as idle, the OS turns off the monitor and puts the computer into sleep state as one of the default operations to save energy.

The Task Scheduler in Microsoft Windows verifies that the computer is in idle state every 15 minutes, by using two criteria to check the idle state: user absence and lack of resource consumption[8]. The user absence is defined as there being no mouse and keyboard input during a given period of time. The Windows expects an active user to interact with the input devices every few minutes, so if there is no input longer than the threshold value, Windows assumes the user is absent. Furthermore, if all of the processors and the disks in the computer were idle for more than 90% of time during last detection interval, Windows will consider this computer is idle.

In our work, we define the idle state in the controlled environment as the period during which there is no user interaction, and no background third-party application is running. On the enterprise computing dataset, due to lack of the ground truth of the computer and user activities information, we consider the computer is in idle state based on its power consumption value. If the power is less than the maximum power data, the standard deviation of the power data during the given period is less than 10W, and only when this period is longer than 30 minutes, we assume this computer is in idle state.

4.1.3 Computer Failure

Even though the modern computers are designed with complex hardware and advanced software, it is still hard to prevent computer failure from happening at all. A Kernel Panic is one of the actions taken by the Unix OS upon detecting a critical failure, and usually the OS cannot safely recover from this type of failure. The Kernel Panic could happen during the booting up period, to prevent the system from being properly loaded. The following list gives the common reasons of Kernel Panic: bug in kernel, missing or corrupted disk volumes, failing hardware driver, unsupported hardware.

After a system failure on a computer installed with Microsoft Windows, the Blue Screen of Death (BSoD) shows the error code and its symbolic name. Although this error information was designed to help software engineers to locate the problem, finding the actual reason that caused the BSoD is a time-consuming job. By default, Windows creates a memory dump file to the local hard disk when the failure happens. After that, Windows will restart the computer. Overheating of components, problems of power supply, memory fault, stack overflow, and hardware driver bugs are the common reasons to cause BSoD on a Windows machine. We trigger BSoD in a controlled lab environment by 8 different reasons and study their power consumption patterns in section 4.2.5.

4.2 Evaluations in a Laboratory Environment

In this section, we study the energy consumption patterns of one desktop computer in the lab environment, so that we are able to generate various events and instrument the power consumption during these events, such as computer boot up, idle and failure. Meanwhile, performing all of the experiments in a controlled lab environment can help us to evaluate our proposed algorithms performance before we apply them on the dataset without ground truth information.

4.2.1 Instrumentation Design

The instrumentation settings include a computer installed with Microsoft Windows 7, a hall-effect current sensor attached to the computer's power cord to measure the current draw on the computer, a Data Acquisition (DAQ) device connected to the hall-effect sensor to record the sensing values from the hall-effect sensor, and a laptop that monitors and records the DAQ's readings. In order to reduce the interference caused by background running services and applications, we re-installed a fresh copy of Windows. We test the current sampling rate from 100Hz to 800Hz with a step size of 100. The results show 100Hz sampling rate is sufficient to capture the current changes, so in our experiments we select 100Hz to measure the current draw.

4.2.2 Design of Algorithm for Event Detection

In order to automatically detect the various events on computing system only from the power draw, we designed an algorithm and evaluated its performance in this section, before we applied it on the large enterprise dataset. The main idea of the algorithm is first to apply Gaussian filter to remove the outliers caused by the meter reading error from the raw data, then to perform Sobel operator on the smoothed data to calculate the gradient value. The gradient value indicates the change step of power consumption readings. Finally, based on the patterns of gradient values, the algorithm identified each of the events from the power draw. The detailed design of the algorithm to detect boot, idle and failure are described in the following three sub-sections.

4.2.3 Boot

We explain and evaluate the algorithm to detect the boot up and Pre-OS boot events here. The top part of figure 4.1 shows power draw measured by the DAQ device during the first 50s starting from when the user pressed the power button on the computer. Near the time 3s when the power button was pressed, the power consumption value rapidly increased from 6W to 100W in less than 1.5 second, which is caused by most of the components in the computer powering up. The bottom part of figure 4.1 shows the corresponding gradient value calculated by the Sobel Operator. The gradient values are all 0 before the moment when the power button was pressed, since there were not changes during that period. Then the gradient

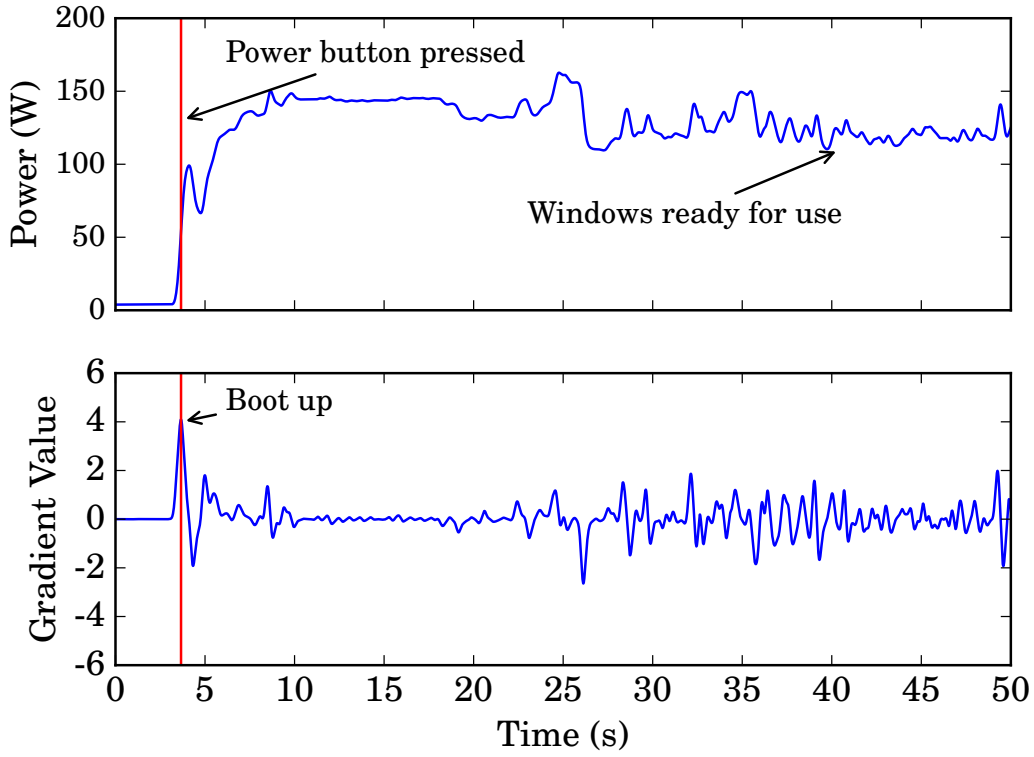


Figure 4.1: Boot up event detected by using power draw

value sharply increased to 4 at the same moment when the power consumption value increased from 6W to 100W. After the computer boot up, although the power draw shows fluctuations, there is no power change sharper than the boot up moment, so the gradient value never exceeded 4 after boot up. Based on this observation, we designed the algorithm to detect a boot up event in algorithm 1.

The red vertical lines in the figure 4.1 mark the boot up time detected by this algorithm. We evaluated this algorithm on two different computers. Among 10 power draw datasets which contain the boot up event, this algorithm achieved to 98% accuracy to detect the boot up events. When we apply this algorithm to computer with

Algorithm 1 Detect Boot

```
1: procedure BOOT(P, n, r)   ▷ P: power consumption array, n: size of array P, r: power sampling rate
2:     Initialize B
3:     G ← Sobel(P)                          ▷ Use Sobel operator to get gradient
4:     for i ← R, n do
5:         if G(i) ≥ 3 then
6:             if P(i − R) ≤ 15 then
7:                 B ← i                     ▷ Add timestamp i to boot up array B
8:             end if
9:         end if
10:    end for
11:    return B                              ▷ The array of Boot up timestamp
12: end procedure
```

different hardware configurations, the threshold value of 3 might need adjustment according to the power consumption on that particular machine. When we have enough power consumption data, we can set this threshold value to the maximum gradient value minus 1.

Next, we describe the algorithm to detect Pre-OS boot. As described in section 4.1.1, a computer boot up includes 10 major steps. The booting process starts with the CPU reset, then hardware self checks, until BIOS locates the storage driver where the OS is installed, where the Pre-OS boot ends. After that, BIOS copies OS files into memory, and launches OS when it is ready. We have carefully tested when the OS starts to copy OS files into memory. Every time when the computer boots up, there is a short period of black screen on the monitor, between when the computer finishes its self tests and the OS logo appears on the screen. To verify that this is the moment when Pre-OS boot up ends, we have plugged a newly formatted hard drive without any OS installed on it. The black screen still appears after the

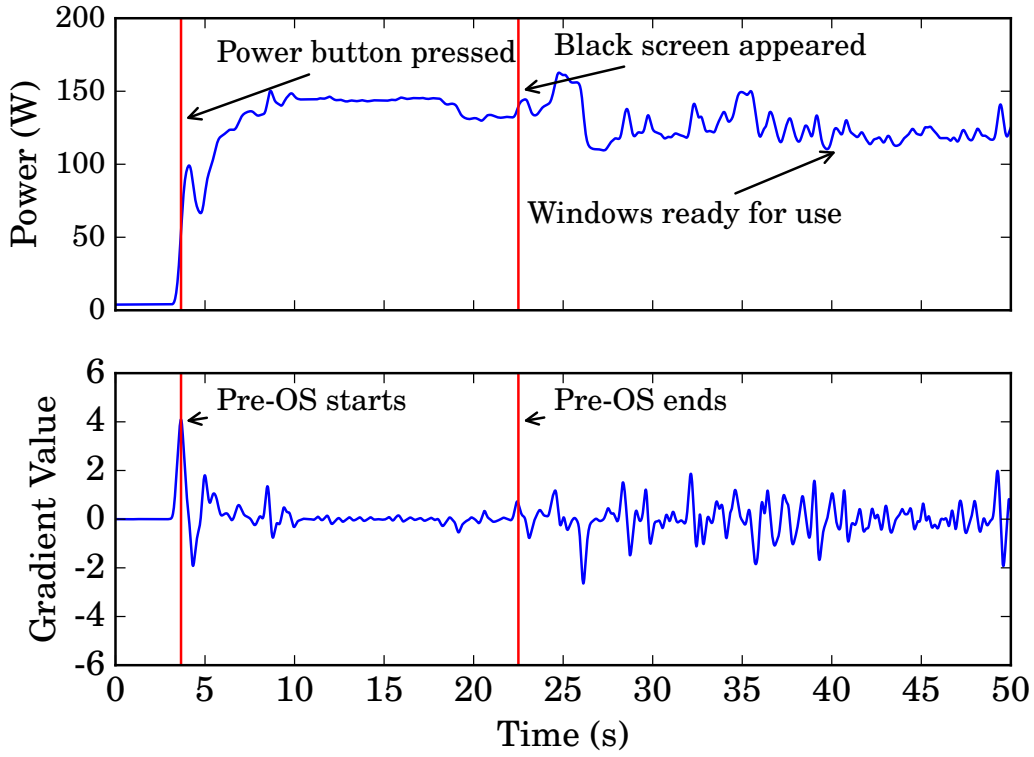


Figure 4.2: Pre-OS boot up period detected by using power draw

computer's self test, then immediately shows on the monitor with the error message "No Bootable Devices Error". With this experiment, we confirmed that after the black screen appeared on the monitor, the BIOS started to looking for the OS files. The main purpose here is to locate a consistent boot up period which is not affected by the OS files. Although we cannot precisely confirm when the BIOS starts to read for the OS, what we can confirm by this experiment is that our algorithm can detect the black screen event, that is consistent across boot up events and not affected by the OS changes.

While the DAQ device recorded the power draw during the computer boot up,

we also recorded the time when the black screen appeared on the monitor, which is annotated in the top part of figure 4.2. A close look at the power draw curve shows a 10W increase. Accordingly, there is a 0.5 rise in the gradient curve, which is shown in the bottom part of figure 4.2.In the algorithm, after we detect the first sharp gradient, in the range of 10s to 20s, we look for the gradient larger than 0.4 for the Pre-OS boot event. We also noted that starting from 10s after the computer boots up, the gradient values were consistent and were always less than 0.5. We used algorithm 2 to detect when the Pre-OS boot ends:

Algorithm 2 Detect Pre-OS Boot

```
1: procedure PreOS-Boot(P, n, r)
2:     InitializeB
3:     G ← Sobel(P)                          ▷ Use Sobel operator to get gradient
4:     for i ← R, n do
5:         if G(i) ≥ 3 then
6:             if P(i − R) ≤ 15 then
7:                 for j ← i + 10r, i + 20r do
8:                     if G(j) ≥ 0.4 then
9:                         B ← i          ▷ Add timestamp i to Pre-OS boot up array B
10:                    end if
11:                end for
12:            end if
13:        end if
14:    end for
15:    return B                                  ▷ The Pre-OS Boot timestamp
16: end procedure
```

Applying this algorithm to the dataset used in figure 4.2, the second red vertical line marks the moment detected by this algorithm. So that the period between the two red vertical lines were the Pre-OS period. To test the stability of this algorithm, we applied it on ten power boot up power consumption datasets. The results showed

that this algorithm could detect the moment of the 10W power increasing. This output may not precisely detect the moment when the OS starts loading the OS files, but the output consistency the locates the same 10W power increase pattern. Consistency is our top priority, since the Pre-OS Boot Up defines a consistent computer boot up period, which avoids the OS's variations.

4.2.4 Idle

Computer idle has been identified as one of the major sources of energy waste in a computer lab environment [27]. In this part, we show the energy instrumentation that could help users to identify computers' extra activities during idle state. We measured the energy draw on an idle computer continuously for 10 hours, to find out how the computer consumed energy during its idle period. Our observation was that the power consumption values on the target computer kept consistent in the range of 104W to 107W, except the power increased to 130W to 140W every 5 minutes, as annotated in top part of figure 4.3. This 30% instant increase in power only lasted less than 200 milliseconds, every time. We checked the Windows default background service settings, and found out that the group policy refresh interval was 5 minutes, which matched the instant current increase interval. In order to verify that this group policy refresh action generated extra energy consumption, we changed this interval value in the computer configuration interface to 30 minutes, and repeated the current instrumentation after rebooting. The measurement results showed that the instant current increase interval changed to 30 minutes, which confirmed that the group policy refresh runs in the background and generates extra energy consumption.

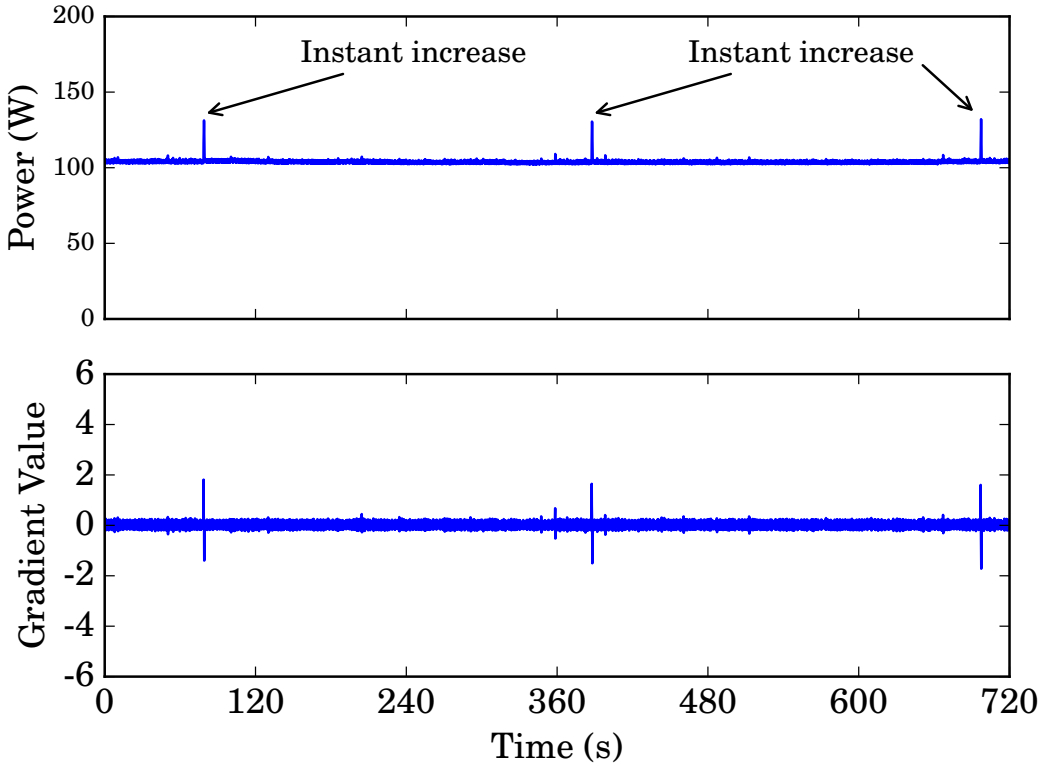


Figure 4.3: Idle detected by using power draw

As observed in figure 4.3, both of the power draw and gradient values are consistent, except the instant increase every 5 minutes. Windows considered the computer as idle when its 90% resource are idle, so we proposed the algorithm 3 to use power draw to detect the computer idle period.

4.2.5 Failure

In this part, we study how the power draw changes when critical failure occurs on the computer. We generated the BSoD on the Windows computer by using eight types

Algorithm 3 Detect Idle

```
1: procedure IDLE(P, Pm) ▷ P: sub-array of power consumption, Pm: Maximum of power consumption on this computer
2:     G ← Sobel(P)                          ▷ Use Sobel operator to get gradient
3:     X ← Std(P)                            ▷ Standard deviation of P
4:     Y ← Std(G)                            ▷ Standard deviation of G
5:     Z ← Ave(P)                            ▷ Average of P
6:     if X ≤ 10 then
7:         if Y ≤ 0.4 then
8:             if Z ≥ 10 then
9:                 if Z ≤ 0.6 * Pm then
10:                    return True           ▷ This period is Idle
11:                end if
12:            end if
13:        end if
14:    end if
15:    return False                          ▷ This period is not Idle
16: end procedure
```

of reasons, which included: high Interrupt Request Levels (IRQL) fault (system-mode), high IRQL fault (user-mode), buffer overflow, code overwrite, stack trash, stack overflow, hard coded breakpoint, and double free. We focused on the power consumption data during the period starting from the moment when the BSoD was triggered, until the moment when the computer restarted itself, which is shown in figure 4.4 starting from 4s until 11s. As the top part shows, the power consumption rapidly increased 80% from 110W to 190W as soon as the BSoD started. During the BSoD period, all of the major components inside the computer, such as the CPU, hard drive, and fans were all running at their maximum speed, which caused the power draw to significantly increase. The instrumentation showed the power draw stayed at its max level until the BSoD screen disappeared on the monitor, then the power draw dropped to 115W. At the same time, the computer began the

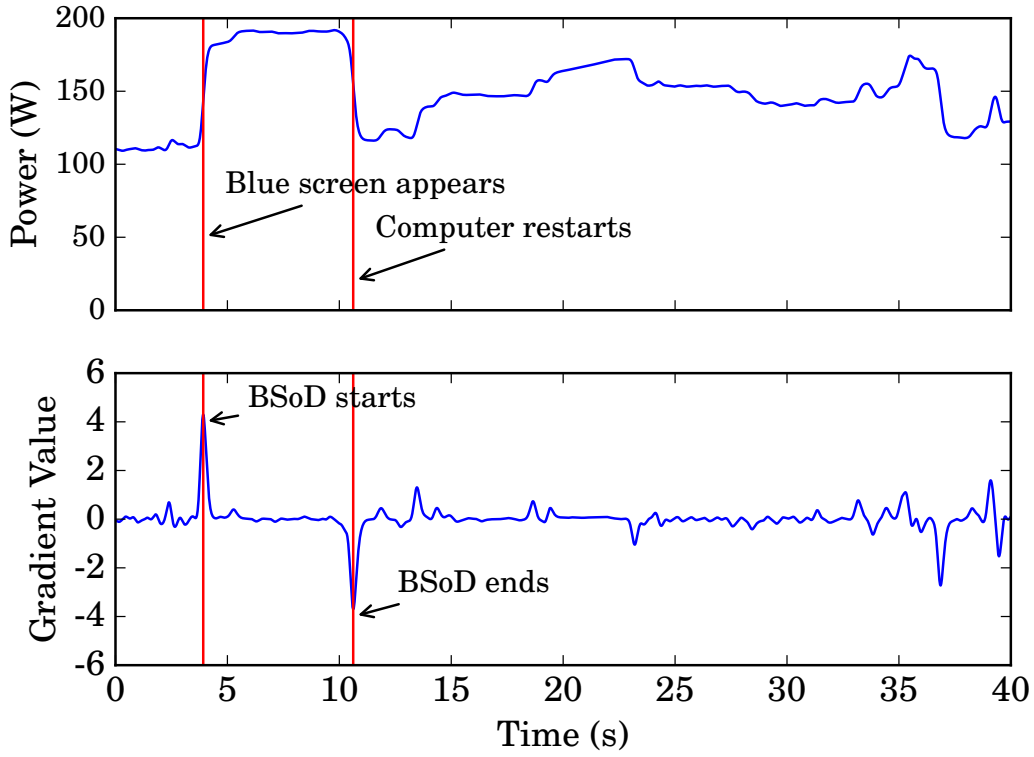


Figure 4.4: BSoD period detected by using power draw

restart process to recover itself from failure. The measurements were repeated 3 times for each reason on the same computer, generating a total of 24 sets of data. Our conclusion is that regardless of which reason triggered the BSoD, the power draw showed the same pattern: rapid increase to 190W and remained at that level for 7 to 8 seconds, then dropped to 115W. During the same period, the gradient value increased to 4 when the BSoD appeared on the screen, the gradient value immediately returned and stayed around 0, until the moment it dropped to -3, when the computer restarted. Totally we have measured 24 times of the BSoD triggered by 8 type of reasons, all of the length of the BSoD are last for 8s. So that below

algorithm we check the gradient in a range of 5s to 15s, to cover to period of the BSoD event. With these observations, we designed the algorithm 4 to detect BSoD failure on the computer.

Algorithm 4 Detect BSoD

```
1: procedure BSoD(P, n, r)
2:     G ← Sobel(P)                          ▷ Use Sobel operator to get gradient
3:     for i ← R, n do
4:         if G(i) ≥ 3 then
5:             for j ← i + 5 ∗ r, i + 15 ∗ r do
6:                 if G(j) ≤ −2 then
7:                     Gstd ← Std(G[i : j])      ▷ Standard deviation of during i to j
8:                     if Gstd ≤ 0.5 then
9:                         return True                          ▷ Detected BSoD
10:                    end if
11:                end if
12:            end for
13:        end if
14:    end for
15: end procedure
```

The period between two red vertical lines in 4.4 indicates the BSoD period detected by this algorithm. In order to test the performance of this algorithm in a more complex dataset, we evaluated it on a 5-minute experiment, which included boot up, short period of idle, and user interactions. The results of this experiment are shown in figure 4.5, we manually triggered the BSoD the around 13:58:50. The red vertical lines indicate the moment when the gradient value was more than 3. As shown in the figure, there are more than red lines marked, but due to the second resection in our algorithm, there is only one green vertical line which shows when the BSoD ended. That left only one period between the red and green line as the BSoD period, which exactly maps to the correct BSoD period.

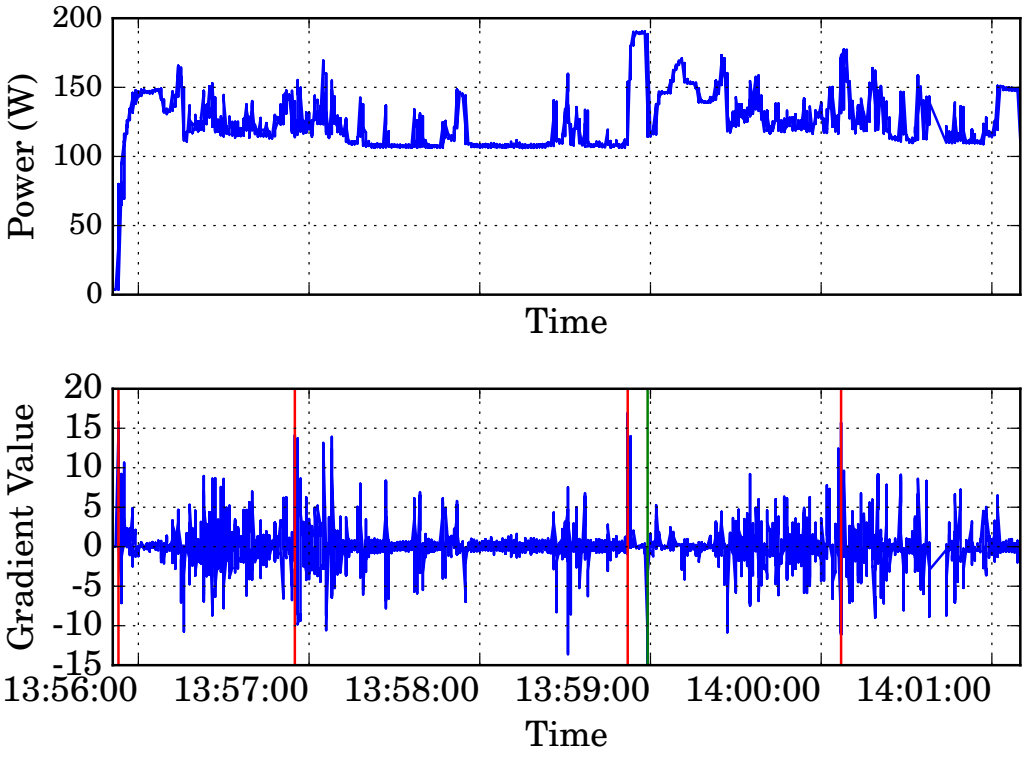


Figure 4.5: Use power draw to detect BSoD period in a longer scenario.

In this part, we designed four algorithms to detect boot, Pre-OS boot, idle, BSoD failure events only based on power consumption data, and also demonstrated that these algorithms could help us to identify that all of these four events have happened on the computer only by monitoring the power draw. Compared to installing monitor programs on the computer to detect failure and abnormal behavior, the method of monitoring the power consumption on the power cord neither requires having the privilege of installing programs on the computer, nor occupying the computer's computing resources.

4.3 Evaluations on Enterprise Computing Environment

In this section, we study the energy consumption from PowerNet [36], a 3-year continuous energy measurements of a large enterprise computing environment, and use our proposed algorithms in last part to detect various events on computer and to reveal the insignificant changes which only can be discovered over a long period of time.

4.3.1 Dataset Background

The PowerNet project from Stanford University was aimed at characterizing and understanding the energy consumption by computing devices and communication equipment. The network of power meters monitored 270 end-devices, including 54 desktops, 31 laptops, 26 servers, 30 printers and 19 network switches. The power cord on each device was plugged to a PowerNet meter, and the PowerNet meter was plugged to an AC outlet. These meters have energy metering ICs and MSP430 micro-controller for sensor control and data processing. The meter measured and recorded the power draw at 1 Hz. The raw data of PowerNet meter readings are more than 30GB, starting from May 2010 to June 2013, spanning 37 months.

4.3.2 Boot

PowerNet contained the energy usage data for more than 3 years, and this part studies the power consumption changes during the Pre-OS boot up period on one desktop computer across 18-month period. This is the only desktop that we can find in the dataset which has regular power on and power off events, the other machines are either always kept power on, or the available energy only covers a very short of time period. First we explain the step by step process about how to identify every Pre-OS boot up event from the continuous power monitor data. In figure 4.6, each dot shows the average power consumption during every 1 minute period, from May 1st, 2010 until December 31st, 2011. During this 18-month period, the power data is mainly distributed in two major ranges. The first range is recorded while the computer is powered off, and the readings are around 5W. This small amount of energy is consumed by the internal components inside the computer, such as the network interface card is kept in stand-by mode and waiting for the remote wake up command. The other major power readings are recorded while the computer is under active use, and the readings are distributed in 55W to 75W range. The energy draw changes depending on the workloads on the computer; a higher workload usually consumes more energy.

Figure 4.7 shows how the power consumption changed during 24 hours on a weekday. According to this figure, the computer was powered on at approximately 09:50am, and was shut down around 4pm. Because the power change values during the power up and the shut down moment are significantly larger than other times of change in power, we could use the maximum positive gradient value during the whole

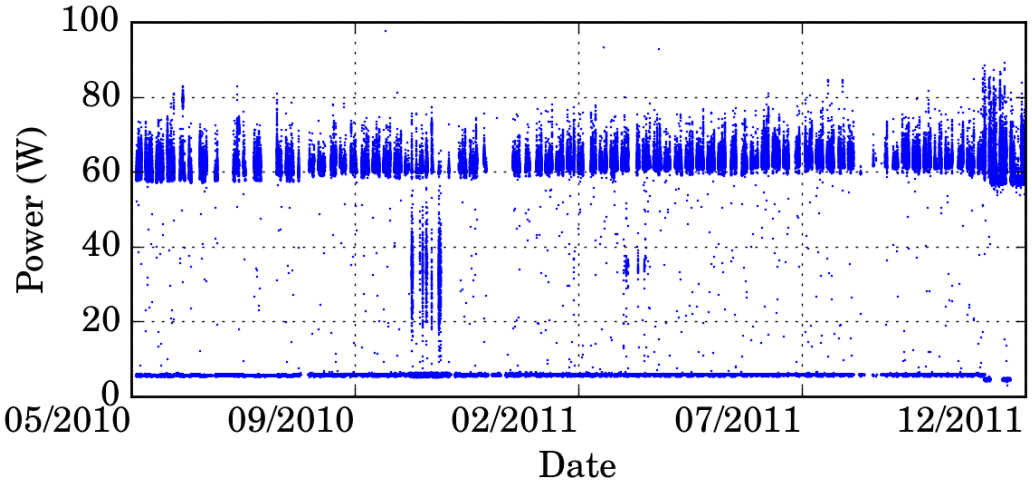


Figure 4.6: Power draw across an 18-month period on same computer

data period to identify the power up event, and use the maximum negative gradient value to identify the shut down event. The green vertical line and red vertical line in figure 4.7 mark the power up and shut down moments detected by this algorithm. A close look at the computer boots up event is shown in figure 4.8, which plots the power draw starting from 1 minute before until 5 minutes after the power button was pressed. Although the computer hardware components are different between the computer measured in PowerNet and the desktop computer that we instrumented in section 4.2.3, the general trend of the power draw during the computer booting are similar to each other. The power rapidly increase immediately after the power button was pressed, then drops 20% to 30% while the computer is loading BIOS.

Based on the definition of Pre-OS booting given in section 4.1.1, we extract the power consumption data during the Pre-OS boot up period and plot it in figure 4.9. Note the time scale we used in this figure only covers 40 seconds. In the rest of this

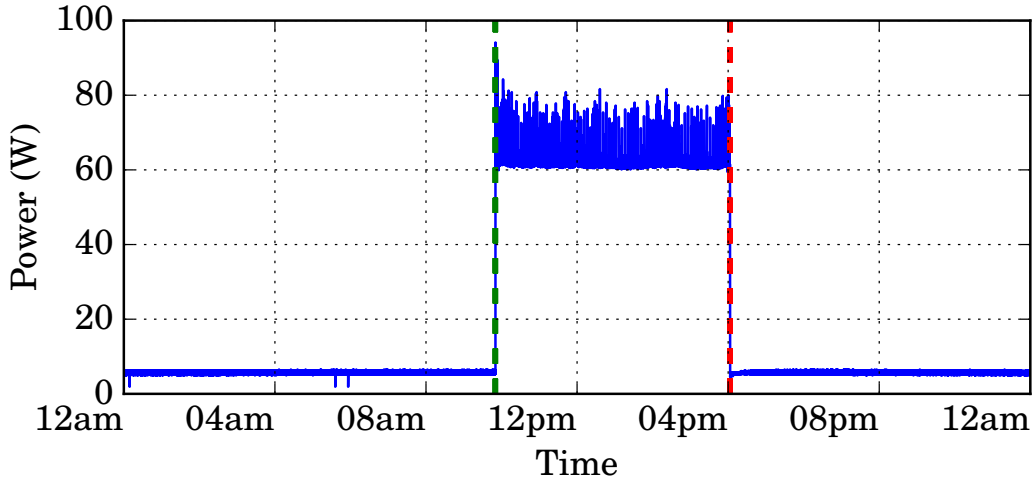


Figure 4.7: Power draw during one day period

part, we study the changes of the power consumption during the Pre-OS boot up period, which is instrumented on the same computer across an 18-month time frame.

Power Consumption during Pre-OS boot: We calculate the average power consumption by using the total power consumption during the Pre-OS boot up divides by its time length. As shown in figure 4.10, the blue dots show the raw average values, the green curve shows moving average value in a sliding window size equals to 10. This green color moving average curve suggests the average power consumption has a rising trend during the 18-month period, which lead us to apply the linear regression and plot it as the red straight line in the figure. The linear regression result reveals this Pre-OS boot up event on this computer consumes 5.9% more energy. While the age of the computer rises, the energy efficiency of its components drops. For example, accumulation of dust reduces the efficiency of the fan, which increases its energy consumption to maintain the air flow. The reason of the linear regression

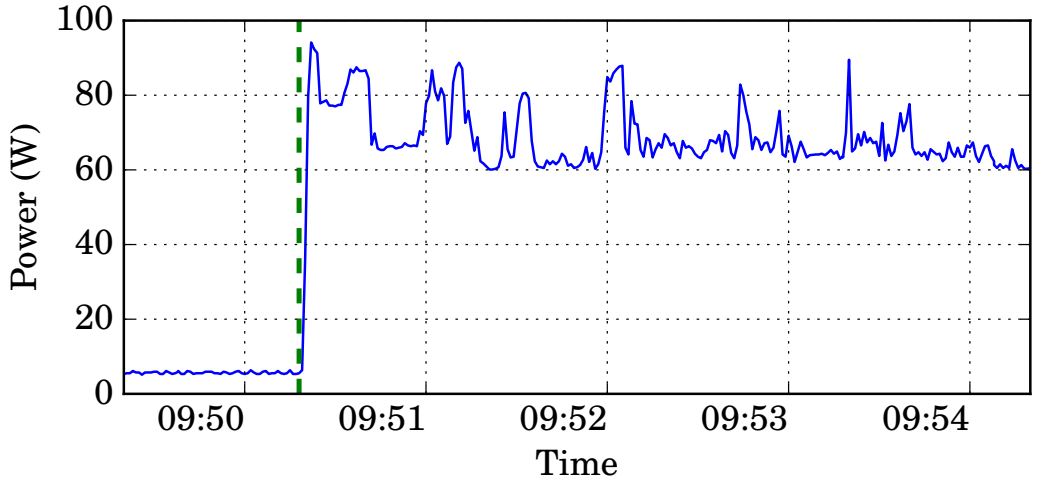


Figure 4.8: Power draw during the boot up period

curve is not a straight line, is because there were some gaps in the raw dateset, we plot the linear regression curve skipped those gaps.

The maximum power consumption value during Pre-OS boot up is another interesting metric to reflect the energy proprieties of the computer. If this value shows a significant change, it could reveal some components are not working properly. We plot the maximum readings during the same time period in figure 4.11. Similar to the trend observed in the average power consumption, the linear regression suggests the maximum value has increased from 92.1W to 94.8W, which has 2.9% increase.

Power Consumption before Pre-OS Boot: We also studied the average power consumption right before the power on event, in order to study the energy draw while the computer is powered off and check how it changes over a long time period. We calculate the average values during a 10s window size which ends at the moment when the computer was switched on. As shown in figure 4.12, the average

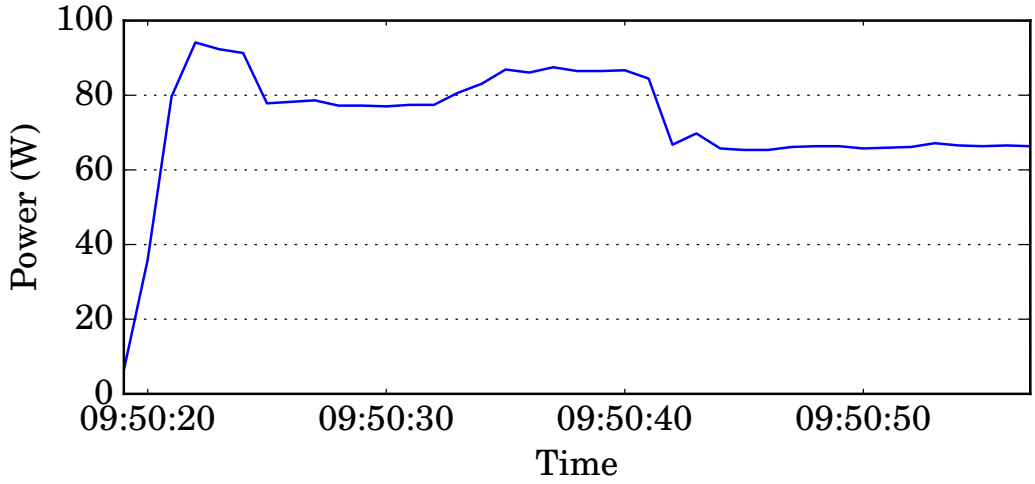


Figure 4.9: Pre-OS booting period detected by using power draw

power consumption values also increased 1.7%. This experiment also confirms that the energy efficiency on the computer drops over long terms of usage. Even during its power off state, the average energy consumption value raised as the value during boot up periods.

We also analyze the total time length of the Pre-OS boot up events. There was a up to 3 seconds of fluctuation in terms of the total time length, the linear regression shows this metric keeps steady during the 18-month period, which confirms with our definition of Pre-OS boot up keeping constants over a long time period. After the computer booted up, it consumes energy every second. So the total energy consumed during the Pre-OS boot up period is highly correlated with the total time length of the Pre-OS boot up event. The linear regression of total energy consumption also suggests this metric stays stable during this 18-month period.

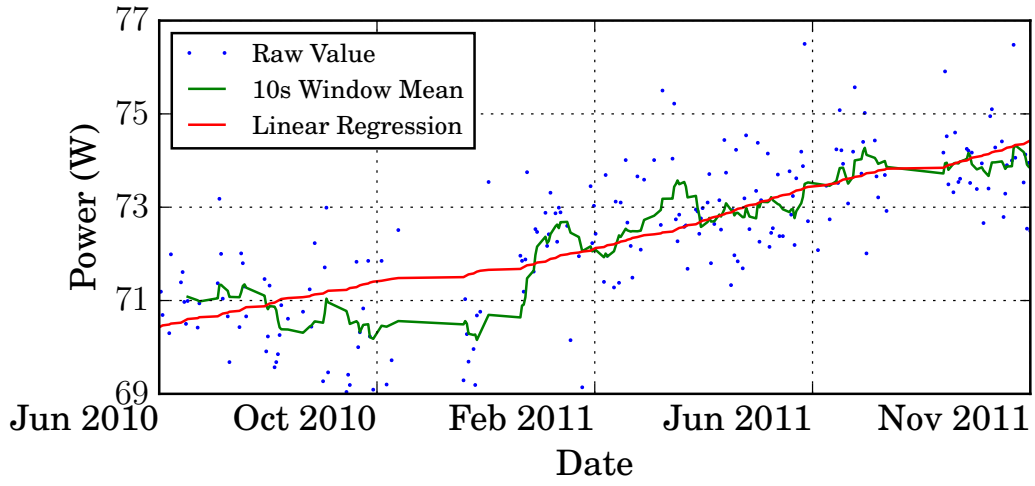


Figure 4.10: Average power draw during the Pre-OS booting period

4.3.3 Idle

In this part, we study the power draw while the computer is in idle state, across a 29-month period. We select two computers in the PowerNet dataset, these two computers are rarely powered off during this 29-month period. Then we plot the power draw during 24 hours every day, this set of figures shows that the power consumption values are consistent during the period between late night until early morning, when usually no user is working at the office. We select the duration between 12AM until 5AM and calculate the average power consumption during this period. Figure 4.13 plots the average values from December 1st, 2010, until May 1st, 2013. The liner regression result, the red line in the figure, shows the average power consumption during the idle period on the computer has increased 0.5W during this 29-month period. Although this is not a significant increase, this observation is consistent with our previous finding in section 4.3.2, which concludes the average

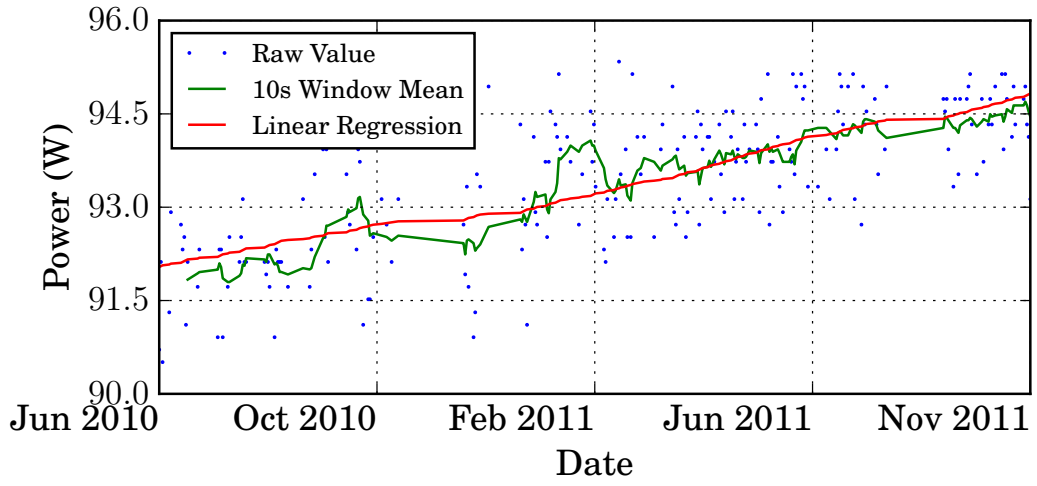


Figure 4.11: Maximum power draw during the Pre-OS booting period

power consumption of a computer will increase over years.

The green curve in figure 4.13 also shows a periodic patterns, in which the value drops to 161.5W around winter in each year, then increases to 164.5W during summer time. This periodic rise and drop per year leads us to study the correlation between the average power consumption and daily average temperature. We retrieved the historic weather data [2] at the city where PowerNet dataset was generated. In figure 4.14, the blue dots are the moving average of the power consumption during idle period with window size 10, the red plus symbol are the same moving average of the temperature value with the same window size. The missing blue dots are due to the interrupted power measurements. This figure shows that the power consumption values are correlated to the temperature value, where the Pearson's correlation is 0.60. When the Pearson's correlation value equals to 1, it means the two variables are have a perfect positive correlation, 0 means has no correlation. The correlation

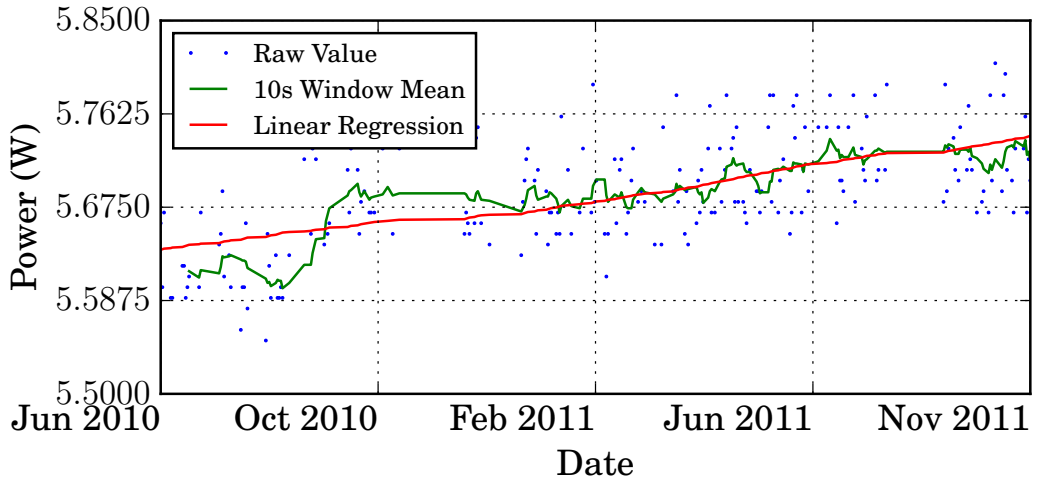


Figure 4.12: Average power draw before the Pre-OS booting period

value between the average daily temperature and average power consumption is 0.60, although it is not a perfect positive correlation, it still shows the power consumption have similar changing trend with temperature. In general, during the summer period, the computer consumes 1.5% more energy compared to the winter period. We found that, the A/C was not working during the off business hours, so that the outdoor temperature affects these computers' working environment. In order to confirm that the power increase during summer time was not caused by the meter reading error, which reports larger readings in the higher temperature environment, we also analyze the power consumption on a network switch, which has fixed power draw regardless of the workload. The same type of meter attached to the switch reports consistent power consumption value over a one year period. The result shows the switch consumes 10W of power consistently without any fluctuation. This verified that the temperature change does not bring any extra reading errors on the power meter. After we verified

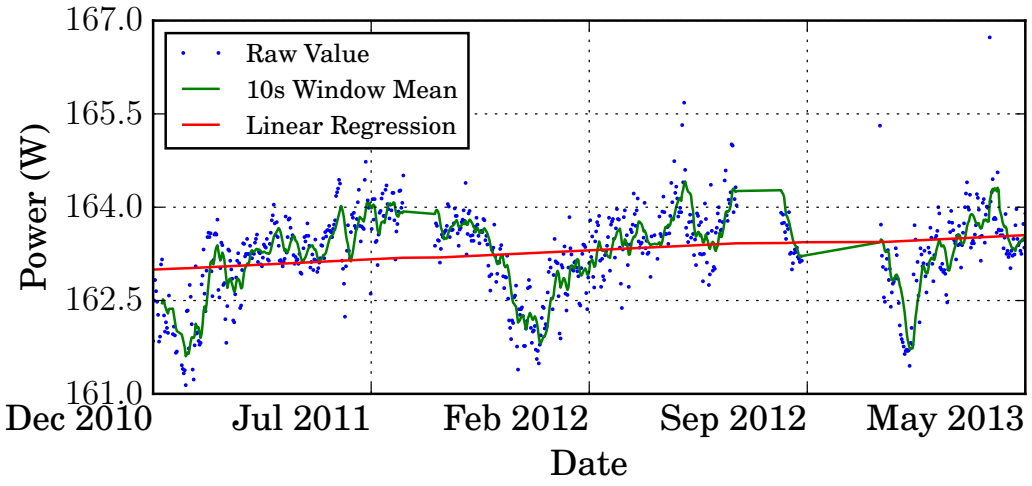


Figure 4.13: Average power draw during Idle Period

the power meter report stable values regardless of the room temperature, then we can confirm the power draw changes of the computer were caused by the workloads changes, which has the possibility caused by temperature variations. One of the reasons that the computer consumes more power during summer period would be because it needs more energy to remove the waste heat. However, since winter has a cooler room temperature, the computer needs less energy to cool itself.

4.3.4 Failure

If we can identify the computer failure only from analyzing the power consumption data on computers, such as detecting computer freeze or BSoD, this method can act as an alternative approach to monitoring the computer states. We are able to use the energy data to identify one type of failure in the PowerNet dataset, on which computers always are forced to have a reboot to recover from that kind of failure.

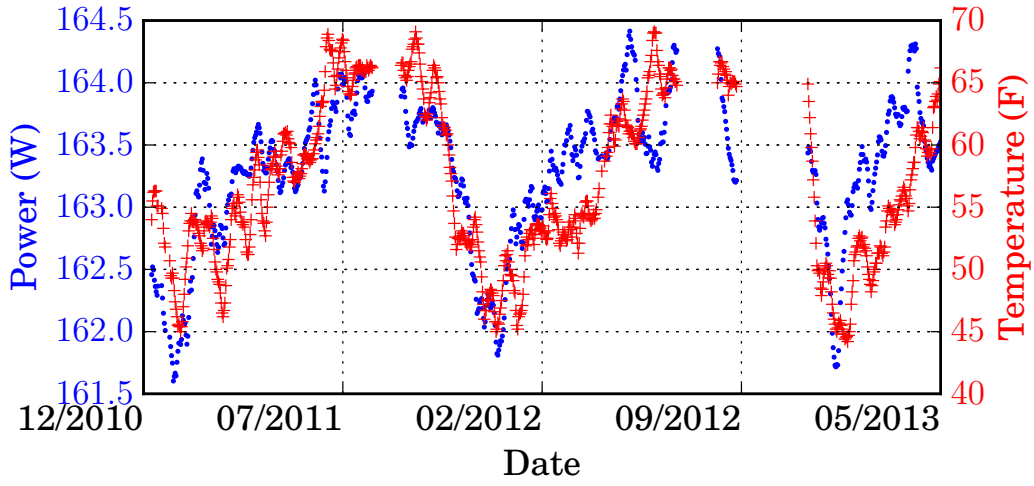


Figure 4.14: Correlation between average value of power consumption and room temperature

Figure 4.15 shows the power consumption of one desktop over 24 hours, on a weekday. During the business hours of 8:30 until 16:30, this computer is being actively used, which can be reflected by the fluctuation of the power readings. This computer was not turned off in the previous day so that the power reading starting from 0:00 is around 40W. As marked in the figure, this computer enters into a failure state around 0:30, from where the power consumption value increases to 62W and keeps very stable at that value for the next 8 hours. This failure ends by a computer reboot which happened at 8:20, when the user started to use the computer. The restart also can be confirmed by analyzing the power draw, which reduced to 0W then followed by a reboot power consumption patterns which we found in section 4.2.3

In the 19 months of power consumption dataset of this computer, we were able

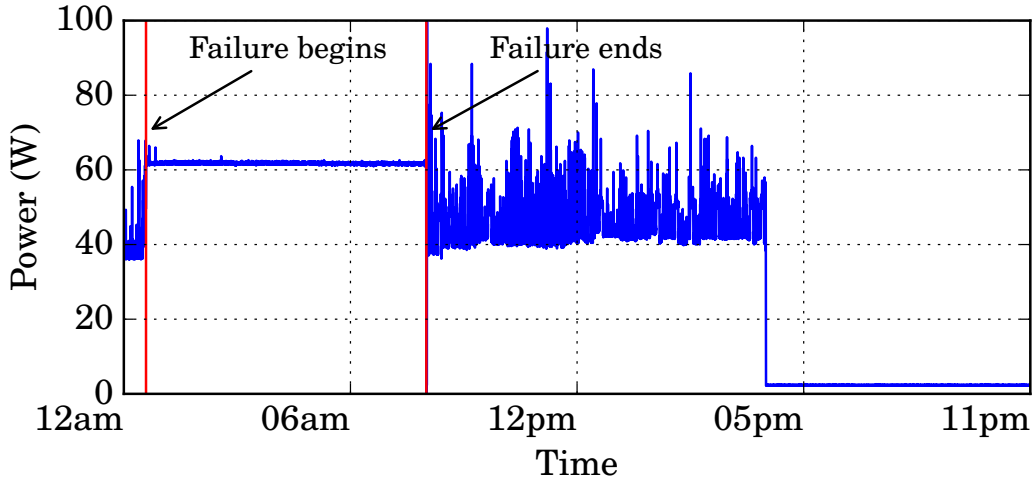


Figure 4.15: Failure event detected by using power draw

to detect this type of failure appeared 8 times. All of these events are followed by a computer restart. We also studied the power consumption data starting from 3 minutes ahead of the misbehavior, no obvious pattern could be retrieved. We need to detect a larger number of computer failures, then cross analyze the energy consumption data from different machines, to check if any potential patterns could be revealed.

4.3.5 Algorithm Stability with Noise

In order to test the error tolerance of our proposed algorithm, we added random noise which following normal distribution to the daily power consumption data. The results show if we add a random noise value in the range 0W to 40W to the every value in the raw data, our proposed algorithm still can accurately detect the boot up event with 94% of precision and 100% of recall. If the the top limit of the random

value increases to 45W, our proposed algorithm performance starts to drop, which reports 39% of precision but with 99% of recall.

4.4 Summary

In this chapter we proposed algorithms to detect various events on computing systems, includes boot up, Pre-OS boot up, idle and failure. These algorithms have been evaluated with the experiments that we performed in a controlled lab environment. The results shows all of these algorithms are able to 95% detect the boot up events. Then we applied these algorithms to understand the power data from a large-scale enterprise computing environment. Our proposed algorithms detected 270 Pre-OS boot up events among an 18-month power trace, 567 idle events among a 29-month power trace, and achieved to locate all of the 8 failures from a 19-month power trace. By detecting all of these events across long period of time, we revealed the insignificant changes of power on computing systems, which could to be due to the components' aging.

Chapter 5

Understanding Networking Systems from Energy Instrumentation

Energy instrumentation has a long history of research in networking systems, especially in wireless sensor network. Energy efficient protocols and applications are one of the key research objectives of Wireless Sensor Network (WSN) research, because the battery life limits the network's operation time. Energy instrumentation and analysis allows us to determine if the proposed protocol is better than the state-of-the-art. Various hardware-based energy instrumentation [21][44], simulation based study of energy footprint [54], and using radio activity as a proxy for energy have all found widespread adoption in the community.

In this chapter, we argue that despite the long history of energy instrumentation

in WSN, we have not fully understood the implications of energy instrumentation in WSNs. Other communities have found that instrumentation of any type must be performed with care. Otherwise, there can be privacy and security implications. Memory access, CPU use patterns, or other types of system information could serve as side channels to reveal information about the user or the workload [68]. Previous work has indicated that power measurements can also act as a side channel with the potential to compromise private information about the users [47]. We study these issues and implications in the context of sensor networks.

To measure energy efficiency of a sensor network protocol, it is sufficient to use some statistical measures such as average of energy used and compare across nodes or protocols. In this work, we found that such coarse statistical measures are inadequate to reveal detailed network activities. Hence, we design a few features based on energy data to be able to better discriminate different activities and other information about the network or the protocol.

We evaluate the design of our energy instrumentation and classification accuracy of those features by doing extensive experiments on three WSN testbeds. In our experiments, we consider four different protocols, two different channels, different topologies, and various application layer packet sizes and transmission intervals. Our results from analyzing four million energy data and radio activity points, indicate that energy instrumentation and carefully designed features can not only reveal information about the network protocol but also some information about the application and the workload.

Although detection of anomalies or validation of node activities is already useful,

we take energy instrumentation one step further by asking: could energy instrumentation be used to understand various aspects of network operation? For example, can we tell what protocol is running in the network? Can we characterize the network topology based on power instrumentation? Can we understand some aspects of the application based on just power data? In this chapter, we attempt to answer these questions as a pretext for a broader discussion on the role energy instrumentation could play in WSN diagnostics. We first examine the properties of power instrumentation necessary to study the above-mentioned implications and opportunities. Coarse-grained power instrumentation, while enough to get a general idea of energy instrumentation, is not sufficient if we want to reveal fine-grained activity information about the network. Such fine-grained energy data may be obtained directly using power meters on some mote platforms or by keeping track of radio on-and-off times if radio is the dominant power using device on the platform.

5.1 Feature and Instrumentation Design

5.1.1 Features Design

In this section, we first describe three commonly used features and then compare them against two new features that we designed to reveal information about the network. The experiment results on three testbeds indicate that our designed features not only significantly outperform commonly used features in terms of classification accuracy, but can also reveal more information that cannot be achieved by existing

features.

5.1.1.1 Energy Consumption (EC)

Statistical values such as average and standard deviation are commonly used to analyse and compare the energy consumption properties of wireless network protocols [6]. On FlockLab, which provides power profiling, we calculate the *mean*, *variance* and *standard deviation* values of energy consumption within a 10s window of the energy profiling data, use these three statistical values as the attributes.

5.1.1.2 Number of Awake Nodes (NAN)

The number of active nodes is the total amount of nodes in the network with their radio chip in awake mode. This number is another commonly used feature to study a protocol's energy performance [66] or its sensing coverage area [32]. Similar to the previous feature, we calculate the *mean*, *variance* and *standard deviation* values of the number of awake nodes within a 10s window size and use them as the attributes.

5.1.1.3 Number of Snooped Packets (NSP)

Snooper node can overhear the nearby transmitting packets. Authors in [58] use multiple snooper nodes to overhear the traffic to study the scalability problem in wireless sensor network. It is common to ask if it is possible to characterize the protocols running on a network using only snooper nodes. We tested this idea by assigning 2 nodes as snooper on FlockLab which can record the timestamps of each

overheard packet. We used a sliding window of 10s, with 1s step size to create a ten-dimensional feature, where each attribute is the number of packets snooped during every second. Our experiment showed that this feature can achieve better classification accuracy compared to using statistical values during 10s window size.

5.1.1.4 High-Power-Length Counter (HPLC)

Before we explain in detail what our proposed feature HPLC is, let us first introduce High-Power Length (HPL). We define HPL as the total time that the power draw on a node stayed in high value during each awake-sleep cycle, e.g., the red dots showed in figure 5.2, which represents the period of radio is in awake state. The length of time a node stays in awake mode is not a fixed value, it depends on the preamble length, the packet size, the time before a node receives acknowledgment, etc. The HPL is calculated by subtracting the previous timestamp of radio-turning-on event from the timestamp of radio-turning-off event. During the experimental period, whenever a node turns on and off its radio, the node will records the timestamp and its related HPL value.

Based on our experiment observations , we used the threshold values 25ms and 100ms to divide the HPL into three categories corresponding to a node performing CCA check, receiving packets and transmitting packets with high possibility, respectively. We name these three ranges as T_1, T_2 and T_3 as defined below, where T presents the HPL of each time:

$$T_1 : T \leq 25ms \tag{5.1}$$

$$T_2 : 25ms < T \leq 100ms \tag{5.2}$$

$$T_3 : 100ms < T \tag{5.3}$$

Within 10s disjoint window size, we count the total number of HPL in each of these ranges, and use these three counters are the feature, named High Power Length Counter, i.e., *HPLC = [m1, m2, m3]* , where m1 is the number of HPLs that satisfy the predicate T_1 (1). m2 and m3 are defined analogously.

In summary, within a given window size, we count the total number of High Power Length (HPL) in each of three ranges (use 25ms and 100ms as the threshold), and use each of these three counters as the feature. The feature vectors of the HPLC are the three counters of the HPL. For example, [199, 20, 50]. On the experiments to classify network protocol, we run four experiments, each of them using one of the four network protocols. Except the network protocols, we tried our best to make the workload similar across protocols, and use identical settings, such as to use the same testbed and the same MAC layer protocol. The purpose of this type of experiment is to identify the HPLC belonging to which category, where each category is the experiment running one of the four network protocols.

The Box plot in figure 5.1 shows the distribution of HPLC across the 1 hour experiments with four protocols each. This figure shows each protocol has its unique distribution of HPLC in the three categories.

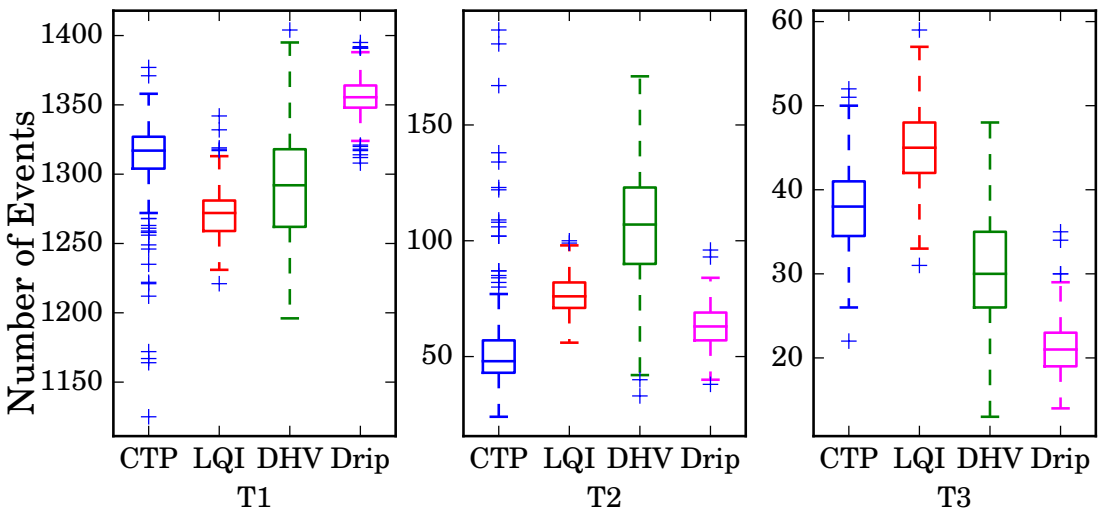


Figure 5.1: Distribution of HPLC of four protocols, each color represents each protocol

5.1.1.5 High Power Overlap Counter (HPOC)

When a node successfully transmits a packet, the radio on both the transmitter and the receiver must be in awake state. In other words, if two nodes have exchanged packets, the intersection of their radio awake time must not be empty, i.e., their power draw in high level must have a overlapped time. We count the times of two nodes have their power draw both stay in high range overlapped during a given period of time, and we call it High Power Overlap Counter. We use this feature to help us infer the network topology in section 5.2.2.1.

In other words, we first calculate the number of times the power draws of two nodes in high range overlapped during a given period of time, in order to get an array with (n-1)*(n-2) values, each value is the HPOC between each node pair.We do not use HPOC for Classification. In this work, for each node A in the network,

we use above array to locate the node B which is at the maximum HPOC with node A. Then we say node A and B have a direct communication.

HPLC and HPOC can be derived from the energy measurements on FlockLab, by using mean-shift clustering algorithm to cluster the energy reading into radio on and radio off classes. On Indriya and Twonet without energy meters, we measure the HPL using software instrumentation. In order to compare the performance between using real energy data and using radio activities, we also use software measuring HPL on FlockLab. In the rest of the paper, we explore which of these four features are useful for different purposes.

5.1.2 Experiment Settings

5.1.2.1 Protocols

The network protocol is a formal set of rules, conventions and data structure that governs how computers and other network devices exchange information over a network [57]. We refer the protocols as the network protocols that are designed for the low power wireless sensor networks.

The purpose of a `Collection Protocol` is to allow a sink to reliably collect the data packets generated from every node in the network. In our experiments, we use Collection Tree Protocol (CTP) [23] and MultiHopLQI(LQI) [60], which are two of the most commonly used multi-hop reliable collection routing protocols. A `Dissemination Protocol` is designed to reliably deliver data packet from the base

station to every node in the network. In our experiments, we use Drip [59] and DHV [13] dissemination protocols.

These protocols use different algorithms, metrics and methodologies to achieve their design purposes. For instance, in the collection protocols, CTP uses Expected Transmission Count (ETX) as the metric to select packet forwarding path, MultihopLQI uses the Link Quality Indicator (LQI) to find the best next hop.

In our work, we define the pattern as the number of packets have been transmitted and received during a given time length. Since, the design principles of the protocols are different, besides transmitting the data packets in the network, different protocol uses different algorithms to generate and transmit control packets. Such as CTP broadcasts its control beacon packet using an adaptive timer, the interval doubles in a stable network. However, MultihopLQI always broadcasts its control beacon packet at a fixed interval. Our proposed feature can not only detect the data packet transmission events, but also can capture the control packet transmissions, therefore, is able to identify different protocols from the energy instrumentation.

We make two assumptions about the protocols for this system to work. The first assumption is that all of these protocols are designed to transmit different amount of control packets during same time period, especially each protocol generates various number of frames on the physical layer even they have same application layer workloads. If all of the protocols sending same amount of packets during the given window size, our proposed work is not able to distinguish them. The second assumption is that there is only one factor has changed in the same set of experiments. Either only the network layer protocol is different, or only the MAC layer protocol is different.

Our proposed work is not able to distinguish the instances with combined different network protocols and different MAC layer protocols.

5.1.2.2 Testbed and Motes

We instrument the power use and radio chips activities on three testbeds. FlockLab provides high-resolution power measurement profiling and GPIO pin tracing. Indriya [15] has over 100 wireless sensor nodes. Twonet [42] is a testbed with dual-radio nodes, which can operate in 2.4 GHz and 900 MHz. We set the Twonet nodes to run on 900 MHz to verify that our proposed approach works with 900 MHz as well. We chose Tmote on FlockLab and Indriya, Opal on Twonet. We use TinyOS for our experiments.

5.1.2.3 Power Readings and Radio Activities

It is conventional wisdom in sensor networking research that radio is the dominant user of energy on a mote. However, even with a simple sensing application, there may be enough energy consumption from sensing, computation, and storage that just keeping track of radio activities may not accurately approximate energy consumption of a mote. To test if this is true on the mote platform we use for our experiments, we ran an application on Flocklab to sample using the light and temperature sensor every 1s and 1.5s, respectively. Every 10 seconds each node combined its last 20 sensor readings and sent it to the root node. In order to make this application closely resemble a real life scenario and use more components on the node, we also

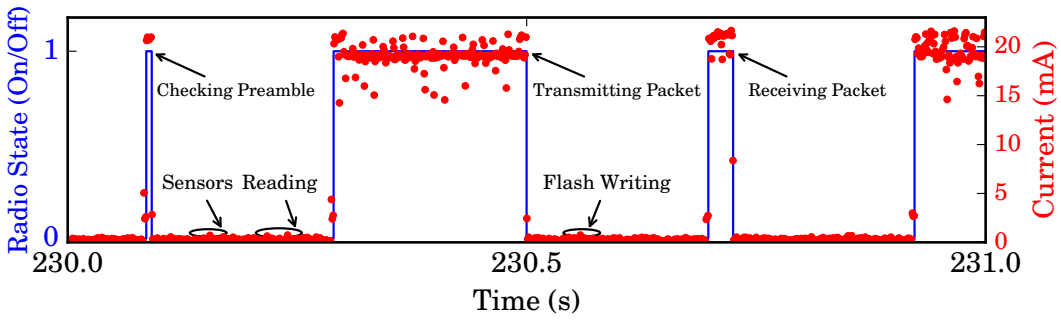


Figure 5.2: The power draw on a wireless sensor nodes

set the node to write the packet contents into its local flash memory every 50s. In figure 5.2, we show the current measurements with 1KHz sampling and annotation indicating various activities on the mote. Even though our sensing application uses sensors and flash storage, radio can still provide a good approximation of total energy use.

5.1.2.4 Low Power Listening(LPL)

The main idea of LPL is to save the energy on the nodes. When using LPL, the node wakes up periodically to perform Clear Channel Assessment (CCA) [48]. The node stays awake until the packet is received if it detects any preamble on the wireless medium. Otherwise it turns off its radio and switches back to sleep mode to save energy. In this study, we set LPL sleep intervals to 200ms.

5.1.3 Evaluation Design

In this section, we describe experiment setup and evaluation design.

We ran four protocols in 16 different configurations (channel, data rate, topology, packet size, sniffers), and we collected the energy instrumentation data and radio activity data. In addition, we performed experiments in 6 configurations with one or two protocols depending on the specific evaluation goals for specific protocols. For example, changing sink node in CTP. We collected 1-3 rounds of data for each protocol-configuration combination. Each round is one hour in length and generates 3.8 - 12.1 million samples of readings.

5.1.3.1 Experiment configurations

The challenge is to identify the network protocol from a identical workload settings. If the workload are different, it difficulties to distinguish each network dropped drastically. Though it is impossible to ensure exactly the same workload across collection and dissemination protocols, we tried our best to make the workload similar across protocols by matching the packet sending interval, using the similar payload size with the same sink node for all of the four experiments in each set.

5.1.3.2 Testbed

We use 30 nodes on FlockLab and 105 nodes on Indriya. These two testbeds use nodes that run on IEEE 802.15.4 2.4GHz. On Twonet, we selected 60 nodes to run on IEEE 802.15.4 900MHz. Together these three testbeds cover the network size from small scale with 30 nodes to large scale with 105 nodes, and also different radio bands of 900MHz and 2.4GHz. To test the stability of our proposed features in the

environment with external interference, we run experiments on overlapping (ch13) and non-overlapping (ch26) channels with WiFi.

5.1.3.3 Classifier Training

For each feature EC, NAN and HPLC, we use a 10s disjoint window to extract the three attributes. Hence, for one hour experiment, we have 360 feature vectors. We then combine all the feature vectors from the 4 experiments belonging to the same set to generate 1440 feature vectors. For NSP, we use a 10s sliding window with a step size 1s to extract 10 attributives for each feature vector.

We test four different types of classifiers, namely J48, Logistic, Bagging and NaiveBayes. These are implemented in Weka [25], which has a collection of machine learning algorithms for data mining tasks.

J48 is an open source Java implementation of the C4.5 algorithm in Weka. C4.5 builds a decision tree, which uses divide-and-conquer ideas to solve the hard problem by recursively split it into several simple sub-problems. At each node of the decision tree, C4.5 selects one attribute to divide the samples into different sub-tree. This operation recurs until all of the samples belong to the same sub-tree, which C4.5 assigns as a leaf node. J48 is one of the common classification algorithms used in machine learning area. One of the main advantages of the J48 classifier is low training cost, and the ability of the algorithm to report the results in a relatively short amount of time.

Nave Bayes is based on the Bayesian theorem of conditional probability with

strong independence assumptions. The design idea of the Nave Bayes is based on the assumption, that the value of a feature is unrelated to the absence or presence of any other feature. The major advantage of Nave Bates is that, it only requires a small amount of training data to train the classifier, and that it is not sensitive to irrelevant features. Due to the design of Nave Bayes, its training and testing speed faster compare to other more sophisticated algorithms, and it usually can give a classification results.

Bagging, also known as Bootstrap aggregating, is an ensemble method that uses random sampling instances with replacement to create separate sample sets of the training dataset, in order to improve the robustness over a single estimator. It is always a good idea to try this method in machine learning problem. For each of the sample sets, one classification algorithm is applied and a model is generated. The results from these sample sets are combined, usually by majority voting or averaging. Bagging classifier might be helpful to build a moderately good classification method by applying the independent aggregating method. Another advantage is that it is possible to run Bagging in parallel mode to reduce the execution time.

Logistic classifier implemented in Weka is using a multinomial logistic regression model with a ridge estimator. In general, the logistic regression is a statistical method adapted to classification, which is used for analyzing a sample dataset with one or more independent variables that determine an output. The main advantage of logistic regression is that only a few statistical assumptions are required for its use, it is simple to use, and have low variance. Our experiments using logistic method shows this classifier performs very well to identify the network protocols.

We use 90% of samples for training and 10% for testing. We perform 10-fold cross-validation to compute the accuracy of different classifiers. The classification accuracy results are averaged across the 10 folds.

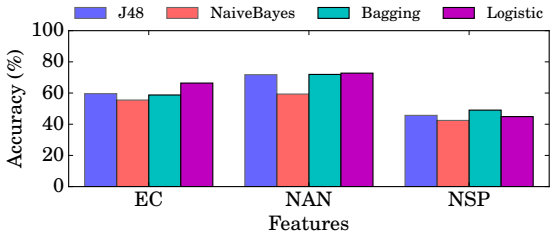
5.1.3.4 Other Settings

We record the routing path in the packet payload and use them only for verification purposes, not for any classification. We skip the instrumentation data collected during the first 20s of an experiment to ensure all the nodes are programmed and the network has warmed up. We use the B-MAC[52], which is the default MAC protocol implementation in TinyOS. We use channel 13 and 26 on 2.4GHz radio band, and channel 6 on 900MHz. We performed experiments on other channels on both 2.4 GHz and 900 MHz bands but do not present those results as they are qualitatively similar to the results we present.

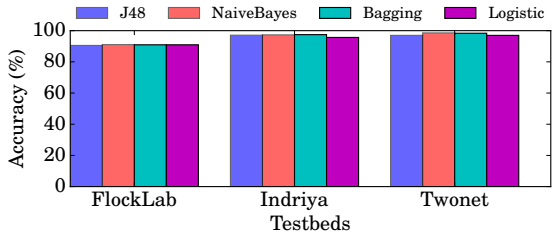
5.2 Evaluation

5.2.1 Identify Routing Protocols

In this section we compare the accuracy of the four classifiers for the task of identifying the protocol running in the network using the four features. If the network is not running as expected, the network has switched to a different protocol, we can detect them by using our proposed feature.



(a) Classification accuracy across three commonly used features on FlockLab



(b) Classification accuracy using HPLC across three testbeds.

Figure 5.3: Classification accuracy using four features.

5.2.1.1 Performance with different features

Figure 5.3(a) shows the results of experiments performed on FlockLab using the three commonly used features in terms of four classification algorithms. EC achieves 59% accuracy using J48. NSP only reports 42.8%. The best accuracy is achieved by using NAN, which is 70.4%. We plot the results by using HPLC calculated by software measurements on radio activities in the first group of figure 5.3(b), which shows all four algorithms of using HPLC could achieve more than 90% accuracy. Figure 5.3(b) also shows the classification accuracy of using software measured HPLC on Indriya and Twonet, where the average accuracy is above 97% and 98%, respectively. These two figures show three highlights of HPLC: it gives the highest classification accuracy compared to the other three features, it gives the most stable results over the four classification algorithms; it generates the most stable accuracy results on three testbeds, with different network layout and different radio bands. Note, in order to compare the performance of four features on the same testbed, we extracted all of them from the same experiment on FlockLab.

To test the performance of HPLC with an external interference, we also study the

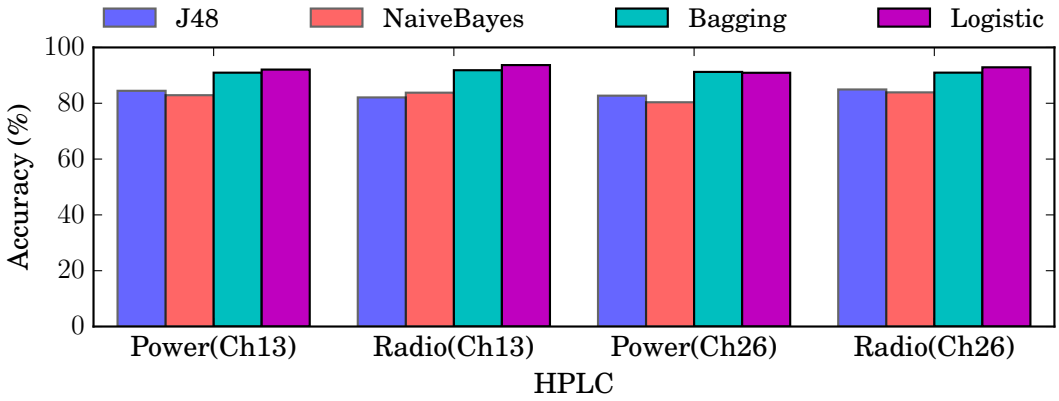


Figure 5.4: Classification accuracy using HPLC

classification accuracy of using HPLC extracted from directly measuring the power consumption or software measuring the radio activities. Figure 5.4 shows the accuracy results in the experiments performed using channel overlapping with WiFi in the left two groups and not overlapping with WiFi in the right two groups. In each channel, the HPLC retrieved by power and radio are also plotted. All of the four groups achieve similar accuracy. Depending on the algorithm, the accuracy is in the range of 82% and 92%. These two figures clearly show our proposed feature HPLC can achieve high accuracy, across different testbeds, network size, radio channels. The reason why HPLC computed using radio instrumentation reports higher accuracy compare to HPLC computed using power instrumentation, is because the radio instrumentation is measured by interrupt, which records the accurate radio On and Off event time. However, the 100Hz power profiling only measures the instantaneous current value every 10ms, it is possible to miss some very short power changes between every two samples. We also evaluate HPLC by training it on sample sets from one experiment configuration and then testing it on data from another experiment

configuration. We have 7 such cases and HPLC's accuracy was between 82-97%. Since power meter is only provided on FlockLab, we are using the software retrieved HPLC in rest of the paper.

The reason why HPLC gives the best classification results is because it can capture the unique patterns between the protocols. Even with similar workload, due to their protocol design, the control messages of each protocol is designed uniquely, e.g. the time interval between transmit control packets and number of control packets. While the workloads from the application layer are the same, using HPLC makes it feasible to distinguish the protocols by looking at the patterns in radio activities triggered by transmit and receive packets, including data packets and control packets.

Although Energy Consumption (EC) could capture the node activities over a period, unlike HPLC, which takes into account the time length of each wake up event, EC lacks this timing information. Also, EC only capture the accumulative energy consumption during that given period, it can not detect any radio behaviors. With an identical application layer workload, EC does not have an accurate measurement of each node's activity, which makes it is hard to classify the protocols correctly.

The Number of Awake Nodes (NAN) in the network is not equal to the number of nodes performing transmission operations. The awake nodes may be performing CCA checks or waiting for the acknowledgement packet. Meanwhile, this accumulative number of awake nodes may remain unchanged when there are a certain number of nodes that switch on and an equal number nodes that switch off at the same time. Hence, NAN does not capture the differences in nodes transmission activities

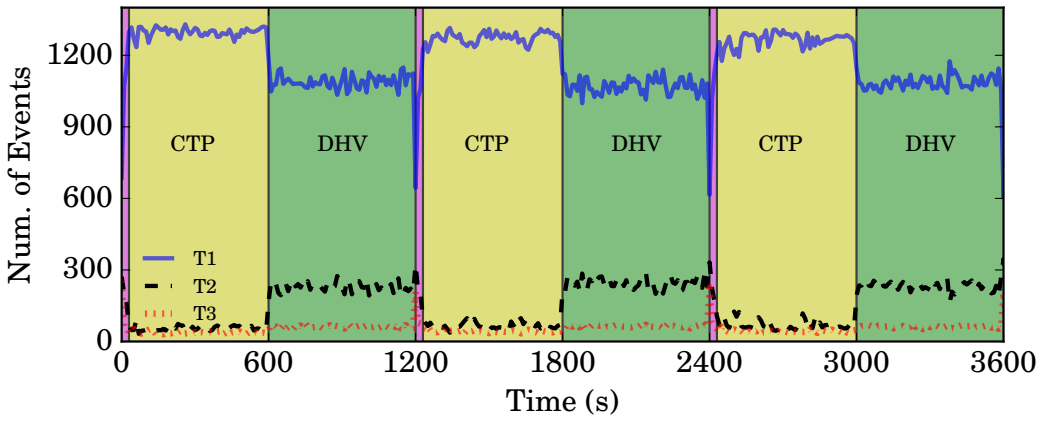


Figure 5.5: Unsupervised clustering results of alternative protocols

in sufficient granularity to be useful for classifying protocols.

Number of Snooped Packets (NSP) are highly related to the physical location of the snooper node. The number of snooped packets is much larger if the snooper is near the sink compared to at edge of network. When we run LPL in the network, the snooper also captures a large number of duplicate packets due to retransmission. Even though we found that using the 10 attributes in a sliding window is better than the statistic values across disjoint window, the overall classification result is less than 45%.

5.2.1.2 Cluster Analysis of Alternating Protocols

Next, we evaluate the effectiveness of HPLC for clustering two protocols running during different periods. We switch back and forth between two protocols several times during a one-hour experiment. All 30 nodes on FlockLab run CTP for 600s, then stop CTP and start DHV for the next 600s, so on and so forth. We use a

general non-parametric cluster algorithm, MeanShift [10] to cluster the HPLC from the measurements. We use a 10s disjoint window to calculate the HPLC. Out of 360 snapshots, only 18 of them were mis-clustered; thus, the percentage of correctly clustered snapshots is 95%. In Figure 5.5, yellow and green backgrounds show the periods with correct clustering, while red shows the mis-clustered period. All of mis-clustered periods (red) happen right after CTP starts. During the warm up period of CTP, the nodes exchange a lot of control packets to setup routing paths, so the radio transmit more packets compared to the stable period. This causes the algorithm to mis-cluster CTP as DHV because DHV has more T_2 events compare to CTP. This experiment confirms that HPLC can correctly identify the protocols running during different periods, and can also detect the moment when the protocol changes. Because of its design, HPLC captures the change in control overhead caused by a protocol switch. Hence, we expect our proposed approach to also cluster three or more protocols, and even to detect the moment when the workload changes.

We also study the number of snooped packets logged by one and two snoopers placed in the network during the same experiment; however, there are no clear patterns that can help us distinguish between the two protocols. Thus, we show that naive features such as number of snooped packets are not sufficient to correctly identify the protocols, while features such as HPLC can accurately distinguish between different protocols.

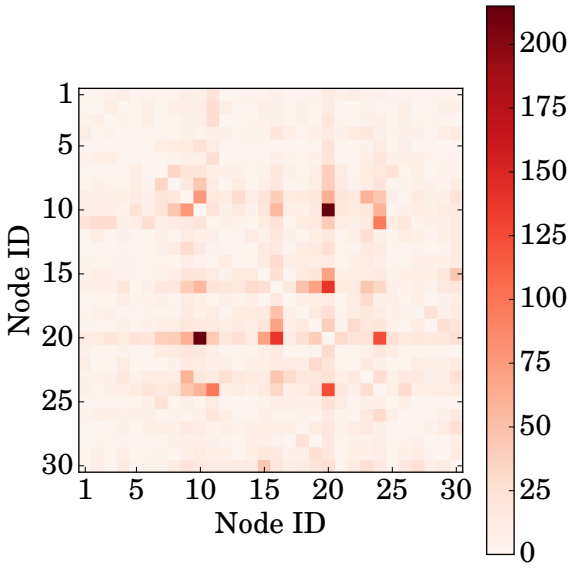


Figure 5.6: Heatmap of HPOC. Darker color shows larger number of HPOC between each pair of nodes.

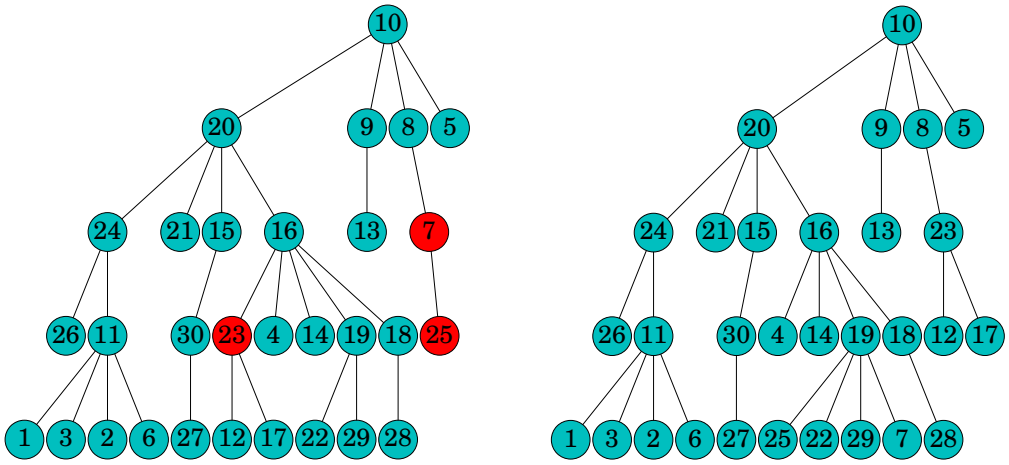
5.2.2 Infer Network Topology

Next, we study the effectiveness of HPOC in revealing information about the network topology and routing path for each node.

5.2.2.1 Parent Node and Routing Path

The HPOC across two nodes can be used to estimate the parent-child relationship across the network running multi-hop collection protocol such as CTP. If two nodes have a parent-child relationship, then each time they exchange a packet, the radio on time on the nodes will overlap. We remove the radio overlap length that are too short (less than 0.025s) to focus on significant overlaps in our analysis. If a parent has multiple children, the parent will have distinct overlaps with each child.

Figure 5.6 shows the heatmap of HPOC across every node-pair during the 200s



(a) Inferred Topology (b) Actual Topology

Figure 5.7: Use HPOC to reveal the topology of whole network. Green circles indicated the node with correct parent node, red circles indicate the node with wrong parent node.

window size on FlockLab. The darker color represents those two nodes having larger overlapped times. A heuristic to find the parent for a node is to simply designate the node with which a node has the largest overlap as its parent. For example, for y=20, the pixel at x=10 is darkest. Hence, we guess that node 10 is the parent of node 20. If multiple nodes have the same overlap length, we use overlap information from adjacent time window. We use this heuristic for each node in the heatmap and construct the routing topology, which is shown in figure 5.7(a). We found that this inferred topology based on the heatmap is surprisingly close to the actual routing topology shown in figure 5.7(b). Only the nodes marked red had the wrong estimate of routing parent. We ran CTP and LQI multiple times on the testbed and used the heuristics above to estimate the routing topology. The estimation accuracy across the experiments was 97.8% and 90.2% for CTP and LQI protocol, respectively.

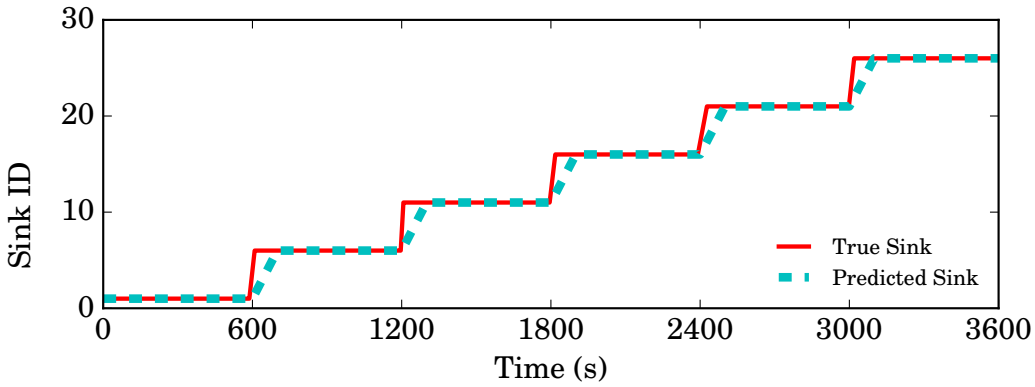


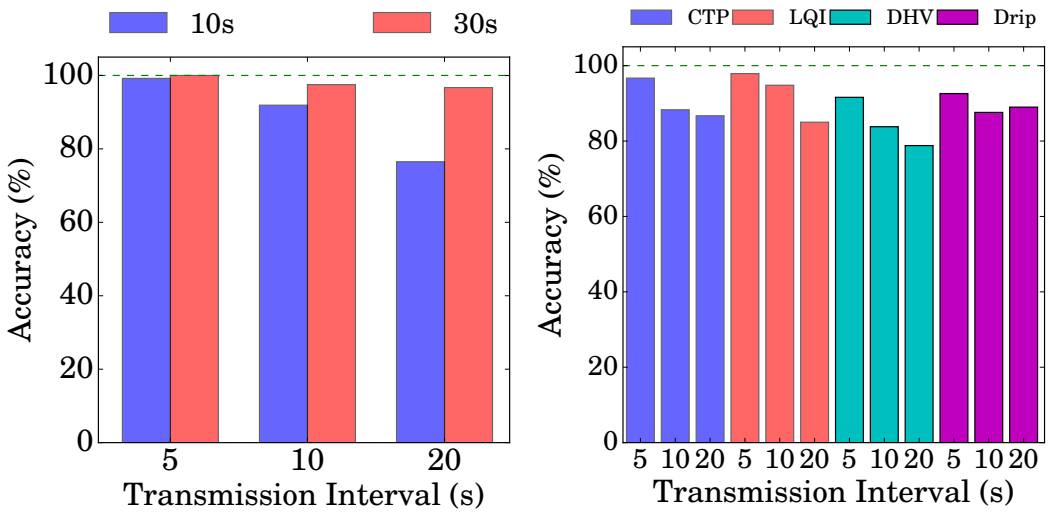
Figure 5.8: Use HPLC to detect sink node changes

5.2.2.2 Sink Node

Next we study how to identify the sink with HPLC. During each 100s window period, the nodes with the maximum number of T_2 had the highest probability to be the sink node, since the T_2 could reflect the number of receive events. We ran CTP for one hour, where the sink node changed every 600s. The red curve in figure 5.8 shows the true sink node ID while the blue curve shows the predicted sink node ID. The result shows that identifying the sink using HPLC is accurate and feasible. The slight lag between predicted and actual sink is due to the 100s window when we calculate HPLC.

5.2.2.3 Leaf Nodes

The number of events belonging to the range of T_3 can give a rough approximation of the number of transmissions by a node. If this number is less than the average of



(a) Packet transmission interval 5s, 10s, 20s on CTP

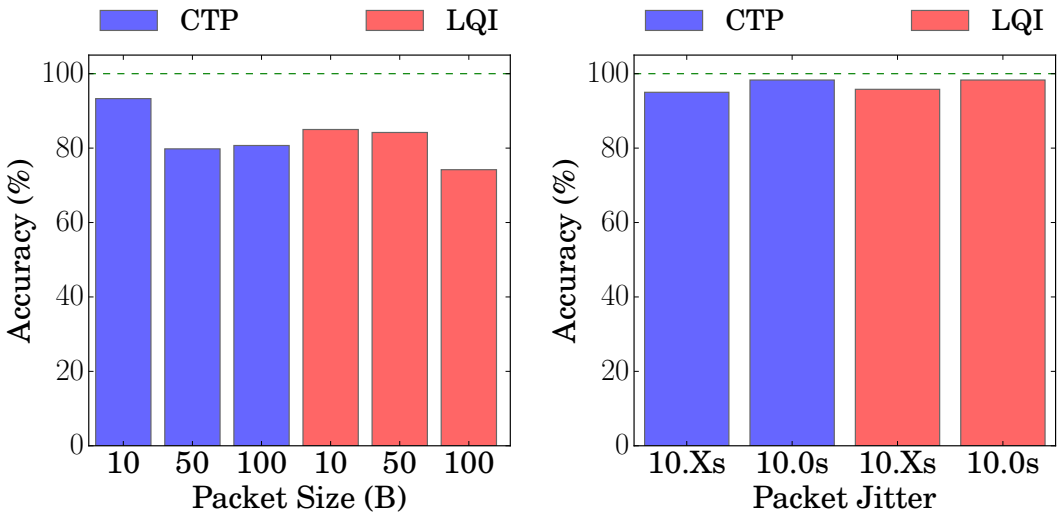
(b) 12 combinations of intervals and protocols

Figure 5.9: True Positive Rate of using HPLC to detect application throughput

all the nodes, we can mark that node as a leaf. It turns out, even this simple method can identify leaf nodes with accuracy of up to 95.1% and 93.3% for CTP and LQI, respectively.

5.2.3 Determine Application Workloads

In this section, we evaluate the effectiveness of HPLC to distinguish different application workloads, including various packet transmission interval, application layer payload size, and packet jitter. In this section, all of the experiments were run on FlockLab. We used J48 algorithm to perform the classification test.



(a) 6 combinations of packet size and routing protocols

(b) 4 combinations of packet jitter and routing protocols

Figure 5.10: True Positive Rate of using HPLC to detect packet size and jitter

5.2.3.1 Packet Transmission Interval on Same Protocol

We run CTP with data being generated at three different intervals: 5s, 10s, and 20s. The same strategy was used to generate the dataset, i.e., retrieve feature vectors from each test case then mix them all together. We calculate the true positive rate when classifying each interval from the mixed dataset. We use 10s as the window size to calculate the HPLC, and its corresponding true positive rate are plotted with blue color in Figure 5.9(a). The first bar shows when the interval is equal to 5s, the true positive rate is 99.2%. Then true positive rate decreases to 91.9% and even drops to 76.5% for interval length 10s and 20s, respectively. The drop is due to the packet transmission interval becoming larger than the HPLC window. Thus, true positive rate increases with a larger HPLC window size (30s), especially when it is longer than the interval value, as shows in the same figure with red color.

5.2.3.2 Packet Transmission Interval over Various Protocols

In Figure 5.9(b), we plot the results from determining packet intervals across four protocols. The true positive rate can reach up to 98.3%. The average accuracy to classify one of the instances from all of the 12 combinations of 4 protocol and 3 intervals is 87.5%. Furthermore, the true positive rate of classifying four protocols with a 5s transmission interval from the 12 combinations are all above 90%.

5.2.3.3 Packet Size

We then vary the data packet size sent with CTP and LQI from 10 to 50 to 100 bytes. The dataset includes a total 6 distinct types of instances, which are the combination of two protocols and three packet sizes. Figure 5.10(a) shows the results. The overall accuracy to classify one instance from 6 combinations is 82.8%, and it turns out that a smaller packet size is more likely to be classified correctly.

5.2.3.4 Packet Jitter

To understand if HPLC can be used to determine if data packets are sent strictly periodically or with jitter, we perform two experiments. In the experiment with jitter, we set the data packet interval to be a random value between 10s and 11s, we name this type as 10.Xs. In the experiment without jitter, we send a data packet every 10s sharply, marked as 10.0s. If most of the nodes are send the packet with exactly the same interval, the possibility of congestion and re-transmission will increase, which will cause more packet transmission events(T_3). In the network with

jitter, the collision chance is lower due to each node sending packets with different intervals. Figure 5.10(b) shows the true positive rate to distinguish one of the four combinations of two protocol with and with out jitter, all of the four true positive rate are above 95%.

5.3 Summary

In this chapter, we demonstrated that energy instrumentation can be a powerful tool to study and reveal information about the network, protocol, or workload. We designed features for classification and analysis based on energy instrumentation. We found that the feature called High Power Length Counter is especially versatile in revealing information across protocols and application workload, such as up to 97% accurate for classifying protocols, 100% accurate for locating the sink node, and an average of 87.5% accurate for classifying workloads. Furthermore, another feature named High-Power Overlapped Counter could reveal the parent node for each node, even to disclose the actual network topology with 98% accuracy. Our extensive experimental results performed on three different testbeds over 100 test cases suggest that our proposed features are robust across the testbeds, radio bands and radio channels, and even outperform existing commonly used features in terms of stability over various algorithms.

Chapter 6

Conclusion

This dissertation addressed the question about how to use the energy consumption to understand the computing and networking systems. We investigated the instrumentation designs, workload classification, patterns during various behavior, and identify the energy waste in a computer lab.

6.1 Summary of Contributions

6.1.1 Understanding Desktop Energy Footprint in an Academic Computer Lab

We study how to design and implement instrumentation for analyzing power consumption in an academic computer lab, and present a large dataset describing both

computing power data and user activities in a shared homogeneous computing environment. Power draw and user activities are compared and their relationship is studied to identify energy waste in the academic lab setting. We found that the energy consumption of each computer was highly related to individual user behavior, and that 60% of energy consumed every day was while the computer was on and no one was logged in.

6.1.2 Energy Measurements and Analysis to Understand Computing Systems

The energy instrument system is implemented to collect and analyze power consumption data on computing systems. With the aim of revealing computer system health status, we studied energy patterns in various computer activities, particularly during the system boot up, idle and failure periods. Over 30 months of data we succeeded in modeling the relationship between these events and the energy patterns.

6.1.3 Revealing Protocol Information and Activity from Energy Instrumentation in Wireless Network

Based on the instrumentation for energy analysis we extended our work from single computer activities to the network level analysis. Network level protocol, and application information were successfully inferred from features based on energy data. The repeated experiments on three real-world testbeds increased our confidence in

the accuracy results, which evaluated the proposed features across multiple protocols, network topologies, and application workloads. The routing protocol was successfully identified with an accuracy of 97%. The application workloads were detected with 87.5% accuracy. We then further demonstrated how to reveal the routing topology in the network, including the next hop for each node, based solely on energy instrumentation; we reached an accuracy of 98%.

6.2 Open Issues and Future Work

6.2.1 Limitations of the Energy Instrumentation System

We have designed the energy instrumentation architecture for each system in this work. Although each system has its own constraints and requirements, it is hard to extend our existing instrumentation system to a new development. The PowerNet meter we used to measure the energy consumption in the computer lab supported wireless transmission, however, it did require a dedicated low power wireless interface device as the gateway to receive the message on the computer. The specific per-outlet sensing power strip which monitored the current draws in the data center is inconvenient to transport and to monitor regular computing devices. The hall-effect sensor held the advantage of accurate sensing, but it require an additional DAQ device to capture the readings.

It might be a benefit for the community, if we can build a portable and universal

power monitoring device. This device should provide following features: (1) Plug-and-play: like a socket adapter, plug this device between the A/C socket and the monitor device, then it monitor function should be ready to work. (2) Support WiFi: the measured energy data should be able to send to a remote database or local computer though WiFi, using an extra cable to read the data is very inconvenient and hard to develop in a large scale. (3) Adjustable sampling rate: the sampling frequency is able to adjust for different purposes. (4) Low error rate: the value of the data becomes low if the error rate of the data is high.

6.2.2 Performance of Healthy Monitor on Computing Systems

Before we can evaluate the information revealed from energy data, we must have a reliable monitoring system to collect the ground truth on the computing system. On our current implementation of system activity monitor system, one of the limitation is it only collects CPU and memory utilization information, but it does not collect other resource utilization information, such as hard drive, network interface or CPU frequency. In our preliminary work, we have demonstrated that besides the CPU, GPU also dominates the total power consumption on a computer which has a GPU. The challenge is how to reduce the resource consumption of the monitoring program when it checks every component on the system at a very fast speed, plus the workload introduced by transmitting these reading results to a remote data center.

In addition to the components utilization for monitoring, how to efficiently monitor the system failure is another open research topic. It is impossible to let the system record all of its failures which have happened on itself, and it is also a challenge for a external system to identify the failures.

6.2.3 Using Energy Data to Predict Failure

In this work we have achieved understanding of the system's various information after we obtain its energy consumption data. We can use the real-time energy data to predict the system's behavior. For instance, if we can predict the system failure, it will be a significant benefit to the community. Recognizing the energy patterns right before the failure not only can predict it, but also enables the chance to prevent the system from going into failure. However, how to reproduce the innumerable failure types on the computing system is more of a challenge than researching the energy patterns.

Bibliography

[1] Windows management instrumentation. `http://msdn.microsoft.com/en-us/library/windows/desktop/aa394582(v=vs.85).aspx`, note = "[Online; accessed Nov-10-2014]".

[2] Weather underground historical weather. `http://www.wunderground.com/`, 2014. [Online; accessed Nov-5-2014].

[3] A. Acharya, G. Edjlali, and J. Saltz. The utility of exploiting idle workstations for parallel computation. *SIGMETRICS Perform. Eval. Rev.*, 25(1):225–234, June 1997.

[4] H. Alemzadeh, R. K. Iyer, Z. Kalbarczyk, and J. Raman. Analysis of safety-critical computer failures in medical devices. *Security & Privacy, IEEE*, 11(4):14–26, 2013.

[5] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE computer*, 40(12):33–37, 2007.

[6] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[7] M. Calle and J. Kabara. Measuring energy consumption in wireless sensor networks using gsp. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pages 1–5. IEEE, 2006.

[8] M. M. D. Center. Task idle conditions. `http://msdn.microsoft.com/en-us/library/windows/desktop/aa383561(v=vs.85).aspx`. [Online; accessed Nov-5-2014].

[9] H. Chen, G. Jiang, and K. Yoshihira. Failure detection in large-scale internet services by principal subspace mapping. *Knowledge and Data Engineering, IEEE Transactions on*, 19(10):1308–1320, 2007.

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[11] M. Cornil, L. Colman, and G. De Mey. Power consumption measurements on portable pc's. In *Mixed Design of Integrated Circuits and Systems (MIXDES), 2011 Proceedings of the 18th International Conference*, pages 549 –551, june 2011.

[12] P. Cremonesi, A. Sansottera, and S. Gualandi. Optimizing cooling and server power consumption. In *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pages 455 –462, aug. 2011.

[13] T. Dang, N. Bulusu, W.-C. Feng, and S. Park. Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks. In *Wireless Sensor Networks*, pages 327–342. Springer, 2009.

[14] S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. Culler. @ scale: insights from a large, long-lived appliance energy wsn. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 37–48. ACM, 2012.

[15] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. Indriya: A low-cost, 3d wireless sensor network testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 302–316. Springer, 2012.

[16] P. Domingues, P. Marques, and L. Silva. Resource usage of windows computer laboratories. In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pages 469 – 476, june 2005.

[17] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. 2011.

[18] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32. ACM, 2007.

[19] EC-Council. Windows, linux,and macintosh boot processes. In *Computer Forensics: Hard Disk and Operating Systems*, chapter 3. Cengage Learning, 1st edition, 2009.

[20] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

[21] R. Fonseca, P. Dutta, P. Levis, and I. Stoica. Quanto: Tracking energy in networked embedded systems. In *USENIX Symposium on Operating Systems Design and Implementation*, volume 8, pages 323–338, 2008.

[22] R. Giri and A. Vanchi. Increasing data center efficiency with server power measurements. `http://resources.spiceworks.com/banners/ibm/europe/whitepapers/ServerPowerMeasurement.pdf`.

[23] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2009.

[24] L. Gong, J. Xie, X. Li, and B. Deng. Study on energy saving strategy and evaluation method of green cloud computing system. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pages 483–488. IEEE, 2013.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[26] D. Han and O. Gnawali. Understanding Desktop Energy Footprint in an Academic Computer Lab. In *Proceedings of the The IEEE International Conference on Green Computing and Communications (GreenCom 2012)*, November 2012.

[27] D. Han and O. Gnawali. Understanding desktop energy footprint in an academic computer lab. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 541–548. IEEE, 2012.

[28] G. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *Technology and Society Magazine, IEEE*, 8(2):12 –16, june 1989.

[29] D. Heap. Taurus-a taxonomy of actual utilization of real unix and windows servers. *IBM White Paper GM12-0191*, 2003.

[30] W. B. Heinzelman. *Application-specific protocol architectures for wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[31] T. Hnat, V. Srinivasan, J. Lu, T. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. The hitchhiker's guide to successful residential sensing deployments. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 232–245. ACM, 2011.

[32] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. *Mobile Networks and Applications*, 10(4):519–528, 2005.

[33] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and implementation of a high-fidelity ac metering network. In *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on*, pages 253–264. IEEE, 2009.

[34] X. Jiang, M. Van Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a high-fidelity wireless building energy auditing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 113–126. ACM, 2009.

[35] M. Kazandjieva, O. Gnawali, B. Heller, P. Levis, and C. Kozyrakis. Identifying energy waste through dense power sensing and utilization monitoring. Technical report, Stanford University, Tech. Rep. CSTR 2010-03, 2010.

[36] M. Kazandjieva, B. Heller, D. Gal, P. Levis, C. Kozyrakis, and N. McKeown. Powernet: A magnifying glass for computing system energy. In *Proc. Stanford Energy & Feedback Workshop: End-Use Energy Reductions through Monitoring, Feedback, and Behavior Modification*, 2008.

[37] Y. Kim, T. Schmid, Z. Charbiwala, and M. Srivastava. Viridiscope: design and implementation of a fine grained power monitoring system for homes. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 245–254. ACM, 2009.

[38] A. Krioukov and D. Culler. Personal building controls. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, pages 157–158, New York, NY, USA, 2012. ACM.

[39] B. Krithika and N. Keerthana. Comparison of intel processor with amd processor with green computing. In *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, pages 737–742. IEEE, 2013.

[40] G. S. Kumar, L. A. Varghese, K. Mathew, and K. P. Jacob. Evaluation of the power consumption of routing protocols for wireless sensor networks. In *Ad*

Hoc and Ubiquitous Computing, 2006. ISAUHC'06. International Symposium on, pages 194–195. IEEE, 2006.

[41] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. Tinyos: An operating system for sensor networks. pages 115–148, 2005.

[42] Q. Li, D. Han, O. Gnawali, P. Sommer, and B. Kusy. Twonet: Large-scale wireless sensor network testbed with dual-radio nodes. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 89:1–89:2, New York, NY, USA, 2013. ACM.

[43] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. Paradiso. A platform for ubiquitous sensor deployment in occupational and domestic environments. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 119–127. IEEE, 2007.

[44] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 153–166. ACM, 2013.

[45] A. Marchiori and Q. Han. Using circuit-level power measurements in household energy management systems. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pages 7–12, New York, NY, USA, 2009. ACM.

[46] F. Mattern, T. Staake, and M. Weiss. Ict for green: how computers can help us to conserve energy. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 1–10, New York, NY, USA, 2010. ACM.

[47] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM, 2010.

[48] D. Moss, J. Hui, and K. Klues. Low power listening. *TinyOS Core Working Group, TEP*, 105, 2007.

[49] S. D. Muruganathan, D. C. Ma, R. I. Bhasin, and A. O. Fapojuwo. A centralized energy-efficient routing protocol for wireless sensor networks. *Communications Magazine, IEEE*, 43(3):S8–13, 2005.

[50] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, 2006.

[51] M. Poess and R. O. Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results. *Proc. VLDB Endow.*, 1(2):1229–1240, Aug. 2008.

[52] 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*, pages 95–107. ACM, 2004.

[53] H. Serra, J. Correia, A. Gano, A. de Campos, and I. Teixeira. Domestic power consumption measurement and automatic home appliance detection. In *Intelligent Signal Processing, 2005 IEEE International Workshop on*, pages 128 – 132, sept. 2005.

[54] V. Shnayder, M. Hempstead, B.-r. Chen, and M. Welsh. Powertossim: Efficient power simulation for tinyos applications. 2004.

[55] T. Stathopoulos, D. Mclntire, and W. J. Kaiser. The energy endoscope: Real-time detailed energy accounting for wireless sensor nodes. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 383–394. IEEE, 2008.

[56] E. T. Winter and E. P. Thubert. Rpl: Ipv6 routing protocol for low power and lossy networks. `http://tools.ietf.org/html/draft-ietf-roll-rpl-19`, 2011.

[57] J. Technologies. *Network Protocols Handbook*. Javvin Technologies Inc., 2005.

[58] S. Tennina, O. Gaddour, F. Royo, A. Koubaa, and M. Alves. Monitoring large scale ieee 802.15. 4/zigbee based wireless sensor networks. 2013.

[59] TinyOS. The drip protocol. `https://github.com/tinyos/tinyos-main/tree/master/tos/lib/net/drip`.

[60] TinyOS. Multihoplqi collection protocol. `https://github.com/tinyos/tinyos-main/tree/master/tos/lib/net/lqi`.

[61] L. Wang and T. Wang. Green computing wanted: electricity consumptions in the it industry and by household computers in five major chinese cities. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, pages 226–229. IEEE, 2011.

[62] C. Wei and Y. Li. Design of energy consumption monitoring and energy-saving management system of intelligent building based on the internet of things. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 3650 –3652, sept. 2011.

[63] M. Woehrle, J. Beutel, R. Lim, M. Yuecel, and L. Thiele. Power monitoring and testing in wireless sensor network development. In *Workshop on Energy in Wireless Sensor Networks (WEWSN)*, 2008.

[64] M. Woehrle, K. Lampka, and L. Thiele. Exploiting timed automata for conformance testing of power measurements. In *Formal Modeling and Analysis of Timed Systems*, pages 275–290. Springer, 2009.

[65] R. Yamini. Power management in cloud computing using green algorithm. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 128–133. IEEE, 2012.

[66] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(3):493–506, 2004.

[67] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.

[68] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 305–316. ACM, 2012.