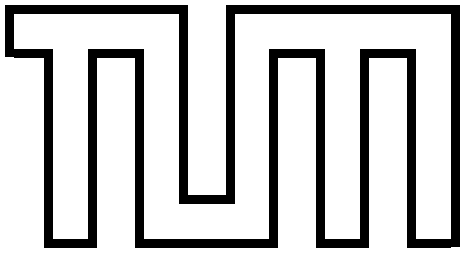


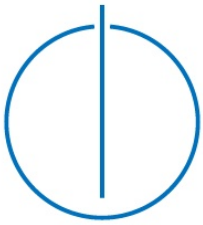
Technische Universität München

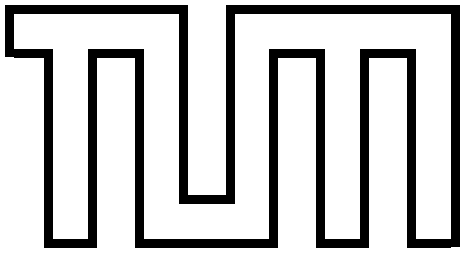
Fakultät für Informatik

Master's Thesis in Informatik

Design and Implementation of a Distributed Real-Time Demand Response Infrastructure for Laptops

Pezhman Nasirifard





Technische Universität München

Fakultät für Informatik

Master's Thesis in Informatik

Design and Implementation of a Distributed Real-Time Demand Response Infrastructure for Laptops

Design und Implementierung einer verteilten, echtzeitfähigen Demand Response Infrastruktur für Laptops

Author: Pezhman Nasirifard

Supervisor: Prof. Dr. Hans-Arno Jacobsen

Advisors: Dipl.-Ing. (Univ.) Jose Rivera, M.Sc. Martin Jergler

Submission: 05.07.2017

I confirm that this master's thesis is my own work and I have documented all sources and material used.

München, 05.07.2017

(Pezhman Nasirifard)

Abstract

With the further integration of Renewable Energy Sources (RES), the complexity of power grids is increasing. Due to the fluctuating nature of RES, ensuring the reliability of power grids can be challenging. For this reason, over the last years, the researchers and practitioners in the field of Smart Grids (SG) have been offering solutions addressing these concerns. One interesting concept of SG is Demand Response (DR) which is defined as changing the demand for electrical energy according to the changes of supply. However, implementing such a system which is capable of maintaining the power consumption of a vast number of electrical appliances introduces several more complications including reliability, scalability, security, practicality, and financial feasibility. To address these issues, we propose a design and implementation of a DR infrastructure for laptops. Our approach is capable of monitoring the energy consumption as well as controlling the power consumption of several laptops in real-time. Furthermore, we emphasize on offering a purely software-oriented approach for executing DR activities, reducing the initial costs for the demand side participants to zero. Moreover, we conduct experiments on a number of laptops to measure the effectiveness and performance of our design. We verify that our system successfully performs effective DR events. However, the accuracy of estimated accumulative energy consumption of all participating laptops is relatively low, directly caused by our purely software based approach which significantly reduces the initial cost of the DR infrastructure.

Inhaltsangabe

Mit zunehmender Integration erneuerbarer Energiequellen (RES) steigt die Komplexität der Stromnetze. Aufgrund der schwankenden Natur der RES kann die Gewährleistung der Zuverlässigkeit von Stromnetzen schwierig sein. Aus diesem Grund haben Forscher und Praktiker im Bereich Smart Grids (SG) in den letzten Jahren Lösungen für diese Anliegen entwickelt. Ein interessantes Konzept des SG ist die Demand Response (DR): Die Änderung der Nachfrage nach elektrischer Energie basierend auf Änderungen der Versorgung. Die Implementierung eines solchen Systems, welches in der Lage ist, den Leistungsverbrauch einer großen Anzahl elektrischer Geräte aufrechtzuerhalten, führt jedoch zu einigen weiteren Komplikationen einschließlich Zuverlässigkeit, Skalierbarkeit, Sicherheit, Praktikabilität und finanzieller Machbarkeit. Um diese Probleme zu lösen, schlagen wir eine Ausarbeitung und Implementierung einer DR-Infrastruktur für Laptops vor. Unser Ansatz ist in der Lage, den Energieverbrauch zu überwachen und den Stromverbrauch von mehreren Laptops in Echtzeit zu steuern. Darüber hinaus empfehlen wir die Anwendung eines rein softwareorientierten Ansatzes zur Durchführung von DR-Aktivitäten, wodurch die anfänglichen Kosten für die Nachfrageseiten-Teilnehmer auf Null reduziert werden. Des Weiteren führen wir Experimente an einer Reihe von Laptops durch, um die Wirksamkeit und Leistungsfähigkeit unseres Designs zu messen. Wir stellen fest, dass unser System erfolgreich wirksame DR events durchführt. Allerdings ist die Genauigkeit des geschätzten akkumulativen Energieverbrauchs aller teilnehmenden Laptops ist relativ gering, verursacht durch unseren rein softwarebasierten Ansatz der die anfänglichen Kosten der DR-Infrastruktur deutlich reduziert.

Acknowledgment

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Hans-Arno Jacobsen for the very useful remarks and engagement during this thesis. Furthermore, my very exceptional thanks go out to my advisors Jose Rivera and Martin Jergler for their many great ideas, knowledge, feedback and the time they invested in my work. Also, I'm profoundly thankful to all my friends and colleagues, especially Thomas Kriechbaumer, at the Chair for Application and Middleware Systems for all their help, inputs and commitment which made this thesis possible and an unforgettable experience for me.

Last but not least, I'm genuinely thankful to my dearest parents, brothers and all my awesome friends, especially Dr. Christian Strobel, who have come along with me with their very great love, patience, and mental support.

Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current.
DLC	Direct Load Control.
DR	Demand Response.
DSM	Demand Side Management.
EMS	Energy Management System.
i13DM	I13 Demand Manager.
i13DR	I13 Demand Response.
i13DRP	I13 Demand Response Provider.
RES	Renewable Energy Sources.
SG	Smart Grid.
SUT	System Under Test.

Chapter 1

Introduction

Motivation

Global warming and climate change are among the greatest threats facing our planet. Several scientific works verify that human activities are the primary cause of the problem, including using fossil fuels for producing electrical energy. As a result, many countries around the globe are moving toward further adaptation of Renewable Energy Sources (RES) which plays a key role in reducing greenhouse gas emissions. Despite several distinct advantages of RES, it's challenging to integrate RES supplies to power grids. One of the main concerns is the fluctuating nature of RES, making already complex and dynamic power grid more vibrant[2]. The utilities are required to provide new supplies for the consumer's electrical demand once RES output decreases due to environmental changes, such as lack of the wind for turning wind turbines.

One widely adopted solution is integrating fossil fueled backup power generators when the current demand is higher than supply[3]. Although an easy fix, these backup generators are expensive to run and they are more pollutant than conventional power plants. For this reason, many practitioners and scientists are investigating approaches to reduce the demand side energy production instead of increasing the electrical power production[3, 4]. Many studies have proposed several solutions from developing more energy efficient appliances and materials to the deployment of Smart Grid (SG) for delivering the electrical energy to consumers in a controlled manner. Therefore, an infrastructure for maintaining demand side energy consumption based on SG is a practical solution.

Problem Statement

When designing an SG infrastructure, capable of compensating for intermittent reduction of electrical energy production of RES, the primary concern is maintaining the demand side power consumption in a way that it matches with currently available RES supply. The concept addressing this concern is known as Demand Response (DR).

We describe DR as changing the demand for electricity according to the changes of supply [5]. According to the description, a minimum DR infrastructure consists of a number of essential components, including power control component on the demand side for maintaining the energy consumption, a power monitoring system on the demand side to measure the demand side energy consumption as well as its contribution to the power grid by reducing the power consumption. Furthermore, it includes a facilitator component located between the demand side and electrical utilities providing the RES. The facilitator coordinates the energy consumption of demand side users according to the changes of supply.

The primary factor for deploying a robust DR infrastructure is the number of participating consumers on the demand side[4]. However, as the number of electrical appliances joining such a complex distributed infrastructure increases, several new concerns regarding scalability and response time arises. These issues are particularity significant when we expect the DR infrastructure to integrate with fluctuating RES, as real-time responses to immediate changes in supply are vital.

Furthermore, motivating demand side consumers to participate in a DR program is challenging. Hardly any electrical appliances are equipped with energy controlling and monitoring as well as communication components required for a DR infrastructure. As a result, the consumers are imposed to the financial burden of purchasing and installing these devices[3]. Moreover, limiting the electrical power consumption of consumers for a period of time may affect their comfort level and hence resulting in less motivation to join a DR program. Most of the existing DR infrastructures offer financial incentives for persuading the consumers, however, due to limited consumption of many users, such a household consumers, the financial gain is not significant enough to compensate for initial costs and convince them to take part in DR.

In summary, an effective DR infrastructure, which is designed and implemented to be integrated with RES, should be able to significantly reduce the energy consumption of a vast number of electrical appliances in real time. Moreover, it needs to be financially affordable for customers, to increase their motivations to participate in the DR program.

Approach

To address the issues explained in section 1.2, we design and implement a distributed real-time DR infrastructure for laptops. We select laptops as our target group because of their widespread usage. Furthermore, we use the computing power of laptops for implementing a purely software-oriented approach toward measuring the current energy consumption by using mathematical regression models and applying power control techniques based on built-in power management features of the operating system. Finally, we use the Internet connection to communicate with another side of DR infrastructure. As we utilize existing resources provided by traditional laptops to perform the required activities of a DR event, we significantly reduce the initial costs of demand side participants to zero cents.

Figure 1.1 summarizes our proposed design for DR infrastructure. On the left side, we present the demand manager desktop application for the laptop which we design and develop to control the demand side activities. Demand manager application encapsulates all the required features and components for performing an effective DR event, including but not limited to components for measuring the energy consumption as well as systems for restricting the power consumption. Furthermore, it includes the capabilities necessary for communicating essential DR related information with the other side of the DR infrastructure.

On the right side, we illustrate the components constructing the DR provider. It consists of two essential components, the primary DR provider, and the real-time database. DR provider is responsible for communicating with electrical utilities and RES to be informed currently available of electrical supply. Furthermore, it executes the tasks for scheduling and maintaining a DR event with cooperation with real-time database and participating laptops to instantly reduce the energy consumption. Finally, the real-time database empowers us to collect the DR related information sent by participating laptops as well as propagating the details of a DR event to them in real-time. As a result enabling us of fast responses to irregular changes of RES.

Moreover, we package and distribute the demand manager application which is accessible from `http://i13dr.de/`.

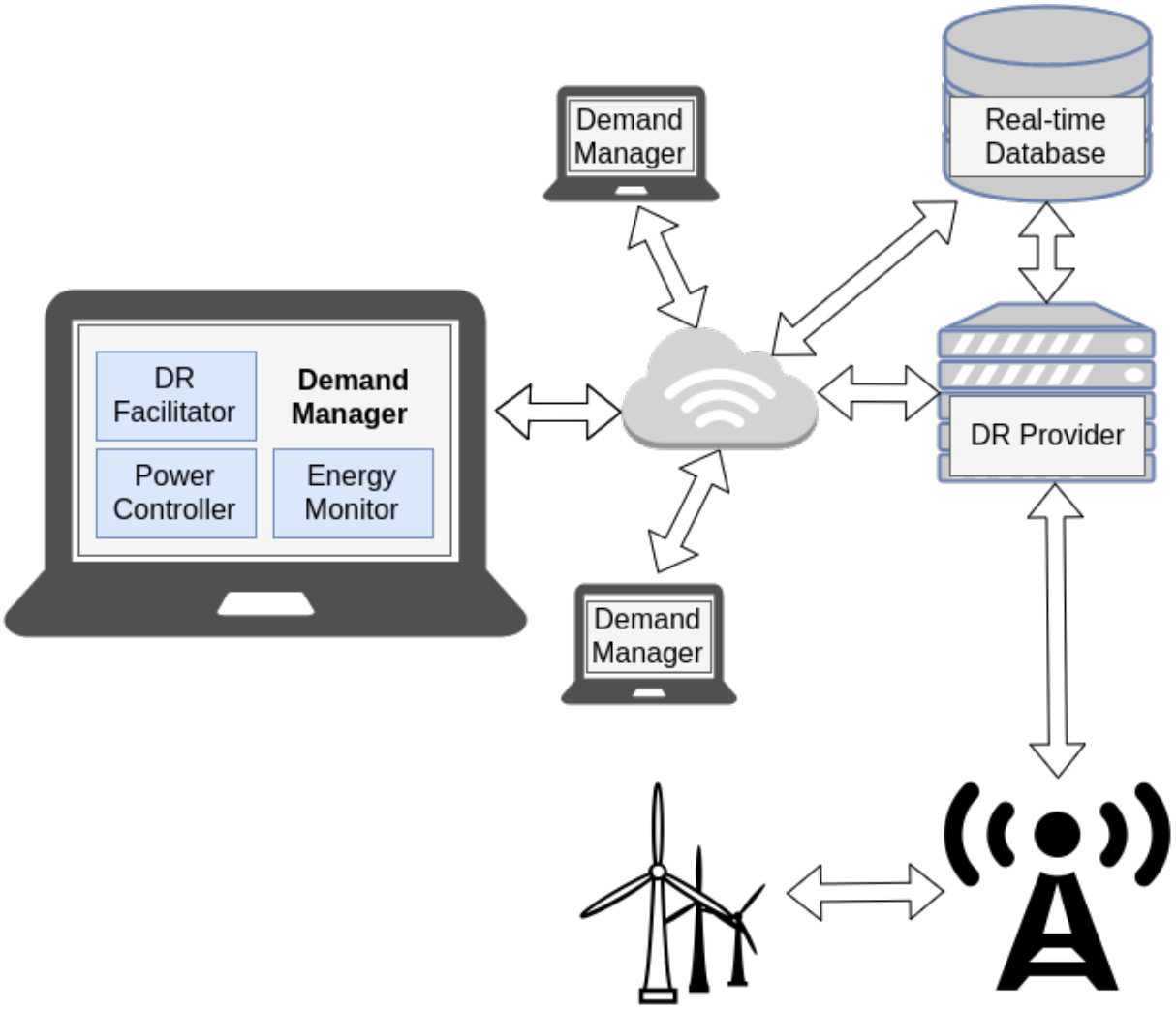


Figure 1.1: Proposed design for DR infrastructure

Contributions

We set the objectives of this thesis to address the issues expressed in 1.2, therefore, our design offers the following contributions:

- Our design significantly reduces the cost of initializing and maintaining a DR infrastructure by building upon existing tools and systems, especially costs of initialization and participations in DR events for demand side consumers. Furthermore, our design offers high usability, and it requires no to little interactions with the users, hence, causing a negligible disturbance on users. As a result, unlike other existing DR infrastructures[3, 6, 7], we increase the motivation of several participates joining our DR infrastructure.
- Several proposed DR infrastructures designs[8, 9, 10] are unable to perform real-time demand control, and utilities are required to plan the event ahead. Hence, these designs are not appropriate to be integrated with RES. However, we take real-time and low delay responses into consideration while developing and implementing our DR infrastructure from communication to DR scheduling and power consumption control systems.
- To reduce the cost of DR infrastructure, we intend to measure the actual energy consumption of laptops by employing a fitted regression model which estimates the real power consumption of a laptop and its accessories including the AC adapter, where the estimation is calculated based on a number of provided system metrics. There has been a significant amount of work on modeling the power consumption of the computers in server environment [11, 12, 13], however, little work is conducted done on small computing devices such as laptops. During this thesis, we propose an approach for constructing and deploying power consumption models for laptops.
- The majority of the academic research done on the concept of DR is based on mathematical simulations without any empirical measurements from real world experiments [14, 15]. Over the course of this thesis, we implement a production-ready DR infrastructure, and we evaluate its effectiveness and performance by conducting several experiments. Furthermore, other researchers can use our implementation as a test bed for to perform DR optimization and scheduling related inquiries.

Organization

First, we introduce the essential concepts, tools, and fundamentals for understanding this thesis in Chapter 2. After, in Chapter 3, we review the existing literature on DR infrastructures as well as power modeling. Subsequently, we analyze the realized requirements of our system and then describe our approach toward decomposing, implementing and deploying our design in Chapters 4 and 5 respectively. Afterward, in Chapter 6 we explain the conducted experiments and discuss our findings on the performance of our work. Finally, in Chapter 7, we summarize our work by reviewing our concluding remarks and offering the future works.

Chapter 2

Background

In the following chapter, we explain the concepts and technical terms which are necessary for obtaining a general insight into the thesis.

Smart Grid and Demand Response

A Smart Grid (SG) is an electrical grid equipped with smart appliances for regulating and controlling the distribution of electricity from utilities to the demand side [5]. The utilization of smart devices, such as smart metering devices, intelligent sensors and Information and Communication Technologies, empowers the smart grid to match the electricity consumption on the demand side over a period of the current supply; also known as Demand Side Management (DSM) [3]. Furthermore, DSM facilitates the necessary real-time response to an irregular availability of RES resulting in more energy efficient power grids [16]. Demand Response is one of the main activities in DSM which is the focus of this thesis.

Demand Response (DR) is officially defined as "a tariff or program established to motivate changes in electric use by end-user customers; in response to changes in the price of electricity over time, or to give incentive payments designed to induce lower electricity use at times of high market prices or when grid reliability is jeopardized" [5]. As the definition implies, DR benefits both utilities and consumers. From one side, consumers enjoy a reduction of their energy cost. From another side, decreasing the total power consumption results in a more reliable electrical grid. Therefore utilities are required to invest less on increasing the energy supplies.

Many factors need to be taken into consideration when designing and implementing an efficient Demand Response infrastructure. In the following, we discuss some of the essential core concepts. However, one should keep in mind that DR covers a vast area of technologies and ideas and we only include the ones directly related to this work.

Demand Response Management

Adjusting the electricity usage to the currently available electrical supply is achieved through executing DR events. A successful execution of a DR event requires a collaboration of following participants[17]:

- Demand side electricity consumers
- DR aggregator communicating with demand side and initiating the DR event
- Independent System Operator (ISO) or Regional Transmission Operator (RTO)

A DR aggregator initiates a DR event once the ISO/TSO informs the DR aggregator about the required reduction in demand side consumption. Afterward, the DR aggregators, according to the participants' availability and energy use profile, select the potential consumers. Next, the DR aggregator accumulates the total demand and offered the energy reduction on consumer's side and reports the result to the ISO/TSO.

During a DR event, the participating users can limit their usage through three methods. First, they can cut down on their electricity usage by reducing the power consumption their appliances resulting in load curtailment, for example dimming the light bulbs. The other option is shifting their usage to slack periods, for instance, turning on the washing machine at a different time during the day. The last option is making use of onsite generators instead of using the electricity provided by the main grid. However this option is not applicable to many participating consumers and also causes undesirable environmental issues[18].

Demand Side Consumers

To design the most efficient DR infrastructure, it's essential to take the characteristics of the demand side consumers into consideration. The electricity consumers, based on their consumption pattern and the amount of usage, can be classified into Residential and Small Commercial Consumers, Large Commercial Consumers, and Industrial Consumers. Every group of consumers has their differing and individual characteristics, but since we focus

on residential and small commercial users in our work, we investigate only this group in depth.

Residential users utilize a wide variety of household appliances with different electrical loads and patterns, contributing to the DR events by either curtailing their load or shifting their consumption to slack times. However, due to the relatively small power consumption of many households which leads to insignificant financial return from participating in DR events in many cases, many residential customers lack the motivation to pay for the initial investment for establishing the required infrastructure for DR [3]. On the other hand, many residential consumers are concerned with their comfort level while participating in the DR events, for example raising a temperature set-point of customer's HVAC for a few hours on a hot day might negatively affect their comfort levels. Furthermore, the relatively random consumption pattern by residential consumers makes it increasingly complex to model their power consumption which is required for executing successful DR events.

Demand Response Events

An effective DR event heavily depends on the number of participating consumers. As the number of participants increases so does the success rate of DR event. The majority of participants cooperate because of financial incentives, while some other participate due to general responsibly feeling toward environmental issues or preventing blackouts. On the contrary, many consumers might not be willing to join due to lack of information about the actual financial gains or amount of discomfort they experience during the event. As a result, it's vital to design the DR infrastructure and events in a way that increase the motivation of customers and addresses their concerns. According to [3, 4], a DR event, based on its characteristics, can be classified according to, first, their control mechanism, second, the incentives offered for encouraging the participants and finally based on the decision-making factors.

When categorizing DR events based on control approach, we focus on the included participants during the decision-making process of a DR event. Accordingly, we can further classify them into centralized and distributed events. During execution of a centralized DR event, every participating consumer directly communicates with the aggregators through one-way or two-way communication channels. Whereas, demand side participants of a distributed also communicate with each other in a peer-to-peer manner when executing the event. The main advantage of distributed events is increasing the scalability when the number of participants grows. From another side, equipping consumer's with enough processing powers and advanced communication infrastructure is expensive and

complicated.

As mentioned, it's crucial to offer adequate motivations to convince participants to join. [4] classifies the proposed motivations into subcategories of time-based DR and incentive-based DR. The purpose of time-based incentive is applying different electricity prices during different time periods which encourages the consumers to shift the consumption to cheaper slack times. On the other hand, the incentive-based DR offers the users some motivations, usually financial, when they agree to reduce the consumption during a peak time. Consumers usually voluntarily agree to reduce their load, however, in some cases, they can be penalized for refusing to enroll. The primary technique used during incentive-based DR events is Direct Load Control (DLC). While executing a DLC over a DR event, the aggregators or utilities can remotely control the power consumption of the consumers by turning the appliances on or off. According to [19], DLC is more effective for residential and small commercial users rather than industrial ones.

The other factor for classifying DR events is the decision variable, which further classifies them to task-scheduling DR event and energy-management-based DR events. The former type of DR events focuses mainly on the time when DR event is running. For example, they shift the charging time of plug-in electric vehicles (PEVs) from peak time to the slack time. Wheres, the energy-management-based DR events focus on reducing the power consumption of running loads, for example, by increasing the thermostat of an AC systems for a few degrees while maintaining consumer's comfort.

Smart Grid Technologies For Demand Response

Implementing a functional and efficient Demand Response Infrastructure requires a minimum set of Smart Grid technologies and devices for facilitating the communication, decision-making and propagating the DR events. [3] categorizes the necessary devices into three groups: control devices, monitoring systems, and communication systems.

As the primary purpose of DR is adjusting the consumer's load according to supply, utilizing effective power control devices are crucial. The control devices cover a broad range of tools from load control switches and smart thermostats meant for not only disconnect the load from the grid but also limit the electricity usage over a period depending on the applied DR event. Furthermore, the control devices can communicate with DR providers for receiving the DR event's information.

The other essential part of the DR infrastructure is monitoring systems, which accurately measure and report the electricity consumption to the DR providers. The reported usage

plays a key role in estimating what the DR participants can offer in the means of reduced load on the grid. The used monitoring systems can vary from smart metering devices, measuring loads of whole buildings, to Advanced Metering Infrastructure (AMI) and Energy Management Systems (EMS) which are capable of measuring and analyzing the consumption down to individual appliances level.

A reliable communication system, which connects all the participating consumers to the DR aggregators/provider as well as the utilities, is the backbone of a DR infrastructure. According to the requirements of the DR infrastructure, many one-way or two-way communication infrastructures with comparable initial costs are available. The communication systems can be built upon the telephone lines, radio wireless technologies, the Internet, etc. However, a stable communication system needs to satisfy requirements:

- Scalability to maintain a large number of participating consumers
- Flexibility toward unavoidable network failures
- Security to protect the sensitive information of the grid as well as the participating consumers
- Protection of the privacy of the participants, especially when the collected data could reveal their identity and lifestyle
- Quality of service for the used communication infrastructure to support seamless connectivity and also low latency for real-time and near real-time systems

Electrical Load Measurement

As discussed in section 2.1, the primary objective of a DR infrastructure is controlling the delivery of electrical loads to the demand side. For this reason, it's essential to measure the amount currently consumed electric power accurately.

When discussing electrical load measurement, two terms are important, first the electric energy and then the electrical power. Electrical energy is a type of energy being delivered to electrical circuits and afterward being further converted to another type of energies, such as thermal or kinetic energy. Furthermore, electrical energy, similar to any energy (E), is defined as the amount of work being executed over a period of time(T), while electrical power (P) is the rate of performing the work[12]. Equation 2.1 represent the mathematical relationship between energy, power and time.

$$E = P * T \tag{2.1}$$

In the International System of Units (SI), Joule is the unit of energy; second is the unit of time and watt is for the power. However, when measuring the electrical load consumed by electrical appliances, it's more common to use the derived unit of the kilowatt-hour (kWh) which is also applied by electrical companies when billing the customers. When measuring the amount of energy consumed over a period in kWh, the power in kilowatts must be multiplied by time in hours. For example, an electrical device which consumes power with a rate of 1000 watts per second, will consume a total energy of 3.6 megajoules over a period of one hour which is equal to 1 kWh. Since the relation between energy and power with regards to time is evident, we use the terms electrical power and energy interchangeably in this thesis, unless stated otherwise.

Active, Reactive and Apparent Power

Electrical devices such as laptops, connected to an Alternating Current (AC) load, consume the AC power in two forms: The Active Power and Reactive Power. The electronic components such as inductors and capacitors store the energy instead of transferring the energy one direction in the circuit. These elements return the stored energy to the circuit during each AC cycle which is known as Reactive Power. Whereas, the Active Power is the portion of AC power transferred in a direction in the electrical circuit and consumed for performing the desired work. The combination of active power and reactive power is known as the Apparent Power. Finally, the ratio of active power to apparent power is known as Power Factor. Measuring the amount of active power is the focus of this work because the active power is the real electrical energy consumed by appliances on the demand side. Furthermore, the measured active power is the metric used for billing the customers by electrical utilities.

The unit for active power is watts and is mathematically symbolized with a capital letter *P*. The unit for reactive power is *Volts-Amps-Reactive (VAR)* and is usually expressed with a capital letter *Q*. Finally, the unit for apparent power is *Volts-Amps (VA)* represented by capital letter *S*. Figure 2.1 is known as the Power Triangle, illustrates the relations between Active, Reactive and Apparent Power. We see that the apparent power is the magnitude of the vector sum of active and reactive power, as mathematically formulated in 2.2.

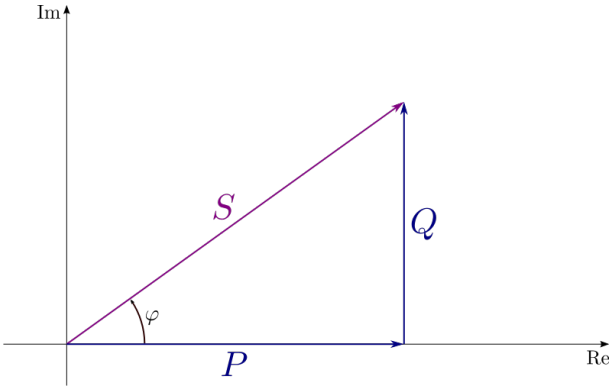


Figure 2.1: The power triangle[1]

$$S = sqrt(P^2 + Q^2) \tag{2.2}$$

Power Estimation of Laptops

Many DR infrastructures are dependent on psychical monitoring devices which consumers are required to purchase and install. However, since one of the primary objectives of this work is eliminating such a devices due to their high costs, we need a software approach for measuring the real power consumption of laptops.

It's worth to mention that the laptops' energy consumption is positively correlated with the load on their circuits. In order words, when any of the processing load, networking traffic or screen's brightness increases, so does the amount of power consumed. Furthermore, when the laptop's battery is charging, it causes and extra load on top of the regular energy consumption.

As a result, when estimating the power consumption of a laptop over a period of time, we detect and extract the system metrics which highly correlate with the energy consumption. Afterward, we feed the derived parameters into a previously built mathematical regression models to calculate the total power consumption. Equation 2.3 shows the mathematical relationship between estimated energy consumption $E_{estimated}$ and the energy model $f()$, where $m_1(t)$ to $m_n(t)$ are the extracted system metrics over a period of time and the estimated energy is the consumed energy over the same period of time.

$$E_{estimated} = f(m_1(t), ...m_n(t)) \tag{2.3}$$

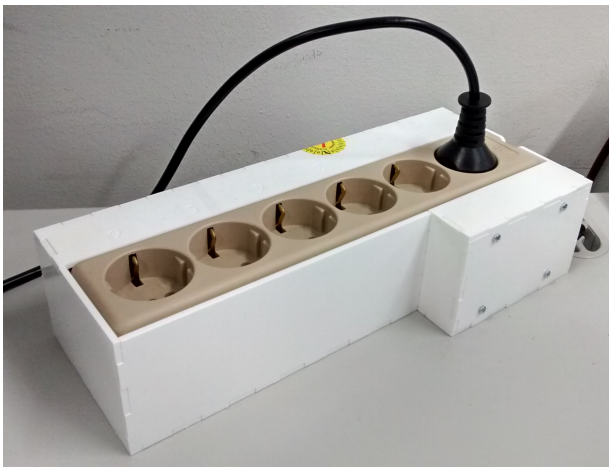


Figure 2.2: The MEDAL high-frequency power meter

The MEDAL High-Frequency Power Meter

Over the course of this thesis, we need to monitor and acquire real power consumption of several laptops to construct power consumption models as well as evaluating the performance of our design. For this purpose, we use an energy metering device called *MEDAL*[20]. MEDAL is a custom built low-cost measurement system for high-frequency energy data built upon a voltage-sensing circuit, current sensors, and a single-board PC as a data aggregator. It can achieve up to 50 kHz sampling rate for the real power, apparent power, and power factor. Figure 2.2 shows the MEDAL device an AC adapter of a laptop connected to one of its sockets.

Chapter 3

Related Work

As discussed in previous chapters, over the course of this thesis we focus on two areas of DR and energy modeling. Therefore, we investigate the works done on these concepts. During the first section of this chapter, we review the existing studies on different DR infrastructures and in the second part we discuss the various approaches for modeling energy consumption on electrical appliances.

Demand Response Infrastructures

In recent years, researchers have studied many aspects of DR infrastructures in depth[3, 4], and several different approaches to DR designs and events have been proposed. However, the majority of works focuses on the isolated areas of DR and do not provide a design for a production ready real-world DR infrastructure. Therefore, we are offering a plan for a practical and complete DR infrastructure in this work.

Laptops as Smart Participants

Residential and small commercial consumers account for about 40 percent of total electricity consumption in Germany[21]. According to [4], the contribution of this kind of users to DR is through load curtailment or shifting. However, to increase their motivation to join the DR events and consequently increasing the effectiveness of the DR infrastructure, their concerns must be taken into consideration. The primary concern of these consumers is first, the initial costs of installing the devices needed for enabling DR infrastructure and second, the experienced discomfort during the DR events.

Despite many obvious benefits of DR, initializing and enabling a DR infrastructure is financially expensive, and DR providers and demand side consumers are obligated to pay[22]. The demand side participants have the financial burden of enabling SG technologies such as load controllers and energy management systems [23]. Since the participants are often left alone to pay for them, it's a significant discouragement for many customers especially the ones with limited consumption and consequently negligible financial gains from DR events [4]. Furthermore, the DR Providers are required to pay for operational costs varying from costs of monitoring and communication systems to administrative and consumer's educative instructional costs. As a result, in our work, we emphasize on reducing the cost of DR infrastructure while maintaining the requirements of a successful and efficient DR infrastructure. To keep the costs as minimal as possible, we make use of existing tools and devices for enabling the required functionalities of DR providers and demand side consumers. For this reason, we target laptops as demand side users to make advantage of their resources.

Several studies have been done on DR designs focusing on residential and small commercial consumers [24, 25]. However, most of the works emphasize on different financial incentives and electricity pricing schemes to encourage the users' participation. Studies indicate that schemes such as Time-Of-Use (TOU) [24, 26], Critical Peak Pricing (CPP) [27, 25] and Real-Time-Pricing (RTP) [28, 29] encourage several consumers by decreasing their monthly electrical bills. However, they put a little effort on minimizing the initial costs.

We select laptops as our target consumers because of their widespread usage in everyday life in residential, educational and commercial environments. In 2016 about 68.5 percentage of German households owned laptops [30] which we can use for performing several activities of a DR event. As an example, Operating Systems (OS) offer power management features for limiting their energy consumption as well as computational powers for measuring the real energy consumption employing mathematical regression models. Moreover, the majority of laptops have constant access to the Internet which we use a communication system for a DR infrastructure. In conclusion, laptops make an excellent choice as targets for DR participants when decreasing costs is a design goal.

[31] conduct two separate studies on DR event simulations based on DLC for laptops. For the first part, they simulate a classic DR programs by controlling the consumption load of a vast number of laptops over a period, and they observe load curtailment in a range of 30 to 90 percents for average baseline load. For the second study, they simulate Continuous Demand Response event by integrating intermittent RES as power supplies, and they report a reduction of grid dependency in the range of 26.8 to 33.8 percents. Despite the relatively good results, their proposed DR approach builds upon on sensing/actuating

wireless mesh networks for controlling the energy, and unlike our work, they do not make use of built-in features of laptops. Furthermore, they only run simulations, and their power consumption models are derived from historical data collected from devices. In contrast, our work is based on real-world implementation and data collection.

Real-time Responses to Intermittent RES

RES account for 28.2 percents of gross electricity consumption in Germany, and it's been steadily increasing over the past two decades [21]. Despite several obvious benefits of incorporating RES, such as sustainability and leaving behind significantly less carbon footprint, high integration of RES to the power raises many concerns with irregularity, uncertainty, and unreliability of supplies [2, 32]. The primary cause of these issues is fluctuating nature of RES sources, which are commonly the wind and the sun. One solution for increasing the predictability of the RES sources is weather forecast. However, it's not without error [33]. The typical solution for compensating the lack of RES source is using on-site generators or emergency generators[32]. Nevertheless, these solutions are financially expensive and they face many governmental restrictive regulations in many countries [3]. As a result, many researchers and practitioners consider DR as an alternative solution for addressing the issues of integrating RES.

Exploiting DR for maximizing RES benefits has shown many promising results. However, it comes with various challenges. Beside the mentioned uncertainties of RES output, the unpredictability of the power consumption on the demand side increases the complexity of the system. [7] optimizes integration of RES to the grid by employing an EMS which aggregates the distributed energy resources and offers them to demand side participants in an open market with real-time pricing. Moreover, their simulation results indicate that real-time pricing yields good result in matching the demand levels with supply. However, in our work instead of real-time pricing, we practice real-time DLC. Furthermore, we emphasize on implementing real-time and low delay response for all aspects of our design.

When emphasizing on real-time interaction and communications, scalability becomes a serious issue as the number of demand side participants increases. To address this problem, [34] proposes an approach for Electric Vehicle (EV) charging control framework by utilizing a pub/sub middleware system to optimize the EV charging. In their work, they aggregate the battery charging demand profile of several EVs, for matching the demand with supply using valley feeling technique. They also illustrate that a major bottleneck is aggregating collected data of charging profiles. Moreover, they run simulations and evaluation for three different aggregation approaches: centralized, decentralized and

decentralized with in-network aggregation. Moreover, they observe that a decentralized with in-network aggregation approach yields a better result with less computational overhead in comparison with other methods. Similar to their work, we require to monitor and aggregate the power consumption of laptops in real-time. As a result, our aggregation approach is similar to their technique with some differences which we discuss in depth in chapter 5.

Real-world Implementation

As discussed in the previous sections, several researcher and practitioners have proposed many DR approaches for residential and small commercial consumers[15, 35, 14]. [15] introduces a DR scheduling approach for household electrical appliances employing genetic algorithms optimization. They report a significant electricity cost reduction and peak-to-demand ratio matching through simulating different electricity pricing schemes. Besides, [35] investigates a DR event for shifting energy loads of residential consumers, allowing a better integration of RES. Their simulation on economic models reveals a moderate reduction in the electrical cost of customers as well as load reduction during peak time.

Both of the discussed studies only perform mathematical simulations and modelings. Although they indicate promising results of utilizing DR, they do little work on implementing a real-world proof of concept. On the contrary, during this thesis, we emphasize on implementing and evaluations a production ready DR infrastructure.

Software-based Energy Modeling

Reduction of initial financial costs of the DR infrastructure is one of our first-order design constraints. Over previous chapters, we explained that one of the expensive components is energy monitoring devices on demand side. Therefore, to reduce the costs of power auditing systems, we make use of built-in features of OS as well as power consumption models to estimate the real power consumption of laptops. Several researchers have explored the energy consumption models for server environments and data centers[12, 36, 13, 37]. Nevertheless, studies on constructing energy models for laptops and battery operated electrical appliances are relatively limited[38]. The reason for the lack of interest in this area is the significantly larger amount of energy consumed by data centers in compare to small personal mobile devices. As a result, we develop an approach

for modeling power consumption on laptops according to the existing studies on energy modeling of computers in the server environment.

According to [36] an appropriate mathematical model for estimating power consumption needs to satisfy several requirements. These constraints include the accuracy of the energy estimation, the speed of prediction, the generality of the model when applying to various systems with different hardware and software specifications, affordable non-intrusive measuring equipment and last but not least the simplicity of the design.

Furthermore, [36] classifies real-time power modeling approaches into two groups, first, detailed analytical power modeling and second, high-level black-box modeling. The former modeling exploits the CPU performance counter for accurately estimating the energy. However it's only applicable one particular processor, and because of its reliance on the micro-architectural of the processor, it's not portable from one system to other. The other modeling technique takes advantage of system metrics, such as CPU, disk and memory utilization for constructing linear or multiple regression energy estimation models. This type of modeling is less accurate than the first one; however, it's more general and portable due independence from system specifications. Furthermore, [36] constructs and evaluate five different energy consumption models by means of Mantis [39], a power modeling non-intrusive system. Mantis uses a one-time model fitting, during which system utilization metrics are fitted to power readings of an external AC power meter. Evaluations of the generated power models indicate that the utilization-based regression models have better performance in comparison to other models. In our work, we follow the same approach for creating linear regression energy consumption models for laptops based on system utilization counters.

All the previously mentioned studies fit and generate energy models using external assistance like AC power meters or a second computer. In contrast [38] introduces an approach for self-constructing energy modeling system for laptops and mobile devices called *Sesame*. It exploits the smart battery interface of laptops for self-power measurement. Despite the accurately estimated power, their energy models only measure the power consumption of laptops without taking the AC battery charger into consideration. In our work measuring the amount of energy drained by laptops through the battery charger is essential. Therefore, we still require making use of external power meters for collecting data needed for fitting the energy models.

In summary, according to different approaches for constructing power models, the workflow for generating power models for laptops consists of three phases:

- *System Utilization Metrics Extraction*: We record the OS utilization metrics of the

system while we read the real power consumption of system from a connected AC adapter.

- *Fitting and Evaluating the Power Model*: By using different regression and Machine Learning techniques the extracted metrics are correlated with the recorded power consumption readings to create a mathematical equation for estimation the real energy consumption.
- *Energy Estimation Model Deployment*: We deploy the generated model on the different machines to be used for power consumption prediction.

Chapter 4

Requirements Analysis

In the following chapter, we elicit the requirements of the DR infrastructure. In section 5.1 we provide an overview of the purpose, objectives and success criteria of our design. After, in section 4.2 we explain the functional and nonfunctional requirements that our system must satisfy. Finally, in section 4.3 we describe the features and functionalities of the system through scenarios, use cases, etc.

Overview

This thesis aims to design and implement a real-time and distributed DR infrastructure for laptop devices with reduction of initial costs as a first-order design constraint. In other words, we focus on creating low delay DR infrastructure with direct load control on laptops in response to immediate changes in the electrical supply of RES, in a way that the curtailed energy consumption on the demand side matches the available electrical supply. Moreover, we make use of several built-in functionalities and features of the OS of laptops to dramatically decrease the initial cost of participants of DR events to zero cents. For simplicity, we refer to our designed DR infrastructure as *i13DR* from now on. The i13 part in the i13DR refers to Informatics 13 - the Chair for Application and Middleware Systems at Technical University of Munich's Department of Informatics, where this thesis is realized and developed, and DR refers to Demand Response.

i13DR is a DR infrastructure that facilitates direct control of power consumption on laptops to match with the available supply provided by RES. We design i13DR according to minimalistic requirements of a DR infrastructure stated in chapter 2. Consequently, our model consists of two parts, one desktop application running on demand side on

laptops and multiple web applications residing on our servers acting as demand response providers, aggregating demand side participants, etc. For this reason and the sake of clarity, we name the bundle of i13DR web applications as *i13 Demand Response Provider (i13DRP)* and the i13DR desktop application as *i13 Demand Manager (i13DM)*.

i13DR Requirements

In the following subsections, we describe the functional and nonfunctional requirements of i13DR. A functional requirement represents the interaction of i13DR with users or any other external systems, independent of i13DR implementation. In contrast, nonfunctional requirements define the functionalities which are not directly related to i13DR's interactions but apply to various aspects of i13DR from performance to reliability [40].

Functional Requirements

In this subsection, we present the functional requirements of i13DR. To make it more comprehensible, we separate the lists of requirements of i13DM and i13DRP. We realize the following functional requirements for i13DRP:

- When scheduling for a DR event, the i13DRP creates and publishes the DR schedules to one or multiple i13DM, during which the i13DM activates the power control mode. i13DRP creates the DR schedule based on one of the defined formats of the weekly, daily or one-time event.
- i13DRP receives and maintains a power consumption profile for each laptop. i13DM creates the profile, and it's a weekly profile containing a record for every minute of the day holding the average energy consumption of the laptop in normal mode and power save mode and the probability that the laptop is connected to electricity grid through AC adaptor.
- i13DRP receives and maintains a location profile for each laptop. i13DM creates the profile, and it's a weekly profile containing a record for every minute of the day with the longitude and longitude, zip code as well as the accuracy of the found location.
- The administrator of the i13DRP uploads and publishes the power model required for estimating the power consumption by i13DM.

- The administrator of i13DRP has access to the power consumption and location profile of all participating laptops.
- i13DRP shows the most recent geographical location of laptops to the administrator as well as their online status.
- i13DRP provides the administrator with a list of all participating laptops. The administrator can filter the laptops based on their location and their unique identification number.
- i13DRP receives the currently available supply of electrical energy of RES in real-time and creates DR schedules for participating i13DM based on their power consumption and location profiles.
- The administrator of the i13DRP can view or download the system metric readings gathered and provided by i13DM.
- The administrator of i13DRP have the option to remove the participating from the system. Furthermore, he/she can observe the reported crashed and diagnostic logs of i13DM.
- The administrator of i13DRP remotely changes the settings of one individual or a group of selected i13DM, including, enabling and disabling logging of the laptops' system metrics, restarting the i13DM, deleting the locally stored data and modifying the update interval of power profiling.

We elicit the following functional requirements for i13DM:

- i13DM creates a weekly location profile on the first start up and publishes it to i13DRP. Furthermore, the profile is periodically updated by i13DRP. The profile represents the most probable location of the laptop at any given time during the day. The location data consists of the latitude and longitude, the accuracy of located location estimate as well as the zip code.
- i13DM creates a weekly power consumption profile on the first start up and publishes it to the i13DRP. Also, i13DM updates the power profile periodically. The profile holds an estimate of laptop's energy consumption rate in normal mode and power save mode and the probability of laptop being connected to the electricity grid through the battery charger. i13DM measures the power consumption according to the provided power models.
- To motivate the users to participate, i13DM displays the statistics of DR events including the number of minutes the laptops attended, the amount of energy saved

in kWh during the execution of DR event as well as the amount of saved money in Euro.

- i13DM provides the user with the option to allow or disallow the laptops participating in the DR event. furthermore, users can limit the time during which laptops can attend a DR event to certain periods of the day.
- Once i13DRP publishes a schedule of a DR event, i13DM fetches the plan and activates the power control accordingly.
- Once i13DRP releases a new power model, i13DM downloads and updates its local energy model.
- i13DM automatically downloads and installs new versions of the i13DM desktop application.
- i13DM pushes informative notifications to the users before executing activities that might affect the normal behavior of laptops. These actions include but not limited to installing a new version of the application, activating and deactivating the power control mode during a DR event.
- In the case of i13DM crashes and system failures, it reports the crash logs and stack traces to i13DRP.
- i13DM automatically launches at the system start up. However, this feature is controllable by laptops' user.
- i13DM continuously runs in the background and user can control the behavior of the i13DM though its tray icon presented in the taskbar.
- Upon the decision of i13DRP's administrator, i13DM logs and reports system utilization metrics to i13DRP. The readings include but not limited to energy consumption rate, voltage and capacity of the battery, remaining capacity of the battery, charging and discharging rate of the battery, screen's brightness, CPU, memory, hard disk and network utilization and read/write rate. Afterward, i13DM send them to i13DRP in a formatted version.
- i13DM reports its online status and availability to i13DRP.
- i13DM should control the power consumption of laptops by using built-in power management features of the OS to put them to power save mode during execution of a DR event.
- At the initial start-up of i13DM, it analyzes the laptop's hardware specification,

including but not limited to the architecture of CPU, memory, storage, video card, networking, peripheral, etc. Later, i13DM submits the collected data to the i13DRP. Moreover, i13DM extracts detailed information on the specification of the battery consisting of battery's capacity, voltage, manufacturer, etc.

Nonfunctional Requirements

Similar to the previous subsection, we separate the nonfunctional requirements of i13DR into two lists, one for i13DR and the other for i13DRP.

First, we name the realized nonfunctional requirements of i13DRP:

- Since i13DRP consists of several web applications to perform the required functionalities; it's necessary to make use of standard APIs and protocols to facilitate the communication among components.
- When any of the i13DRP applications fail, they should recover automatically and with a small delay.
- i13DRP should scale up according to the number of participating laptops for maintaining the reliability and real-time responses.
- i13DRP should only acquire and manage the data required for performing a successful and effective DR program. It's not allowed to obtain any additional data which violates user's privacy or reveals their identity.

Finally, we list the realized nonfunctional requirements for i13DM:

- i13DM must require the minimum interaction with the human users. It should run in the background and perform all necessary tasks independently on its own.
- A robust DevOps infrastructure is responsible for building, testing, packaging, releasing, configuring and monitoring the i13DM.
- At any given time only one instance of i13DR must be running on one single laptop.
- i13DM should support real-time bidirectional communication with i13DRP.
- When i13DM is running, it should cause negligible overhead on the laptop without noticeable effect on the laptop's performance.
- All the system failures exceptions caused by i13DM should be handled gracefully without distracting the user.

- Participating users should not bear any financial burden when installing i13DM and participate in a DR events.

i13DR System Models

In the following section, we explain the i13DR infrastructure using abstract models to help us analyze and understand the functionalities and requirements of the system. First, we describe a few scenarios we find most relevant. Afterward, we model the most important use cases of the system, and finally, we explain the analysis object model and dynamic models of i13DR.

Main Scenarios

A scenario is "a narrative description of what people do and experience as they try to make use of computer systems and applications" [41]. In other words, it's a detailed and informal description of a feature of the infrastructure from the viewpoint of a user helping to extract and understand the requirements of the system better [40]. Here we describe three main scenarios of i13DR presenting the most important features

The first scenario demonstrated in table 4.1 represents the i13DR response to irregular changes of RES. As the production decreases, i13DR creates and releases DR schedules for reducing the accumulative power demands of laptops.

Table 4.2 illustrates the scenario, during which i13DM launches and monitors the laptop's resources and power consumption.

Finally, in the third scenario presented in table 4.3, we describe the first startup workflow of i13DM.

Use Case Model

In this part, we discuss the use cases of i13DR using UML use case diagram. A use case describes a feature or a functionality of the system which the user of the system comprehends. Furthermore, we take advantage of use case diagram to aggregate all the features from the actors' point of the view[40]. For illustrating the diagrams, we use the Unified Modeling Language (UML) which is the standard modeling language for designing a system in the field of software engineering[42].

Scenario name	*RES Supply Reduction*
Participating actor instances	*Wind turbines as RES* *Multiple laptops as DR participants*
The flow of events	1. A 1 MW wind turbine steadily produces electricity at its 50 percent capacity. After a while, due to a drop of wind, its production falls off to 45 percent of the capacity resulting in 100 kW loss in production. The electrical utility immediately notifies the i13DRP of this loss through. 2. i13DRP aggregates the existing power consumption profiles and location profiles of the registered laptops which are within the operational area of the wind turbine. Accordingly, the i13DRP performs optimization tasks to generate a DR schedule for the candidate laptops to compensate for the 100 kW loss of wind turbine. Once the potential participating laptops are selected, i13DRP immediately publishes the DR schedule. 3. Once i13DM fetched the new DR plan, they immediately download and update their internal schedule accordingly. 4. i13DM regularly checks its local DR schedule to activate the power control mechanism at the correct time. Finally, when the DR program is over, it deactivates the power control and sets back the laptop to normal power consumption mode.

Table 4.1: Scenario 1: i13DR response to irregular RES changes

Figure 4.1 illustrates the use cases we realize for i13DR. Typically an actor is described as any entity interacting with the system including a user, another system or system's physical environment [40]. However, to enhance the understandability of our diagram and because i13DR internally consists of two separate parts of i13DM and i13DRP interacting with one another, we recognize both i13DM and i13DRP as the studied systems as well as actors. As represented in figure 4.1, we have three external actors, on the left, we have the DR participant who directly interacts with i13DM. On the left, there is administrator of i13DRP and electrical utilities interacting with i13DRP, and finally, in the middle,

Scenario name	*Cycle of i13DM activites*
Participating actor instances	*OS as the launcher of i13DM*
The flow of events	1. OS launches the i13DM on system start up. 2. After a full launch, i13DM queries i13DPR to retrieve any recent power models, application settings and DR schedules as well checking for new version of the i13DM application. 3. Meanwhile, i13DR monitors the resources of laptops periodically and update the power consumption profile by utilizing power model and extracting system metrics. Moreover, it repeatedly locates the laptop's location using locations APIs and updates the location profile. Once i13DM successfully updated the local power consumption profile and location profile, it publishes them i13DRP.

Table 4.2: Scenario 2: Cycle of i13DM activities

i13DRP and i13DM communicate with each other for initiating and performing the DR related activities.

Since we already describe several functionalities and behaviors of i13DR in section 4.2 and 4.3.1, we believe most of the mentioned use cases in the diagram are clear for the reader. However, we describe five essential use cases in detail because of their complexity. First, table 4.4 explains the use case for executing one DR event initiated by the electrical utility. Then, table 6.6 illustrates the initial procedure of analyzing system specification and generating the first location and power consumption profiles and tables 4.6 and 4.7 describes the use cases of updating location profile and power consumption profile respectively. Finally, table 4.8 shows the use case of the i13DM executing one cycle for updating running profile which is extended by use case of power profiling.

i13DR Analysis Object Model

In the following subsection, we use class diagrams for describing the structure of i13DR through classes and objects. Classes are abstractions of the behavior and attributes of the system and objects are entities that encapsulate behavior and state of components

Scenario name	*Initial Startup Workflow*
Participating actor instances	*User launching the laptop*
The flow of events	1. The user decides to join the i13DR infrastructure. First, he downloads the i13DM compatible with his OS from i13DR homepage and installs the application. 2. Afterward, he launches the application for the firs time. i13DM opens a window and thanks him for joining the i13DR. In the background, first, i13DM generates a unique random anonymous ID of the machine and sends it to i13DRP to be registered. Then, it sets up the required initial settings. Then, i13DM downloads the deployed power model and initialize it for energy estimation and power profiling. Later on, i13DM extracts the hardware and software specifications of the laptop, including but not limited to CPU architecture, memory and storage specification, networking, battery capacity and voltage, etc. Finally, i13DM creates the first weekly power consumption profile for the laptop by using the power model as well as the weekly location profile and submits them to i13DPR. 3. The user is not required to do any further tasks. He closes the welcome window and i13DM runs in the background and communicate with i13DRP as required. Furthermore, by default, i13DR launches on system start up automatically without causing the user any distraction.

Table 4.3: Scenario 3: Initial startup workflow of i13DM

[40]. For keeping the class diagrams precise but yet rather informative, the figure only represents the most valuable objects, relations, and attributes. Furthermore, we model i13DM and i13DRP by eliminating actors and their roles.

Figure 4.2 illustrates a UML diagram representing the system model of use cases mentioned in the previous subsection. On the left side of the figure, we present the i13DR classes, and on the right side, we include the classes for i13DRP. The i13DM classes characterize the system required for power and location profiling as well as performing DR events. *LocationProfile* and *PowerConsumptionProfile* inherit the *Profile* class holding

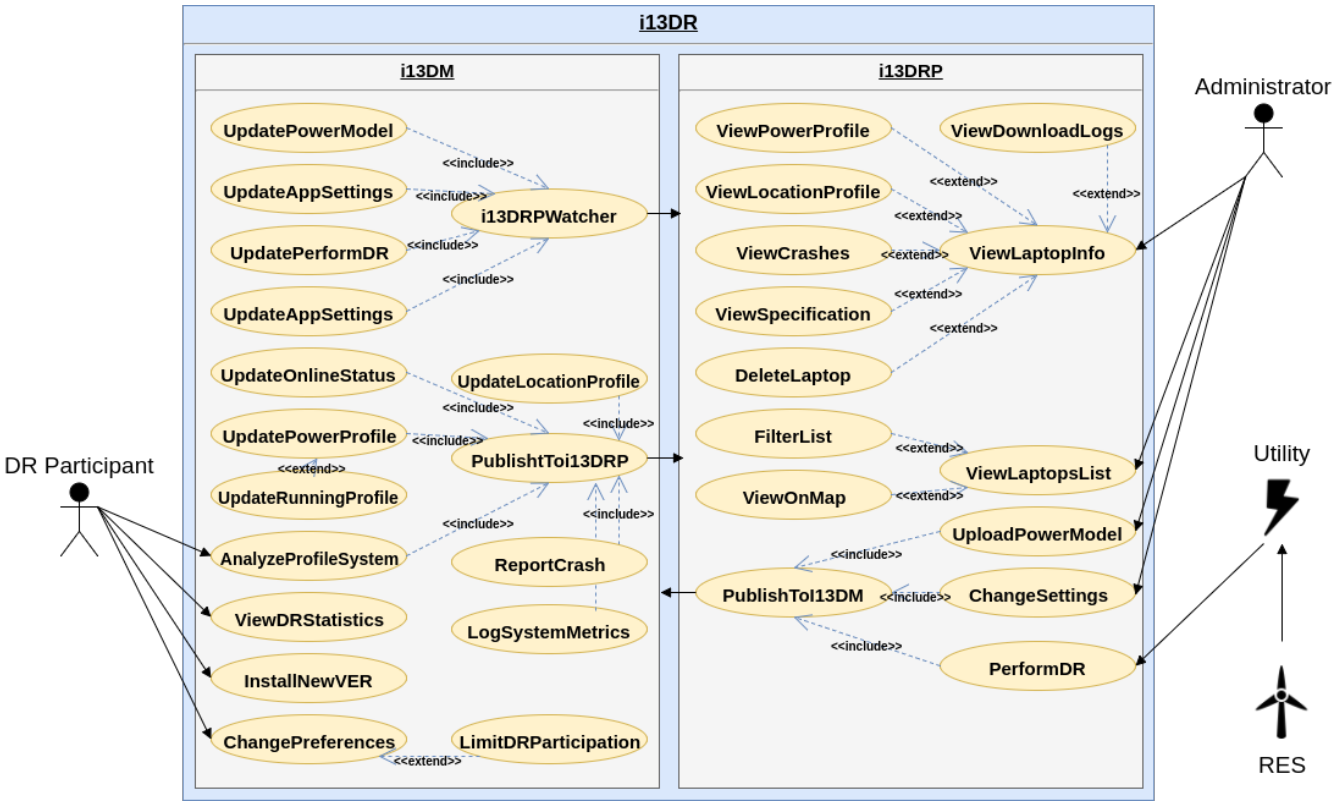


Figure 4.1: Use case diagram of i13DR

all the necessary attributes and methods for creating, storing and updating profiles. *PublisherSubscriber* class is responsible for publishing the profiles. *MonitorResources* in association with *Locator* and *PowerModeler* initiates the activities for profiling and monitoring resources. Furthermore, *Watcher* subscribes to *PublisherSubscriber* for receiving power models and DR schedules and *PowerController* is responsible for activating and deactivating the power controls according to the schedules. From another side, *i13DMProfiling* subscribes to *PublisherSubscriber* for obtaining the power consumption and location profile from i13DM and *Scheduler* is responsible for scheduling a DR event with a specific start and end time.

i13DR Dynamic Model

In the last subsection, we use state machines to illustrate the internal behavior of i13DR as a whole. The state machine diagram represents the dynamic behavior of the system through states and the transitions between them. The states are a specific set of values for an object and transitions are the conditions which need to be satisfied to move from one

Use case name	*Perform DR Event*
Participating actors	*Initiated by Utility* *Communicates with i13DM and Administrator*
The flow of events	1. Electrical utility informs the i13DRP about X kWh reduction of supply. 2. i13DRP selects the candidate participating laptops according to existing power profiles and location profiles of all registered laptops and by the help of an optimization and scheduling algorithms in a way that amount of energy saved by selected laptops matches the X kWh of loss. Once i13DRP picks the participating laptops, it creates a DR schedule containing the start time and duration of the DR event, during which laptops should activate the power control. Finally, i13DRP submits the schedule to the real-time database. 3. Once the new DR plans are available, the subscribed i13DMs fetch the program from i13DRP and update their local schedule accordingly and activate the power control at the appropriate time.
Entry condition	• The aggregated energy consumption and location profiles of laptops are available.
Exit condition	• An adequate number of laptops are participating in matching the X kWh reduction of supply with curtailed demands.
Quality requirements	• DR schedules are submitted no later than 1 second after utility's notification. • i13DM is informed of new DR schedule in near real-time.

Table 4.4: Use case 1: Performing one DR event

state to the other[40]. We believe that the majority of primary requirements of i13DR have been realized so far, for this reason, and to avoid stating the obvious we only demonstrate the state diagram for performing a DR event.

Figure 4.3 shows the UML state machine diagram of a DR event. In the beginning, i13DRP assumes that the demand matches with the supply, and it continues watching the current supply. Once RES supply goes down and the system requires to cut back on

Use case name	*Initial Profiling and Analysis of Laptop*
Participating actors	*Initiated by laptop's user* *Communicates with i13DRP*
The flow of events	1. The user launches the i13DM for the first time after downloading and installation. 2. i13DM realizes it's the initial setup of the application. Therefore, it executes several platform specific commands to extract the hardware and software specifications including but not limited to the motherboard, BIOS, cache memory, disk drives, operating system, physical memory, peripherals, video card, processors, sound devices, display and battery specifications. Once the analysis is over, i13DM submits them to i13DRP in properly formatted style. 3. i13DRP receives the system specifications and stores it in the database under participant's profile. 4. i13DM downloads the power model from i13DRP containing a mathematical equation of a regression model for calculating power consumption of the laptop in normal mode and power save mode. Afterward, i13DM acquires the required system metrics for the power model including but not limited to CPU, disk, and memory utilization and inserts them to power model for having an estimate of laptop's power consumption. Then, i13DM initializes an object for every minute of the day and every day of the week from Monday to Sunday and pushes them to a list. After, for every object in the list, i13DM set the normal mode and power save mode power consumptions according to the measured values. Also, i13DM fixes the probability of the laptop running, the i13DM application running and the laptop is connected to AC adapter to 50 percents. Once all the objects in the list are initialized with proper values, first, i13DM stores the list in the local database and then submits them to the i13DRP. 5. i13DRP receives and stores the initial power consumption profile under the particpatns profile.

Table 4.5: Use case 2: Initial analysis and profiling of laptop by i13DM

	6. i13DM uses geolocation APIs to locate the current position of the laptop. Then, it initializes an object for every quarter of an hour of the day and every day of the week from Monday to Sunday and pushes them to a list. Afterward, for every object in the list, i13DM sets the longitude, latitude, accuracy of measurement and the zip code of the location. Furthermore, it fixes the probability of the laptop being presented in this area to 100 percent. Finally, once the location profiling is over, it stores the list in the local database and submits it to i13DRP. 7. i13DRP receives and stores the initial location profile for the corresponding i13DM laptop. 8. i13DM finishes initial system analysis and profiling and keeps running continuously in the background.
Entry condition	• It's the initial startup of i13DM.
Exit condition	• The analysis of system specifications are submitted to and stored at i13DRP. • The initial power consumption and location profiles are created and sent to i13DPR.
Quality requirements	• The overhead of system analysis is negligible on performance of the laptop and users do not notice any adverse effect on usage.

Table 4.5: Use case 2: Initial analysis and profiling of laptop by i13DM

the consumption, it goes to optimization and scheduling state and publishes the schedules for i13DM. On the other side, i13DM continuously watches for new DR schedules. Once there is a new timetable available, it updates the local plan. Then, it activates the power control when DR event starts and subsequently deactivates the power control when it finishes.

Use case name	*Updating Location Profile*
Participating actors	*Initiated by i13DM* *Communicates with i13DRP*
The flow of events	1. Every 90 seconds i13DM locates the current geographical location of the laptop, and it inserts the determined longitude, latitude, accuracy of measurements and zip code to its local database. 2. After insertion of the new location, i13DM queries the previously recorded sites with the same recorded time of the day and day of the week of the latest recorded location and groups them by their zip code. Afterward, it counts the number of rows for every zip code and selects the longitude and latitude of the group with the highest count. Then, it divides the maximum count by the total number of retrieved records and labels them as the probability of the laptop to be presented this location. Finally, it updates the corresponding location profile in the local database. Once the profile is updated, i13DM submits the new location profile to i13DPR. 3. i13DRP receives the new location profile and update its version accordingly.
Entry condition	• i13DM already executed the initial system analysis and profiling. • i13DM have access to the Internet.
Exit condition	• i13DM stored the new location profile in the local database. • The new location profile is sent and stored at i13DRP.
Quality requirements	• Updating the location profile has minimal adverse effect on laptop's usage.

Table 4.6: Use case 3: Updating location profile by i13DM

Use case name	*Updating Power Profile*
Participating actors	*Initiated by i13DM* *Communicates with i13DRP*
The flow of events	1. Every few seconds (dependent on the OS), i13DM measures the current power consumption in normal mode and power save mode using the power model and system utilization metrics. Moreover, it checks that if the laptop is connected to AC adapter. Finally, it inserts the readings to the local database. 2. After insertion, i13DM queries the previously stored power consumption records to find the rows with equal time of the day and day of the week with the last recording and groups them by the status of the AC adapter connectivity. After, it calculates the arithmetical mean of power consumption in normal mode and power save mode for all retrieved records in each group and divides the number of rows in each group by the number of all fetched rows and labels it the probability of being connected to AC adapter. Finally, it updates the corresponding record in the local database with new measurements and sends it to i13DRP. 3. During power profiling, this use case extends the use case of *Updating the Running Profile*. 4. i13DRP receives the updated energy consumption profile and updates its version.
Entry condition	• i13DM already executed the initial system analysis and profiling. • i13DM have access to the Internet.
Exit condition	• The new power consumption profile is stored in local database. • The new power consumption profile is sent and stored at i13DRP.
Quality requirements	• The overhead of updating power consumption profile is negligible on laptop's performance.

Table 4.7: Use case 4: Updating power profile by i13DM

Use case name	*Updating Running Profile*
Participating actors	*Initiated by i13DM* *Communicates with i13DRP*
The flow of events	1. Every 90 seconds, i13DM inserts a record to the database indicating that i13DM, as well as the laptops, is running. Afterward, it checks if the most recent record before the last record is older than 90 seconds. If it is the case, then it inserts several new rows to database setting the status of i13DM and the laptop as not running. The timestamps of the new records begin from the timestamp of the most recent record before the last record, and it adds up 90 seconds until it reaches the last inserted record. 2. After insertions, i13DM fetches and groups the previously stored running profiles for every minute of the day and every day of the week. Then, for every group, it divides the number of records with stored i13DM running status as *true* to a number of all rows in that group and labels them as the probability of i13DM running. Moreover, with the similar approach, it calculates the probability of the computer running during that time. Finally, it updates the corresponding power consumption profiles in the local database and submits them to i13DRP. 3. i13DRP receives the updated energy consumption profile and stores it accordingly.
Entry condition	• i13DM have access to Internet.
Exit condition	• The new power consumption profile is stored in local database. • The new power consumption profile is sent and stored at i13DRP.
Quality requirements	• The overhead of updating power consumption profile is negligible on laptop's performance.

Table 4.8: Use case 5: Monitoring and updating the running profile of i13DM

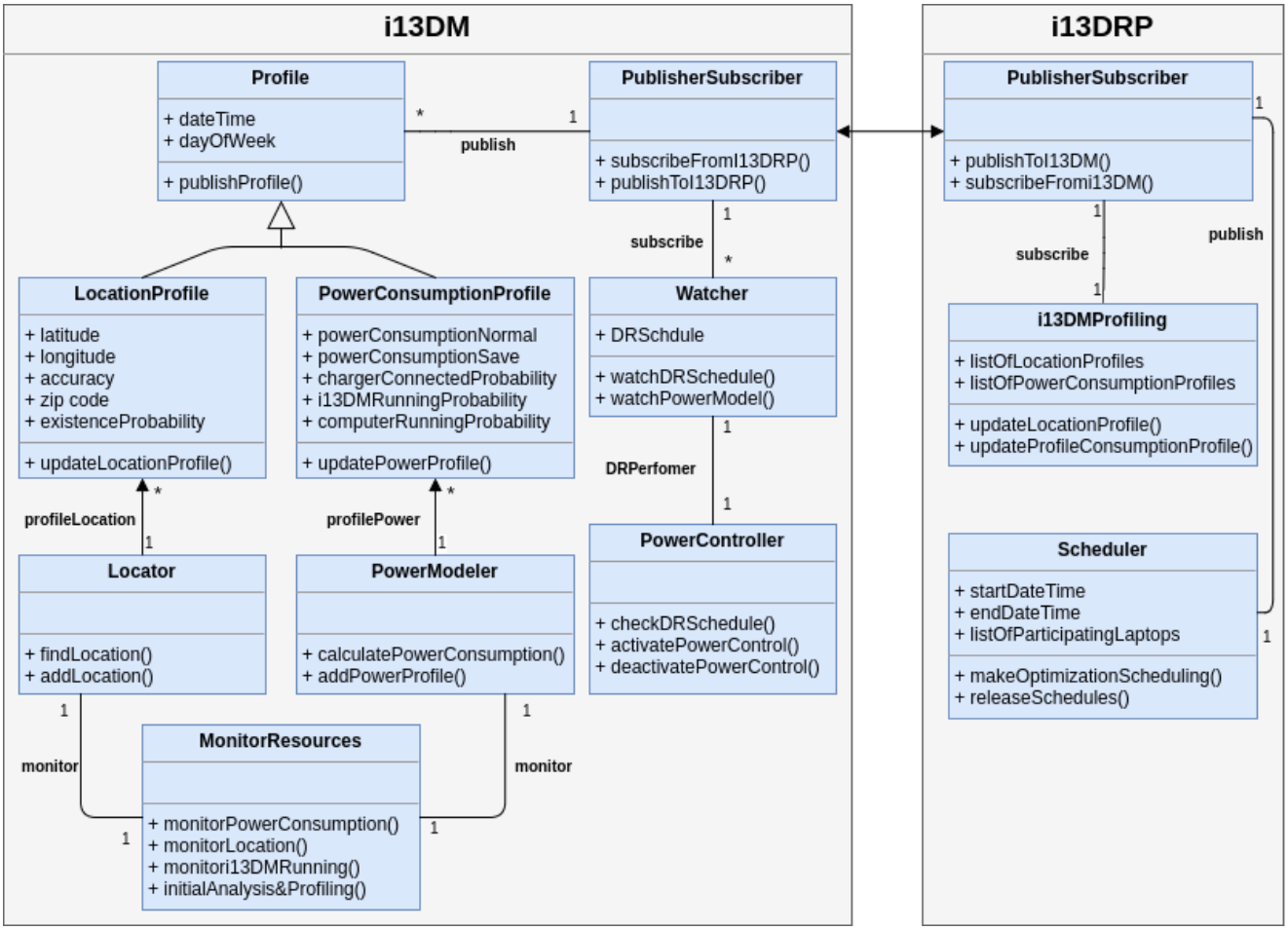


Figure 4.2: Class diagram of i13DR

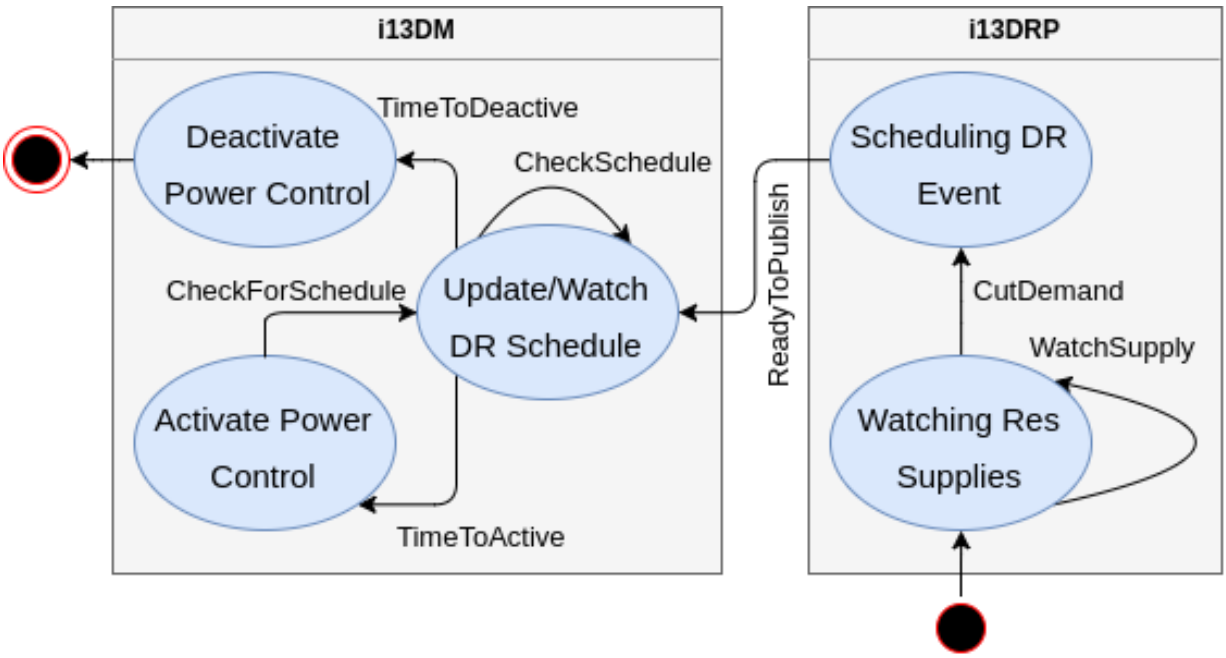


Figure 4.3: State machine diagram of a performed DR event

Chapter 5

System Design

In this chapter, we transform the analysis models of i13DR to system design models during which we define the design goals of 13DR and decompose our system into smaller subsystems[40]. In the following sections, first, we provide an overview of our system design, then we identify the design goals of our work and represent the subsystems decomposition in a way they are realized and implemented precisely. Furthermore, we describe the selected strategies and techniques for building and implementing i13DR including hardware/software mapping, persistent data management approaches, access control policies, and finally managing the boundary conditions of the system.

Overview

In the previous chapter, we explain the requirements, features, and functionalities of i13DR. We acknowledge several functional and nonfunctional requirements of i13DM and i13DRP explaining the specific outcome of our work. Furthermore, we clarify the most important and complex use cases of i13DR by taking advantages of scenarios and use case diagram and finally, we use analysis object model to illustrate the critical classes and relationships in our work. However, during the requirement analysis, we do not describe our software architecture and the way i13DR is realized and implemented. As a result, we utilize the requirements analysis to achieve the qualities of the system as well as the approaches for developing the subsystems of i13DM and i13DRP. Furthermore, we explain several technologies, such as web servers, databases, security measure, development frameworks, which we do not develop but utilize, to facilitate the activities of i13DR.

i13DR Design Goals

During the first phase of system design, we identify the most critical qualities of our system also known as design goals which are realized by the help of functional and nonfunctional requirements and application domain[40]. According to [40] system recognizes and satisfies multiple design goals under several criteria such as performance, dependability, cost, maintenance and end user experience. As a result, we recognize and develop the following design goals for i3DR; for the sake of clarity, we present design goals of i13DM and i13DRP separately.

First, we realize the following goals for i3DRP:

- i13DRP should be fault tolerant to failures of any included applications and loss of connectivity.
- All parts of i13DRP should recover automatically and immediately in case of failures and system crashes.
- i13DRP should constantly maintain bidirectional real-time connections with every instance of i13DM.
- Performance of i13DRP should not noticeably decrease as the number of connected i13DM increases. Also, it should scale up accordingly.
- i13DRP should be tolerant to systematic or intentional failures caused by i13DM demand side users.
- i13DRP should securely protect the sensitive DR participants and utility information.
- Any communication between i13DRP and i13DM should be securely encrypted.
- To keep the financial cost of development and deployment minimal, all the utilized technologies and tools should be supplied from free and open source resources. However, when a free alternative is not available, the necessary tools should be purchased.
- All the applications developed for i13DRP should be designed according to the proper software engineering design patterns for minimizing the required effort for extending and modifying their functionalities.
- i13DRP should use standard communication protocols for connecting different i13DRP applications to minimize the effort required for porting the system to other

platforms.

- i13DRP should offer high usability for human administrators.

Then, we explain the following design goals for i13DM:

- When i13DM needs to communicate with i13DRP, it should be able to establish a non-blocking bi-directional real-time connection with i13DRP with less than one second delay.
- All the essential activities of DR events which are expected to run on the demand side should be encapsulated, packaged, and included in i13DM.
- i13DM, while running on participant's laptop, should have a negligible effect on the performance of laptops as well as consumed resources such as network and battery.
- i13DM should be tolerant to any intentional or unintentional malicious interaction of participant.
- i13DM should recover from any crashes and system failures immediately and automatically. Furthermore, i13DM should submit all the collected exceptions and stack traces to i13DRP.
- By enforcing security measures, i13DM should minimize the abilities of any attackers to sabotage the participants' laptops.
- Execution of i13DM and performing DR programs should not impose any financial cost on the users.
- Building, packaging, releasing, configuring and monitoring of i13DM should be done automatically by utilizing DevOps tools.
- Installing updates and new versions of i13DM should be executed automatically and seamlessly on users' laptops.
- i13DM should be a cross-platform desktop application running on a variety of platforms and operating systems.
- Users of i13DM should be able to easily, without and prior required knowledge, control the behavior of i13DM to the extent it allows. However, all that required operations for performing DR, monitoring, etc should be executed automatically in the background.

Finally, while designing, implementing, building and releasing different parts of i13DR, we emphasize on satisfying all the stated design goals.

i13DR Subsystem Decomposition

To decrease the complexity of the solution domain, we break down the whole system into smaller parts which are more straightforward to be realized [40]. These smaller, simpler parts, are called subsystems, encapsulating the behavior of a few solution domain classes. Moreover, they provide a set of operations through well-defined interfaces, called services. We decompose the i13DR into subsystems with low coupling and high coherence for decreasing the dependencies between two or multiple subsystems and also for increasing the dependencies among classes within a subsystem [40]. Finally, to maintain the complexity of i13DR, we utilize the suitable system architecture for i13DM and i13DRP.

Figure 5.1 illustrates a UML diagram representing the system decompositions of i13DR. We break up the system according to discovered materials in chapter 4, and for better readability, we are omitting attributes and operations from classes. On the left side of diagrams, the subsystems constructing i13DM are included, and on the right side, we present subsystems of i13DRP.

DRManager, *DeviceManager* and *DeviceAnalyzer* are the fundamental parts of i13DM carrying out the necessary tasks of a DR event. *DeviceAnalyzer* contains the entities responsible for power consumption, location, and the i13DM running profiling. Furthermore, it also performs the first-time full-system analysis and submits it to i13DRP through *Communicator*. *DeviceManager* consists of classes responsible for the general behavior of the application and its interaction with the platform including classes for managing the application's settings, for reporting the application crashes and failures to i13DRP, for auto-launching i13DM at start up finally classes for updating i13DM to the latest version. Moreover, *DRManager* holds all the classes which are responsible for the activities required during a DR event, for example, scheduling the event and controlling the power consumption. Furthermore, i13DM takes advantages of a notification subsystem for informing the user of laptops when changing the standard behavior of laptop. Finally, i13DM is equipped with a relational database for storing records produced over the lifecycle of i13DM.

As previously stated in 4, i13DRP relies on cooperations of multiple server-side applications, however, to keep the diagram simple, we do not specify the borders of each application in this picture. Also, subsystems of different application are depicted together in one diagram. We will specify the order of applications more clearly in section 5.4. As we stated the system requirements in detail in the previous chapter, the main task the of i13DRP is managing the participating laptops and

performing DR event. For this reason, we realize the subsystems of *DRManager*, *i13DMManager* and *i13DMProfiler*. *DRManager* carries out the essential scheduling operations in association with *i13DMProfiler* which is responsible for managing the power consumption and location profiles collected from i13DM devices as well as caring for the on-line status and activities of them. Moreover, *i13DMManager* provides the administrator of i13DRP with essential operations for managing the registered laptops. In addition to described subsystems, several other subsystems facilitate i13DRP operations including *CrashManager* handling the reported i13DM crashes, *i13DMUpdater* and *DevOpsManager* maintain building and releasing new versions of i13DM and i13DRP and finally, *AuthN/AuthZ* which maintains the authentication/authorization and security policies of i13DM and i13DRP. In the end, i13DRP is dependent on a storage subsystem consisting of a file storage for maintaining power models and a real-time database for storing DR related records.

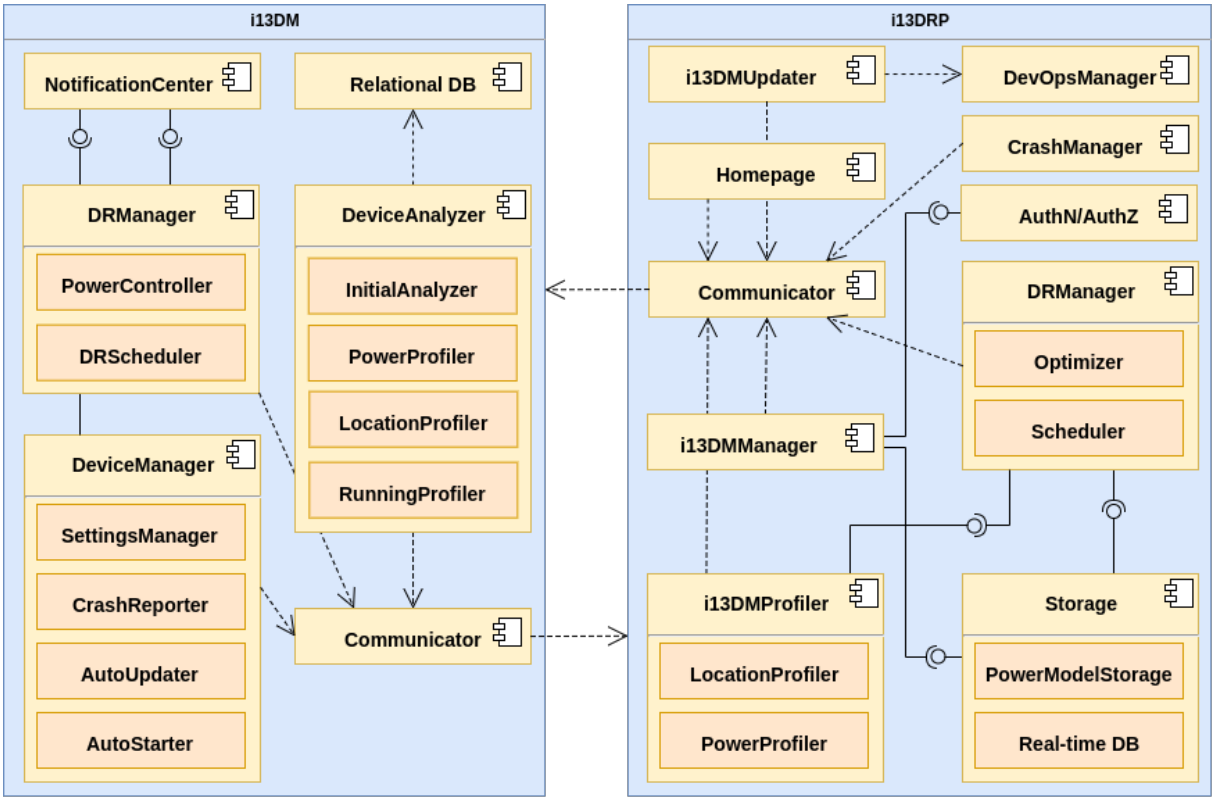


Figure 5.1: Subsystem decomposition of i13DR

i13DR Hardware Software Mapping

In the following section, we explain our approach toward mapping the realized subsystems to different hardware and software components in a way that we satisfy the mentioned functional and nonfunctional requirements as well as the design goals. Moreover, we exploit UML deployment diagrams for simplifying the illustration. According to [40] deployment diagrams demonstrate the relationship among run-time components of the system. The components encapsulate the activities of subsystems and provide them as services to other components. Figure 5.2 presents our approach toward separating our subsystems and mapping them to appropriate devices and services. On the left side, i13DM is installed and running on user's laptops, and the right side, the activities of i13DRP is distributed among multiple applications and physical devices. In the following subsections, we discuss in detail how each subsystem is mapped and what technologies and tools used for implementing the i13DR.

i13DM Implementation

According to chapter 4, i13DR is inherently a distributed system consisting of multiple server-side applications and several participating laptops with i13DM installed. We design and develop i13DM in a way that all required demand side components and subsystem are encapsulated and packaged into one installable desktop application. In our work we used *Electron Framework*[1] to develop a cross-platform desktop application for Microsoft Windows 7,8 and 10 32/64-bit as well as Ubuntu 16.04 LTS 64 bit. *Electron Framework* is a JavaScript based open-source framework meant for creating front and back end components of a cross-platform desktop application. It makes use of *Node.js*[2] for handling the back end and *Chromium web browser*[3] for the front end. The main reason for selecting this framework, in addition to being cross-platform, is taking advantage of event-driven architecture and non-blocking asynchronous nature of Node.js, empowering us to develop an application capable of real-time communication with other components.

To implement the realized functionalities of subsystems mentioned in 5.1, we implement i13DM according to best practices of JavaScript, Node.js and Electron Framework. However, we also make use of several built-in functionalities of Node.js and Electron

[1] `https://electron.atom.io/`
[2] `https://nodejs.org/en/`
[3] `https://www.chromium.org/`

Framework as well as the Node.js packages provided by *Node Package Manager (npm)*[4]. In the following paragraphs, we discuss how every subsystem and its corresponding classes are mapped and implemented.

First, we review the subsystems and their classes which are implemented based on built-in features of Electron framework or Node.js packages. We used several other packages. However, we only include the most important ones:

- *NotificationCenter*: We use the built-in Notifications API of chromium to configure and display notifications to user.
- *RelationalDB*: We employ *Lovefield*[5], relational database built upon built-in IndexedDB of Chromium to mimic SQL-like API.
- *SettingsManager*: It is heavily dependent on the npm package *configstore*[6].
- *CrashReporter* and *AutoUpdater*: We make use of built-in features of Electron framework for implementing crash reporting features.
- *AutoStarter*: This subsystem is based on npm package of *auto-launch*[7].
- *Communicator*: i13DM either communicates with the real-time database through npm package *firebase*[8] or directly communicates with the i13DRP web applications by means of built-in HTTP and HTTPS interfaces of Node.js.

The rest of the subsystems which we develop from scratch include:

- *PowerController*: We use the built-in power management features of the operating system to limit the energy consumption of the laptop. The ideal would be disconnecting the battery charger from the plug and only drain energy from the battery. However, the OS does not provide any functionalities for this purpose. Hence, we use the OS API to put the laptop on power save mode. On Windows we rely on *powercfg*[9] to import, activate and deactivate a Windows specific energy save scheme which we previously generated and exported from a Windows machine and we ship it with i13DM. On the other hand, Ubuntu is equipped with very few built-in functionalities for controlling the power consumption. For this reason, to save power, we only dim the screen though the built-in command *xrandr* or turn the screen off when the laptop is idle.

[4] `https://www.npmjs.com/`
[5] `https://github.com/google/lovefield`
[6] `https://github.com/yeoman/configstore`
[7] `https://github.com/Teamwork/node-auto-launch`
[8] `https://www.npmjs.com/package/firebase`
[9] `https://msdn.microsoft.com/en-us//library/hh824902.aspx`

- *InitialAnalyzer*: For implementing this subsystem we make use of built-in functionalities of OS to extract the hardware and software specification. On Windows, we are heavily dependent on *Windows Management Instrumentation (WMI)*[10] Commands to query the system specification. However, on Ubuntu we rely on several commands, including but not limited to *lshw*, *dmidecode*, *lshw*, *lspci* and *lscpu* to collected the necessary system specifications.
- *DRScheduler*: We develop this subsystem based on pure JavaScript to compare the internal clock of the machine with a DR schedule earlier received from i13DRP. Once the current time falls into any DR plans, it informs the *PowerController* to activate the power control and afterward to deactivate when the DR event is over.
- *LocationProfiler*: This subsystem is a combination of JavaScript methods that we develop to monitor the actual location of laptop every 90 seconds in cooperation with *Google Maps API* [11], built-in Geolocation API of Chromium and FreeGeoIP [12] an open-source HTTP API for searching the geolocation of IP addresses which we host on our servers.
- *RunningProfiler*: This subsystem is implemented based on pure JavaScript performing the tasks responsible for profiling the i13DM.
- *PowerProfiler*: This subsystem runs in interval exploiting the built-in functionalities of OS to extract the essential system utilities to be used with the deployed power model. On Windows, similar to *InitialAnalyzer*, it uses WMI commands to query the utilization metrics and on Ubuntu it executes multiple commands including by not limited to *upower*, *free*, *iostat* and *vmstat*.

The source code of i13DM is open-source and publicly available at *GitHub* repositories[13].

[10] `https://msdn.microsoft.com/en-us/library/aa384642(v=vs.85).aspx`
[11] `https://github.com/epezhman/demand-manager-app`
[12] `http://freegeoip.net`
[13] `https://github.com/epezhman/demand-manager-app`

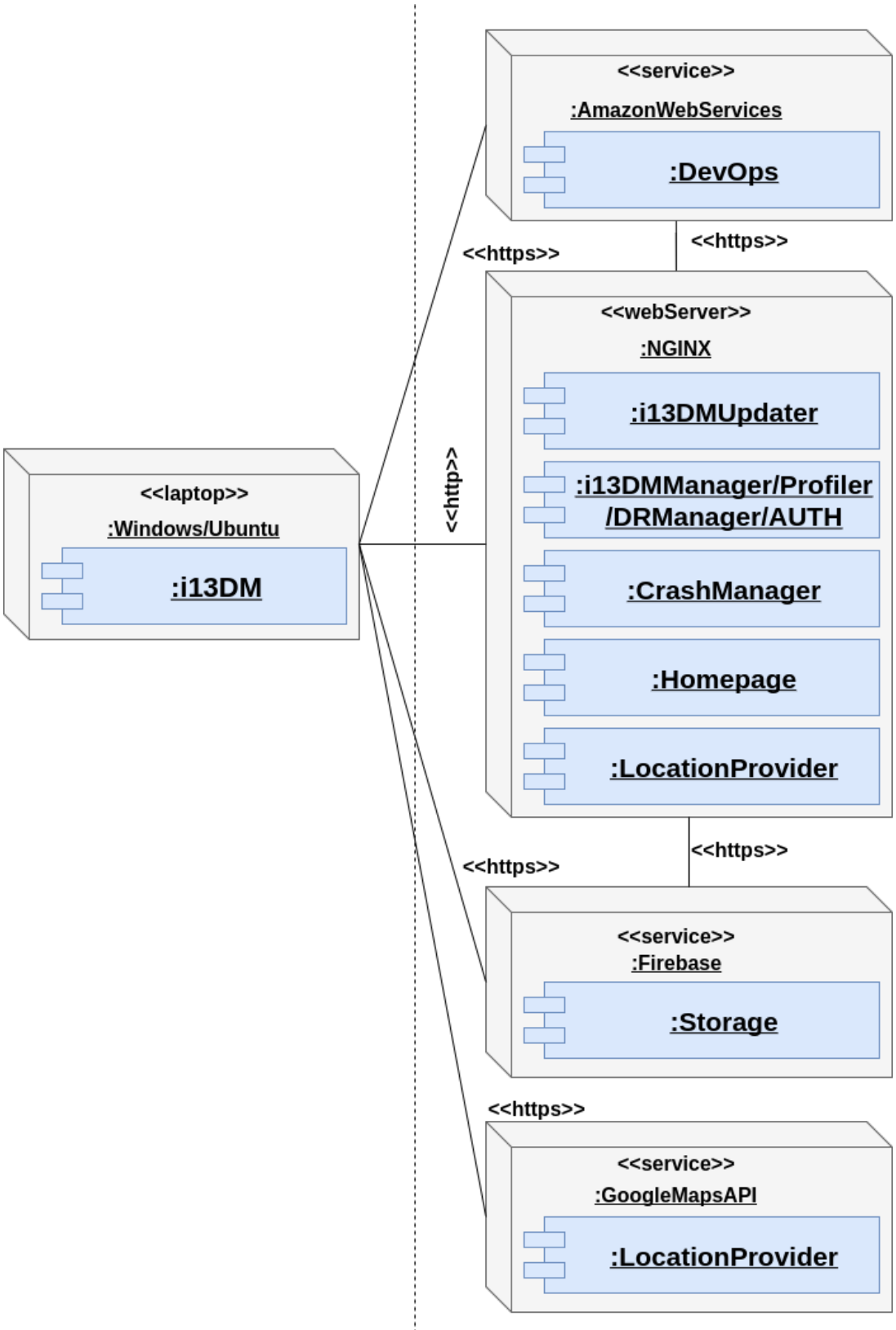


Figure 5.2: Hardware/software mapping of i13DR

i13DRP Implementation

According to requirements discussed in chapter 4, i13DRP comprises of several web-based applications, and we emphasize on real-time communication and scalability as first-order design constraints. As figure 5.1 and 5.2 illustrate, we implement and host some of the subsystems independently on our servers and for the rest we use available services provided by commercial platforms. First, we review the subsystems and classes which we deploy on commercial platforms:

- *DevOpsManager*: The backbone of our DevOps infrastructure is *Amazon Web Services* and *Amazon S3 (Simple Storage Service)* [14], a web service providing storage. The very high availability, scalability, and low cost make *Amazon S3* a good choice for hosting and distributing the packaged i13DM. Furthermore, we use *Gulp.js*[15], an open-source JavaScript toolkit used as streaming build system, and *electron-builder*[16], the de facto builder and packager for electron framework, to automate the building, configuring, testing, packaging and releasing the i13DM to *Amazon S3*.

- *Storage*: The main storage provider of i13DRP is *Firebase*[17] providing us with a real-time cloud-hosted NoSQL database for storing and retrieving the i13DM and i13DRP data in JSON format. Additionally, *Firebase* offers a cloud storage which we use for distributing the constructed power model files to i13DM in real-time. The main reason for choosing this platform is the high scalability of the offered services which are empowered by Google infrastructure to support multiple thousands of simultaneous connections as well as appropriate prices, high security, and availability.

- *LocationProvider*: We provide various options for detecting the location of i13DM to be used as a back-up in the case of any provider's failure. The main service we use is *Google Maps API*[18].

In the following list, we explain all the subsystems and classes which we host on our servers. We develop the necessary subsystems. However, we also make use of open-source projects and tools when necessary:

- *DRManager(Optimizer/Scheduler)*, *i13DMProfiler(LocationProfiler/PowerProfiler)*

[14] `https://aws.amazon.com/s3/`
[15] `http://gulpjs.com/`
[16] `https://github.com/electron-userland/electron-builder`
[17] `https://firebase.google.com/`
[18] `https://developers.google.com/maps`

and *i13DMManager*: We develop a web application based on *Meteor*[19] and *Angular 2+*[20] encapsulating and packaging the three subsystems. Meteor, an open-source JavaScript web framework, makes real-time communication possible by utilizing the *Distributed Data Protocol (DDP)*[21] and publish-subscriber pattern to propagate data changes. For this reason, we employ Meteor to develop the back-end of i13DRP. However, Meteor does not provide adequate functionalities to develop the front-end of i13DRP. Therefore, we use Angular 2+, an open-source *TypeScript-based* front-end web application platform.

- *i13DMUpdater*: We develop a simple *Express*[22], a Node.js web framework, to propagate the latest version of i13DM to participants when a new version is available at *Amazon S3*.
- *CrashManager*: Two different web applications are responsible for receiving and maintaining crash reports and stack traces from i13DM. First, we use *mini-breakpad-server*[23], a small crash collector server inspired by *google-breakapd* and developed by Electron's development team. The second crash manager application is developed by us, based on *Django Web Framewrok*[24] to collect the internal run-time exceptions sent by i13DM.
- *AuthN/AuthZ*: By using the measures and rules provided by Meteor and Firebase we enforce authentication and authorization. We discuss this topic in detail in section 5.6.
- *Homepage*: We develop a static HTML page as a landing page for our project, where a user can download i13DM as well as informing themselves about the project. Moreover, the administrators of i13DRP can navigate to management areas. Our homepage is accessible under `http://i13dr.de/`.
- *Communicator*: We serve ll the described subsystems and application by *NGIX*[25], a high-performance HTTP and reverse proxy server, running on a Ubuntu 16.04 LTS.

In the end, the source code of the discussed open source project can be found under the given links. Our implementations are open-source and freely available on ourGitHub

[19] `https://www.meteor.com/`
[20] `https://angular.io/`
[21] `https://www.meteor.com/ddp`
[22] `http://expressjs.com/`
[23] `https://github.com/electron/mini-breakpad-server`
[24] `https://www.djangoproject.com/`
[25] `https://nginx.org/`

repositories for i13DRP Meteor/Angular application[26], the crash collector[27] and update server[28]. Furthermore, we include a few snapshots of i13DM and i13DRP in the appendix.

Power Model Construction

In chapters 2 and 4, we discuss the purpose and requirements of a power model. In our approach, we propose a modeling technique for generating a mathematical model which estimates a laptop energy consumption in real-time according to the measured system metrics. Once the model is created based on a System Under Test(SUT), it can be deployed on participating laptops to be used for predicting the power consumption.

We need to create the following power models to estimate the power consumption according to the operating system and power consumption mode:

- Microsoft Windows on normal power consumption mode
- Microsoft Windows on power save mode
- Ubuntu on normal power consumption mode
- Ubuntu on power save mode

In order to construct any of the power models we use the procedure illustrated in figure 5.3 consisting of six distinct phases. We perform the calculations by making use of *R Programming Language*[29] and *R Studio*[30].

The first phase is collecting the necessary power related data required for training the energy model of an operating system running on an SUT. Therefore, we make use of *Lenovo ThinkPad L540*[31] with specifications described in Table 5.1. Furthermore, the mentioned laptop is connected to *MEDAL* power meter which constantly measures and stores real power, apparent power, power factor, voltage and current of the connected device. Furthermore, we log the system metrics of the laptop with a one-second interval on Ubuntu and a three-seconds interval on Windows. The logged system metrics include CPU Utilization in percent, display brightness in percent, power drain of battery in Watts, the charging/discharging status of laptop, the remaining capacity of battery in percent,

[26] https://github.com/epezhman/demand-manager-web-app-v-2
[27] https://github.com/epezhman/demand_manager_admin
[28] https://github.com/epezhman/demand-manager-update-server
[29] https://www.r-project.org/
[30] https://www.rstudio.com/
[31] https://www.notebookcheck.net/Review-Lenovo-ThinkPad-L540-20AV002YGE-Notebook.113894.0.html

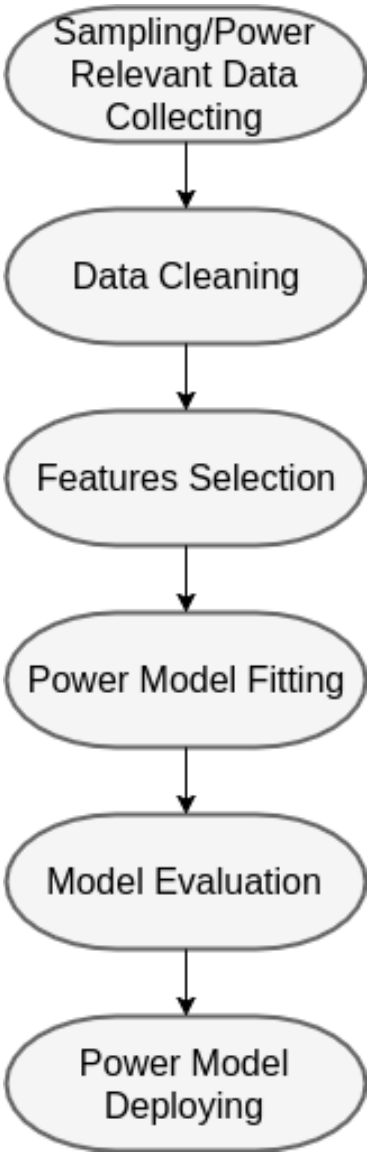


Figure 5.3: Power model generation procedure

memory usage in percentage, disk read/write kb per seconds and request and network download/upload rate in kb per seconds. We collect the data over a period of four days for each OS and power consumption mode, while the user works with laptops performing his common daily activities.

The next phase is preprocessing, cleaning and removing the outliers from raw data. We consider the rows with measured real power below 8 watts and above 65 watts (the maximum output of the used AC adapter of the SUT laptop) to be outliers because of errors is measurement, and hence, we remove them from the dataset. Furthermore, we

normalize the data types of features in the dataset as well as transforming the recorded values of download/upload rate and disk read/write rate to values between 0 and 100. Once the correction of raw data is finished, we save them to CSV files and use them later for training the model.

For predicting the power consumption, we fit a linear regression model based on the collected data. However, before training the model, for the purpose of improving the accuracy and reducing the complexity of the constructed model, we examine the quality of different combinations of features for selecting a subset of features for a more accurate model. For this reason, we use package *leaps*[32] to perform an exhaustive search to determine the best subset of power-related features. Figur 5.4 illustrates the result of an exhaustive search on all subsets of the regression model for the dataset collected from Windows while in normal power consumption mode. *leaps* sorts the results by *Adjusted R-Squared* and *Bayesian Information Criterion (BIC)*[33] and examines them for the five best models reported for each subset size (one feature, two features and so on). The Adjusted R-Squared represent the proportion of variation in the measured power consumption that has been explained by the model with taking the number of features in the model into consideration. BIC measures the goodness of a model based on the maximized value of a likelihood function. The higher the calculated value for Adjusted R-squared, the better the constructed model whereas as the value of BIC decreases, the model improves. According to the result of the evaluations, we select a subset of features presented in Table 5.2 to be included in the final model. We include the plots for Windows in power save mode, Ubuntu in normal power consumption mode and power save mode in appendix due to lack of space.

Afterward, first, we use the selected features to fit a simple linear regression model by using the *lm* method in *R*. Then we export the model to be deployed to i13DM. However, before deployment, we evaluate their accuracy of constructed model which is explained in detail in chapter 6.

The scripts for cleaning data, feature selections, fitting and evaluating models are available at a GitHub repository under the following address, `https://github.com/epezhman/demand-manager-data`.

[32] `https://cran.r-project.org/web/packages/leaps/leaps.pdf`
[33] `http://r-statistics.co/Linear-Regression.html`

Vendor	Lenovo
Version	ThinkPad L540 (20AV002YGE)
CPU	Intel - Core i5-4200M CPU @ 2.50GHz
Video Card	Intel - 4th Gen Core Processor Integrated Graphics Controller
Memory	8GB SODIMM DDR3
Wireless interface	Intel Wireless 7260
Storage	Seagate - ATA HGST HTS725050A7
Battery	Sanyo - 45N1769 -56160mWh
Operating System	Ubuntu 16.04 LTS / Microsoft Windows 10

Table 5.1: Lenovo ThinkPad L540 specifications

Persistent Data Management

i13DR requires storing two sets of objects, one set for i13DM and the other for i13DRP. According to our needs and resources available on each platform, we select different approaches.

The persistent data on i13DM falls into two categories. The first group is the data written and modified few times, but i13DM reads them several times over its lifetime. i13DM stores this kind of data in configuration files provided by *npm* package *configstore* and it consists of application settings, power model location on i13DRP, DR schedule details and the power profiling interval in milliseconds. The second group contains several rows of data which are required to be stored, queried, retrieved and updated regularly. Because of this high interactions, we make use of a relational database. To avoid adding extra dependencies, we install and use *lovefield* package for exploiting the IndexedDB[34] integrated with Chromium shipped with Electron framework. Lovefield provides an SQL-like API to interact with NoSQL IndexedDB, mimicking the behavior of a common relational database. Figure 5.6 shows, five tables we use for storing data. On first-run, i13DM creates an initial power consumption and location profiles by populating the tables *PowerProfile* and *LocationProfile* respectively. During execution of i13DM, during each cycle of power consumption, location and running profiling, it adds a new

[34] `https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API`

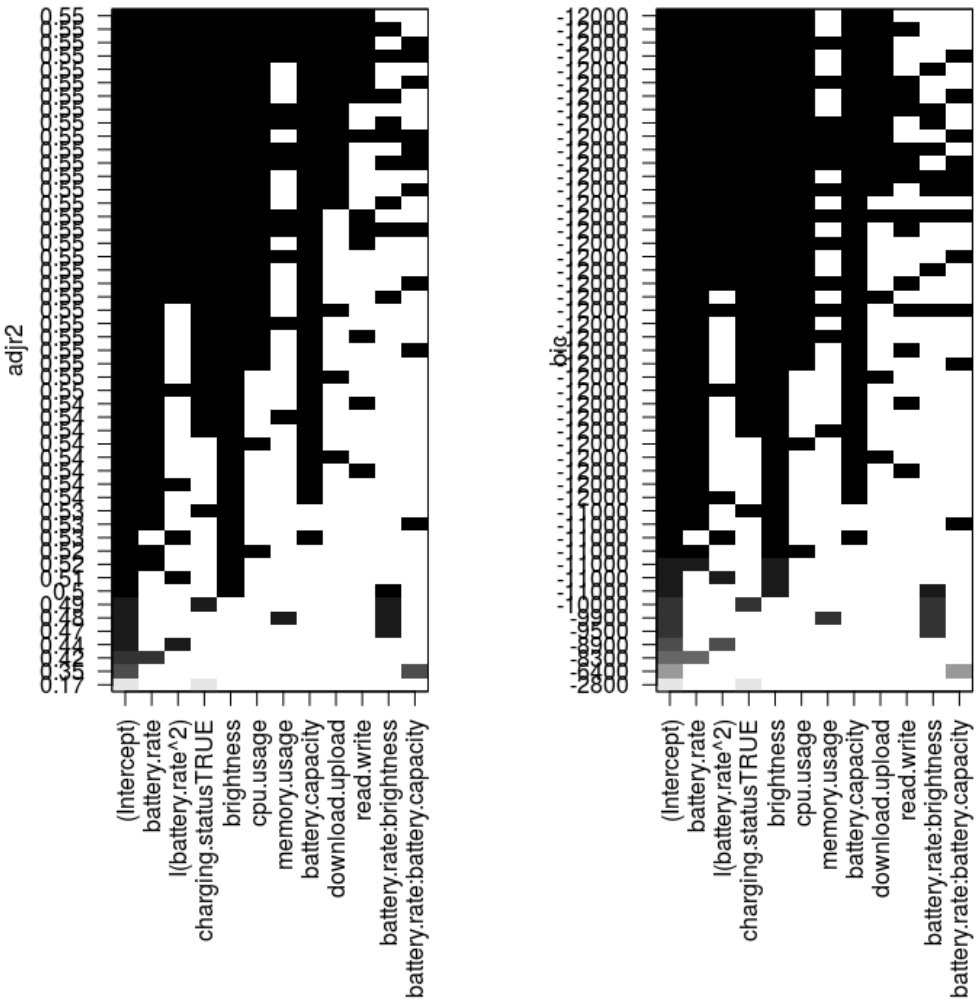


Figure 5.4: All subsets regressions for Windows in normal power consumption mode

records to *Power*, *Location* and *Running* tables respectively and the corresponding records are updated on tables *PowerProfile* and *LocationProfile*. According to our default settings, data older than 30 days is deleted from *Power*, *Location* and *Running* but the data in *PowerProfile* and *LocationProfile* remains on the system during the lifetime of i13DM. Because of limitations of lovefield and performance issues we avoid using foreign keys to keep our database design simple, however, we define indexes when necessary.

The requirements for i13DRP emphasizes on the necessity of using a real-time scalable file storage and database for serving potentially a significant number of i13DM instances.

OS	*Power Mode*	*Selected Features*
Windows	Normal & Power Save	*Battery charging/discharging rate*, *Battery charging/discharging rate squared*, *Interaction of battery charging/discharging rate with display brightness* , *Interaction of battery charging/discharging rate with remaining capacity of battery*, *Charging status*, *CPU usage*, *Memory usage*, *Remaining capacity of battery*, *Download/upload rate in kb*, *Disk read/write request per second*
Ubuntu	Normal & Power Save	*Battery charging/discharging rate*, *Battery charging/discharging rate squared*, *Interaction of battery charging/discharging rate with remaining capacity of battery*, *Charging status*, *CPU usage*, *Memory usage*, *Remaining capacity of battery*, *Download/upload rate in kb*, *Disk read/write request per second*

Table 5.2: Selected power related features for power modeling

As a result, we choose Firebase to maintain our data. Firebase is a real-time cloud-hosted NoSQL database storing data as JSON objects which we structure according to their guidelines and best practices[35]. Figure 5.5 represents how we structure our data on the JSON tree by flattening the data structure to increase the performance of Firebase and to ensure the scalability. As figure 5.5 illustrates, all the children of the root node are specifying the topic of the data they are containing as well as containing as many children as the number of registered i13DM (here three laptops with three different IDs). The main advantage of Firebase is i13DM can directly read from it and write data to it through the available web API. The following list briefly explains the nature of the data stored in each node:

[35] `https://firebase.google.com/docs/database/web/structure-data`

- *activity-status*: We store the latest status of i13DMs under this node such as last on-line time.
- *device*: This node is responsible for storing the detailed list of registered laptops including the version of i13DM as well as the date of joining i13DR.
- *hardware*: The hardware specifications gathered during initial analysis are stored under this node.
- *last-location*: i13DRP tracks the most recently determined location of i13DM to be reported to the administrator.
- *location*: This node holds the location profiles sent by i13DM.
- *logging*: If the administrator decides on logging the laptop for generating the power model, i13DRP stores the collected logs under this node.
- *online*: i13DRP maintains a list of currently on-line i13DM under this node.
- *power*: The power consumption profiles sent by i13DM is stored under this node.
- *schedule-period*: After DR optimization and scheduling, the generated schedule for every i13DM is published under this node.
- *settings*: This node keeps the settings for every i13DM which can be modified by the administrator.
- *statistics*: Firebase maintains the statistics of the stored data such as the number of registered laptops.

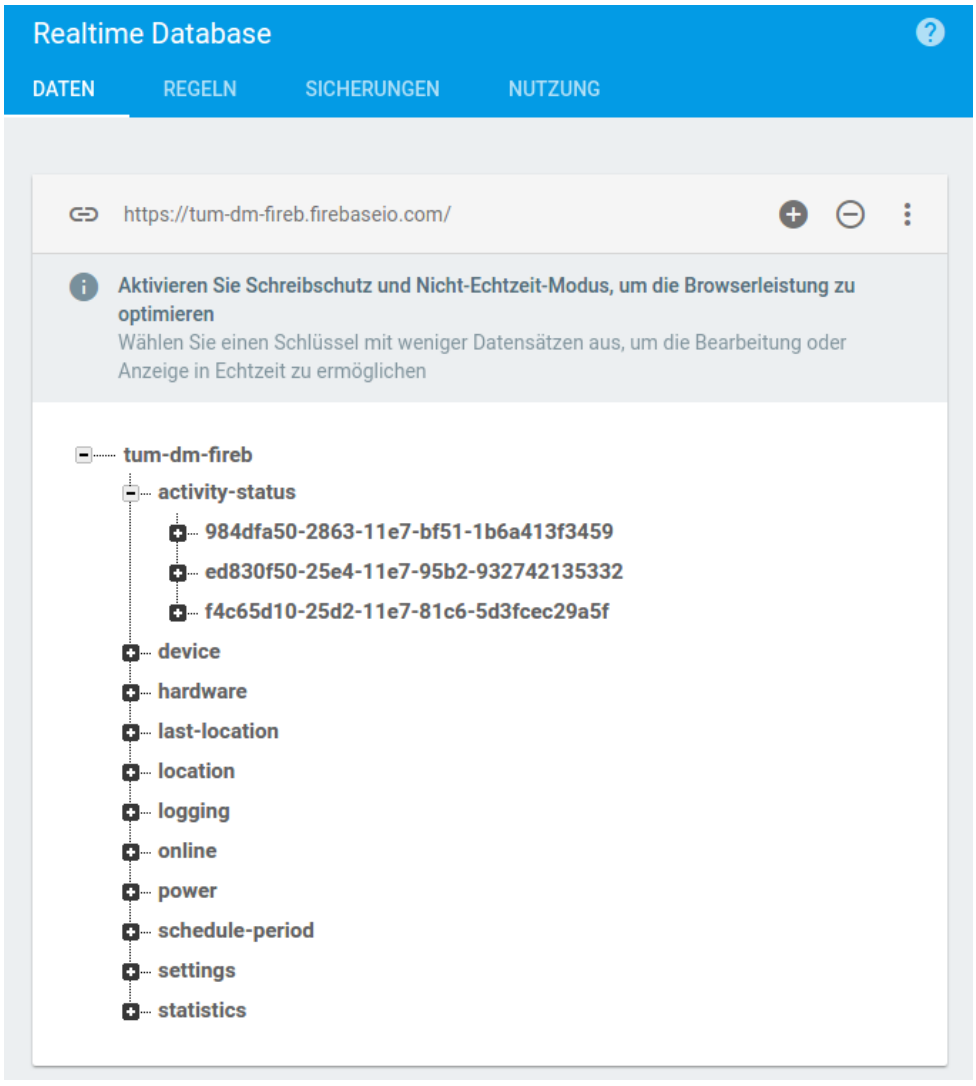


Figure 5.5: Data structure on Firebase

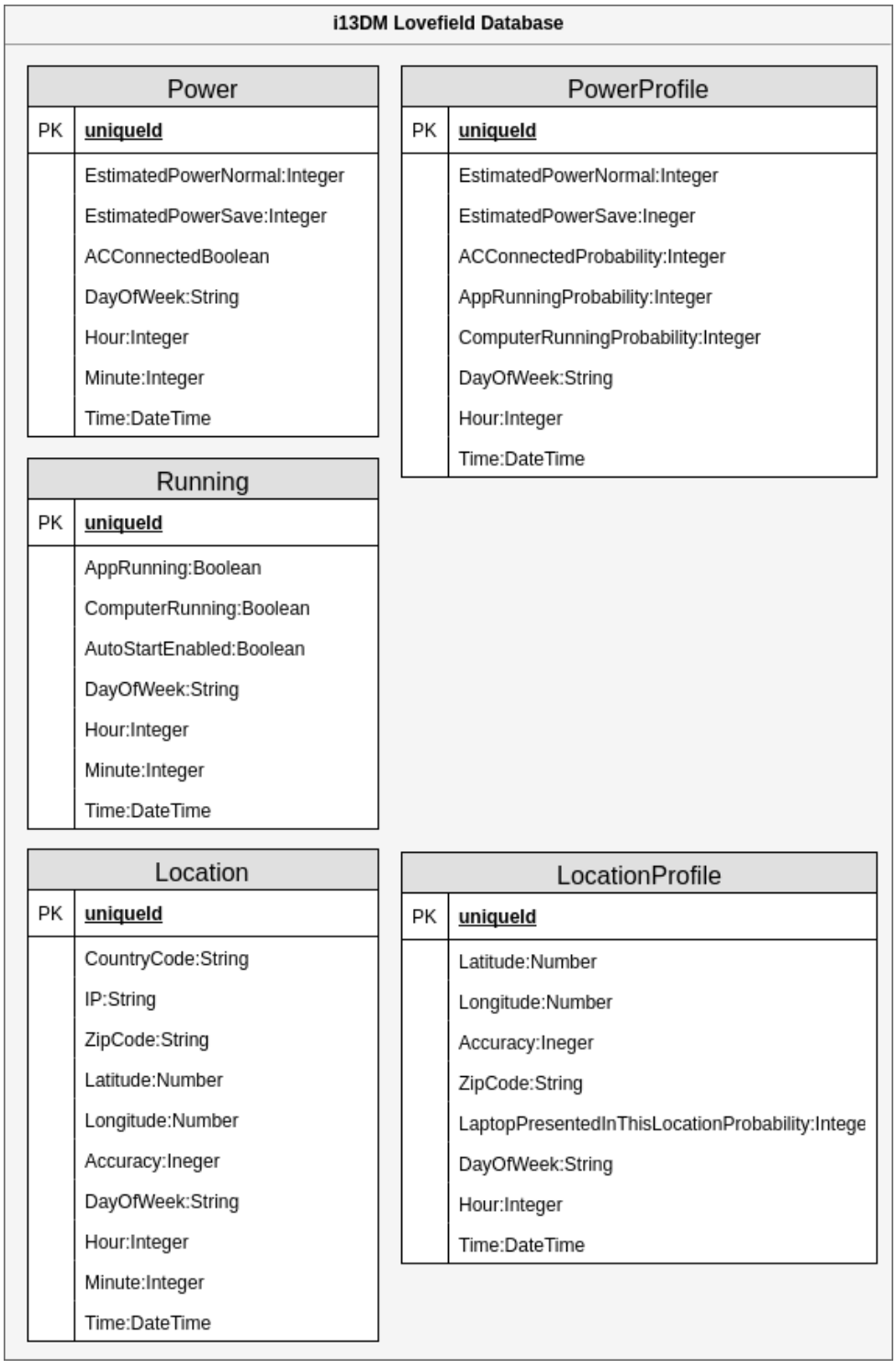


Figure 5.6: Database diagram of i13DM

Access Control

In general two kinds of human users interact with i13DR. On one side, the users of i13DM who have the role of DR participants and on the other hand, there are administrators of i13DR who directly interact with i13DR having an administrative role. DR Participants must only have access to the information directly related to their device and not any other DR participants, whereas, the administrators have access to the entire data on i13DR infrastructure. Therefore, we implement a number of security measures to ensure granting correct access rights to the user.

The administrators of i13DRP first must be approved by one of the previously approved administrators. Afterward, to access the front-end of i13DRP and then the data stored on Firebase, first, they should login by using the built-in authentication functionalities of Meteor framework and then authenticate themselves with authentication providers of Firebase.

Because of direct interaction of i13DM with Firebase, our approach for implementing the security measures of DR participants heavily depends on Firebase user-based security integrated with Firebase authentication providers. To ensure the anonymity of our DR participants, we do not request any private information such as names and emails. However, we employ a Firebase authentication capability known as *Anonymous Authentication*[36] to create temporary anonymous accounts allowing users to have access to protected data. Moreover, we use *Firebase Realtime Database Rules*[37] to protect i13DM data and determine to what extent users have access to the data. We specify that only anonymously authenticated i13DMs have read and write access to the power consumption and location profile, logs, and first analysis records and only read access to the published power models and DR schedules.

Finally, we make use of secure connections to transfer most of the i13DR data, including all the interactions with real-time database and cloud-based file storage. To guarantee the security of our servers running the i13DRP, we make use of Firewalls as well as secure SSH connections. Furthermore, we assure that all of the running platforms are updated to the latest version, and we follow the recommended security guidelines of all employed programs and frameworks.

[36] `https://firebase.google.com/docs/auth/web/anonymous-auth`
[37] `https://firebase.google.com/docs/database/security/`

Boundary Conditions

In the last section, we review our approach for handling i13DR boundary conditions, such as installing i13DM, starting and stopping i13DM and i13DRP applications as well as handling their run-time exceptions and system failures.

As mentioned in 5.1, we create a homepage for accessing information on the i13DR project as well as downloading the installable version of i13DM which is available at this address: `http://i13dr.de/`. After, downloading the platform compatible version of i13DM, either Windows 32/64-bit or Ubuntu 64-bit, DR participants can install and launch the i13DM in a similar way to any other software. Furthermore, DR members can stop the execution of i13DM from the application's tray icon from system tray. The exceptions, caused by system failure, software fault, network failure, etc. are handled gracefully in the background, and the user is informed with notification only if the failure affects the normal behavior of the system. In any case, a report of the exception is sent to i13DRP to be reviewed and processed by administrators.

i13DRP consists of multiple web based application which either is developed by us and is running on i13DR's private servers or is being offered by third-party web services. Therefore, we only need to pay attention to the execution and availability of services hosted by us. Depending on the application and its architecture and requirements we use a variety of tools and techniques for setting them up and monitoring them. For deploying, starting up and running our i13DRP Meteor/Angular app we use *Meteor Up*[38] a tool powered by *Docker*[39], a software container platform. Furthermore, we use *PM2*[40], a Node.js process manager, to manage our Node.js based applications including *stack trace collector*, *i13DM update server* and the *homepage*. Finally, The *Django-based crash collector* is served by *uWSGI*[41], an application container server. We should mention that we configure the employed platforms and services to restart automatically in a case of system failures and crashes.

[38] `https://github.com/zodern/meteor-up`
[39] `https://www.docker.com/`
[40] `http://pm2.keymetrics.io/`
[41] `https://uwsgi-docs.readthedocs.io/en/latest/`

Chapter 6

Evaluation

As discussed in the previous chapters, the primary focus of the thesis is implementing a real-time scalable DR infrastructure for laptops based on combining two lines of research. On the one hand, we make use of studies on creating effective DR infrastructure, on the other side, with the help of previous works on energy consumption modeling, we construct mathematical models to measure the power consumption of laptops in real-time. Therefore, in the following chapter, we evaluate the performance and accuracy of the developed i13DR framework and constructed power models. First, we explain the methodologies we use for evaluating energy models followed by an experiment we design for quantifying the abilities of i13DR infrastructure. Then, we describe the objectives and hypotheses we expect to realize followed by presenting the results gathered from the experiments. Afterward, we interpret the findings and discuss them in details. Finally, we mention the difficulties we faced while conducting the tests.

Power Models Evaluation

In chapter 5, we explain the procedure for fitting four different linear regression models for estimating real power consumption based on operating systems and energy consumption modes. Before training the model with four different collected datasets, first, we split each dataset into two smaller ones, with one holding 80 percent of the data used for fitting the model and the other 20 percent of the data meant for testing the accuracy of the model for predicting out-of-sample values. Table 6.1 summarizes the evaluation results based on different metrics including *p-Value* of the constructed model, *Adjusted R-Squared* of constructed models, *MAPE* in percentage, *Min/Max Accuracy* in percentage

and *Correlation Accuracy* between the actual and predicted values. *P-Value* and *Adjusted R-Squared* are reported by *R* after fitting the model. *P-Value* points out the statistical significance of the models, and because the *p-Values* of all four models are less than the pre-determined statistical significance (0.05), we can verify that all the constructed models are statistically significant. Furthermore, the *Adjusted R-Squared*[1] represent the proportion of variation in the measured real power that is explained by the model with taking the number of features in the model into consideration and higher *Adjusted R-Squared* implies a better model. The other metric, *Mean Absolute Percentage Error (MAPE)*, which is defined as 6.2, measures the prediction error, and as its value decreases the quality of model increases. Furthermore, *Min/Max Accuracy* is defined as 6.1 and it measures how far the model prediction is from the actual real powers. The better the quality of regression model is, the higher the *Min/Max Accuracy* will be. Finally, *Correlation Accuracy* is a simple correlation between predicted and actual real powers and as the correlation among the values increases, it indicates both values have similar directional moves. We also include a more detailed summary of all models, created by *summery()* method of *R* in the appendix.

According to our evaluation results, all the generated models have relatively good accuracy for the purpose of our thesis. However, the power models constructed for Ubuntu on normal power consumption mode and power save mode have the highest accuracy; in comparison, models for Windows have relativity lower accuracy. The main reason for this variation is the different sampling rate we use on Windows in compare to Ubuntu which is respectively 3 seconds and 1 second. We choose 3 seconds sampling frequency on Windows because WMI is rather slow in updating utilization metrics and any faster sampling would cause in incorrect values with too much overhead on the performance of the laptop.

$$MinMaxAccuracy = mean\left(\frac{min(actualRealPowers, predictedRealPowers)}{max(actualRealPowers, predictedRealPowers)}\right) \quad (6.1)$$

$$MAPE = mean\left(\frac{abs(predictedRealPowers - actualRealPowers)}{actualRealPowers}\right) \quad (6.2)$$

Finally, we perform 5-folds cross-validations on all four models. Figure 6.1 illustrates all the resulting cross-validations side by side executed by package $DAAG$[2] of *R*, where

[1] `http://r-statistics.co/Linear-Regression.html`

[2] `https://cran.r-project.org/web/packages/DAAG/index.html`

Model	*Power Mode*	*p-Value*	*Adj R-sqr*	*MAPE (%)*	*Min/Max ACC(%)*	*Correlation ACC*
Ubuntu	Save	<2.2e-16	0.968	5	95	0.984
Ubuntu	Normal	<2.2e-16	0.9065	8	93	0.954
Windows	Save	<2.2e-16	0.8248	18	85	0.914
Windows	Normal	<2.2e-16	0.5223	19	85	0.713

Table 6.1: Power model evaluation results

the small symbols are predicted real powers and bigger ones are actual real powers. Additionally, table 6.2 represents the average squared errors of five folds. According to the plots, we verify that model's prediction accuracies are approximately uniform among different samples, and the slopes and level of fitted lines have relatively low variations.

Model	*Power Mode*	*Mean Squared Error*
Ubuntu	Save	8.15
Ubuntu	Normal	33.44
Windows	Save	21.61
Windows	Normal	53.97

Table 6.2: Power models cross-validations mean squared error

i13DR Evaluation Overview

We design a comprehensive experiment, consisting of two parts, to evaluate the performance and effectiveness of i13DR infrastructure with a focus on real-time responses and its abilities to control the power consumptions of laptops. The first part of the experiment is designed to estimate the i13DM capability in reducing the power consumption of the laptop while connected to the electricity grid using AC adapter. For the second part of the experiment, for the purpose of investigating the effect of DR schedule on demand load, we perform a demo scenario of a DR event with a number of laptops located at the i13 - the Chair for Application and Middleware Systems of Technical University of Munich's Department of Informatics.

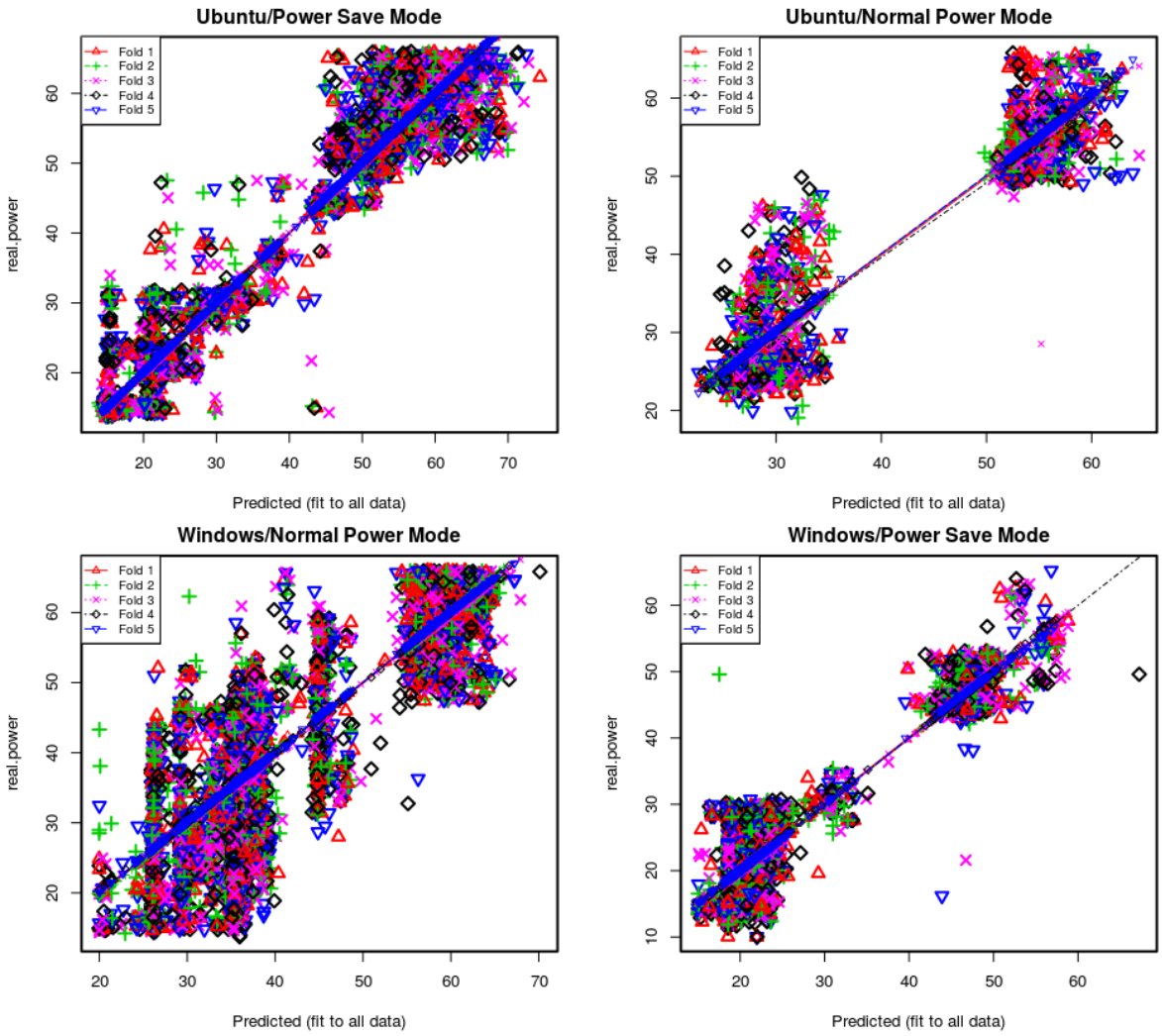


Figure 6.1: Power models cross-validations

Power Control Approach Evaluation

In chapter 5, we explain that i13DM utilizes built-in energy management features of OS to control the power consumptions of laptops. To determine the impacts of i13DM's power control techniques on demand load curtailment, we execute various workload on different platforms on different power consumption modes.

Power Control Evaluation Methodology

We use two workload generator applications to stress various components of a *Lenovo ThinkPad L540* laptop used as a testbed with specifications described at 5.1 with a Windows 10 and Ubuntu 16.04 installed side by side. To eliminate the impact of battery charging on the demand load, we execute all the workloads on the fully charged laptop. We execute five minutes long workload on both platform, three times while the laptop is running on normal power consumption mode and three times while running on power save mode which i13DM previously activated. During execution of workloads we monitor and record the real power consumption in watts and consumed energy in kWh through a connected *ZigBee Smart Energy Meter*[3].

On Ubuntu, we use *stress*[4] to create 5 minutes long workload spawning three workers to stress three out of four cores of CPU by executing *sqrt()*, spawning two workers spinning on *sync()* stressing IO, spawning two workers spinning on *write()/unlink()* with 1 GB files stressing disk and finally, spawning two workers performing *malloc()/free()* for stressing the memory. For generating workload on Windows, we make use of *HeavyLoad v3.4*[5] to generate 5 minutes long workload for stressing various components. *HeavyLoad* stresses all cores of CPU to the maximum by performing complex calculations, it contentiously writes large files on disk, allocates memory and stresses GPU by manipulating a 3D rendered graphic.

Workload Execution Results

Table 6.3 and 6.4 summarizes the workload execution results on Windows and Ubuntu respectively where *Mean Power* reports the arithmetic mean of sampled real power consumption in watts and *Consumed Energy* is the amount of electricity consumed over the period of workload execution. Moreover, both tables include the arithmetic mean of three sampled power and energy consumptions.

Finally, table 6.5 illustrates the percentage decrease of average real power and accumulated consumed energy when switching from normal power consumption mode to power save mode on Windows and Ubuntu. The power drop and energy drop values are calculated according to equations 6.3 and 6.4 from values presented in tables 6.3 and 6.4.

[3] `http://www.didactum-security.com/en/zigbee-wireless-monitoring/zigbee-voltage-monitoring/zigbee-smart-energy-meter-zbs-110-v2.html`

[4] `http://manpages.ubuntu.com/manpages/zesty/man1/stress.1.html`

[5] `http://www.jam-software.com/heavyload`

$$PowerDrop = \left(\frac{AveragePowerOnNormalMode - AveragePowerOnSaveMode}{AveragePowerOnNormalMode} \right) * 100 \tag{6.3}$$

$$EnergyDrop = \left(\frac{AverageEnergyOnNormalMode - AverageEnergyOnSaveMode}{AverageEnergyOnNormalMode} \right) * 100 \tag{6.4}$$

Power Mode	*Sample*	*Mean Power (watts)*	*Consumed Energy (kWh)*
Normal	1	40.28	0.003
	2	38.72	0.004
	3	37.66	0.004
Average Normal		38.89	0.0037
Save	1	28.58	0.003
	2	28.81	0.003
	3	28.40	0.002
Average Save		28.60	0.0027

Table 6.3: Power and energy consumption benchmark on Windows

Findings

As represented in table 6.5, i13DM is capable of decreasing the power consumption on Windows up to 26.64 percents and up to 6.95 percent on Ubuntu. The main reason for such a significantly better performance is the availability of more extensive built-in energy management features on Windows in comparison with Ubuntu which we explained in chapter 5.

Although the drop in real power consumption on Windows is much noticeable than Ubuntu, the reduction in total consumed energy is relatively in the same range for Windows and Ubuntu with 27.27 percents and 22.22 percent respectively. We believe

Power Mode	*Sample*	*Mean Power (watts)*	*Consumed Energy (kWh)*
	1	35.26	0.003
Normal	2	33.27	0.003
	3	33.98	0.003
Average Normal		34.17	0.003
	1	31.93	0.003
Save	2	30.29	0.002
	3	33.17	0.002
Average Save		31.80	0.0023

Table 6.4: Power and energy consumption benchmark on Ubuntu

OS	*Power Drop (%)*	*Energy Drop (%)*
Windows	26.46	27.27
Ubuntu	6.95	22.22

Table 6.5: Mean drop on power and energy consumption

the low accuracy of the measurement device and rather limited consumption of laptop over the short period of sampling is causing the low variation of both values. For this reason, we argue that the percentage drop in the power consumption is a better indicator of i13DM's ability for controlling electrical energy consumption of laptops.

DR Event Scheduling Scenario

To investigate the effectiveness and measure the performance of i13DR infrastructure, we design a scheduling scenario for performing an experimental demonstration of a DR event. For the purpose of this experiment, we develop a scheduling mock-up component for i13DRP where the administrator can schedule and manage multiple DR events as well as monitoring the participating laptops as they are joining and leaving the DR events. Because i13DR is not connected to any RES supplies while performing the experiments,

the mock-up component copies the behavior of a wind turbine integrated with the local power grid. Table 6.6 describes the scenario we develop for scheduling a DR event initiated by an administrator. For planning a DR event, the administrator first requires determining the location of the wind turbine. Afterward, for the sake of simplicity during the experiment, we presume that the wind turbine supplies every laptop located within a 1000 meters radius; consequently, when the production of turbine decreases, the demand load of laptops needs to decrease simultaneously.

Experimental Setup

We conducted the described DR scheduling scenario on three laptops running at the Chair for Application and Middleware Systems of Technical University of Munich's Department of Informatics with the hardware specification listed in table 6.8. All the participating laptops are fully charged and are connected to MEDAL power meters which monitor and records the power consumption during the experiments. We conducted the experiments five times according to the described scenario with a different number of participating laptops and DR event durations as stated in the table 6.7.

Figure 6.2 shows a snapshot of the mock-up component during execution of a DR event, where on the left side, the administrator provides the i13DRP with the location of the wind turbine and further required inputs. On the right side, he inspects several details regarding the progress of experiment including the exact time when scheduling started and finished. Also, he can observe the time all the participating laptops joined, an estimate of the amount of power saved by activating the power control on laptops. Finally, the panel also shows a list of laptops which were selected as candidates and received the new DR schedule as well as the exact time they joined the DR event.

Experiment Objectives and Evaluation Methodology

The main objective of this thesis is developing an effective real-time DR system with a significant impact on the real power consumption of laptops. As a result, the objectives of our experiments are proving the two claims that, first, our design is capable of real-time response to immediate changes of RES supply, second, with regard to the number of participating laptops and their power profiles we can significantly reduce the demand load of laptops.

In more details, to evaluate how our design responses to the immediate fluctuating changes

Scenario name	*DR schedule demonstration*
Participating actor instances	*i13DRP Administrator* *A number of laptops at i13 chair*
The flow of events	1. The administrator of i1DRP navigates to scheduling mock-up page. From there, he chooses the location of the wind turbine on the provided map, thereby selecting the longitude and latitude of the wind turbine site. Afterward, he specifies the possible reduction of wind turbine's electrical output in watts. The administrator is also required to provide the length of DR event in minutes. He also has the option to either start the DR event immediately or provide the start time at when the event begins. Once he provides all the required inputs, he hits the button to start scheduling for a DR event. 2. I13DRP starts the scheduling procedure, once it receives the DR event request. First, it queries all the currently online laptops to find the ones located within a 1000 meters radius of the provided location of the wind turbine. Next, for each retrieved laptop, it fetches the power consumption profiles from 20 minutes before the start time of DR event up to the start time. Then, it accumulates the reported difference of real power consumption in normal power mode and power save mode to have an estimate of the amount of energy one certain laptop can contribute to energy reduction. Finally, i13DRP creates a schedule with the provided start time, or if the start time is not specified, it schedules an event which immediately begins and lasts as defined by the administrator. When all the schedules are created, it submits them to Firebase, so the selected i13DM can download them. (To be continued)

Table 6.6: DR schedule experimental demonstration

of RES supply, for each experiment, we measure how long it takes from the moment a scheduling request for a DR event is initiated till the moment the schedule is published to Firebase. Furthermore, we measure the amount of time from the moment a schedule is published till the moment all participating laptops have received the schedules and activated the power control. Afterward, we show that our system performs both tasks

3. i13DMs are immediately notified of the new DR event through the bidirectional real-time connection to Firebase. Afterward, they fetch the new schedule and activate and deactivate the power control according to the new timetable. Moreover, they send a status code to Firebase announcing that they either joined or left the DR event.

4. while the experiment and DR events are carried out. the administrator observes the statistics of the experiment reported by i13DRP.

Table 6.6: DR schedule experimental demonstration

Laptop	*Vendor*	*OS*	*CPU*	*RAM*	*Screen size*	*Max AC Adapter Output*
Laptop 1	Lenovo ThinkPad L540	Windows 10	Intel Core i5	8 GB	15.6 inch	65 W
Laptop 2	Lenovo ThinkPad X230	Ubuntu 16.04	Intel Core i5	16 GB	12.5 inch	170 W
Laptop 3	Apple MacBook Pro	Windows 10	Intel Core i7	16 GB	15 inch	85 W

Table 6.7: Hardware specification of participating laptops

Index	*Number of Participating Laptops*	*Length of DR Event (inutes)*
1	3	10
2	3	10
3	3	5
4	3	5
5	3	5

Table 6.8: Experiments' details

within a specific time bound which in our case is one second. We also present that how the response time changes for a different number of participating laptops.

For evaluating the total impact of participating laptops on curtailing the demand load, we obtain the estimated demand reduction according to the laptops' power profiles. After that, we compare the calculated estimates with the real power reduction, provided by the readings of MEDAL power meter. Finally, we investigate the accuracy of our estimation.

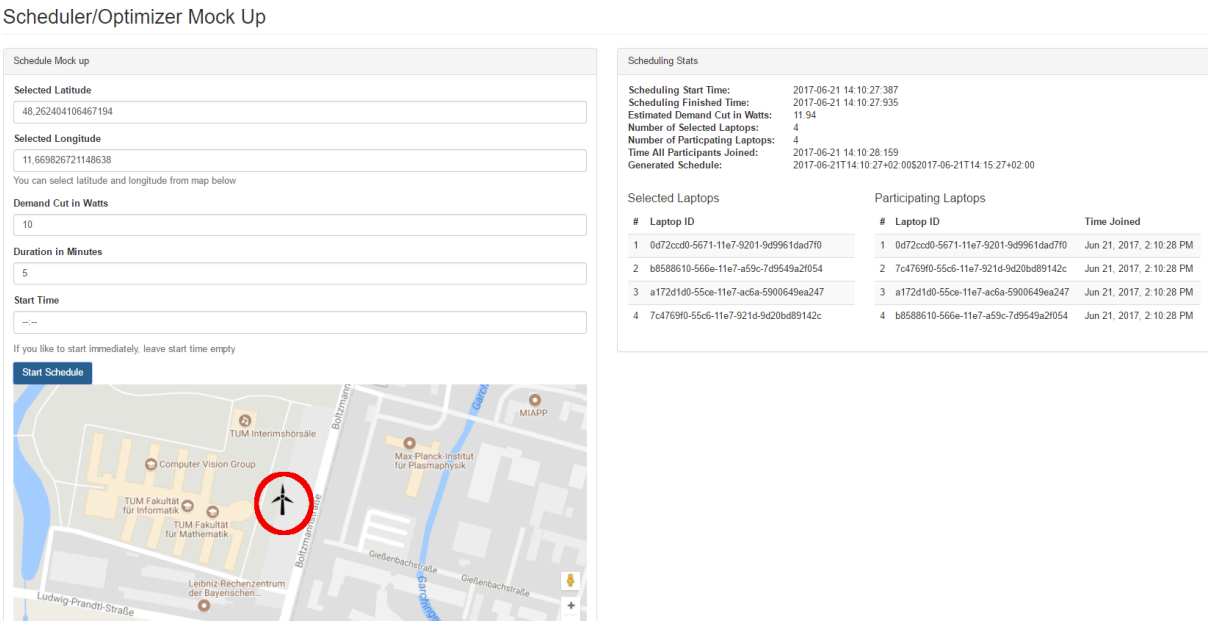


Figure 6.2: DR event scheduler mock-up

Results

As stated, we conducted five experiments, initiating five different DR events. Each time we measure a number of metrics required for evaluating the effectiveness and performance of i13DR infrastructure. Table 6.9 and 6.10 summarize the estimated demand load reduction in watts according to the laptops' power profiles as well as the observed reduction measured by MEDAL power meters. To measure the amount of load reduction, we calculated the arithmetic mean and median of the measured power consumption for three minutes before and three minutes after the activation of power control. The reason for including the measured median power consumption, besides the measured mean, is reducing the effect of outliers. However, based on the results, the difference between median and arithmetic mean is negligible.

Moreover, table 6.11 illustrates, two columns of time measurements. First, the amount of time it takes from the moment i13DRP receives a scheduling request from administrator till the point all the candidate laptops are selected and the schedules are created and published to Firebase. Second, it shows the period from the moment that all schedules are posted to the Firebase till the point all participating laptops have downloaded and activated the power control mode.

Finally, figures 6.3, 6.4 and 6.5 illustrates the demand load plots for the participating laptops during the execution of each experiment, where the dotted line represent the

Index	*Estimated Reduction (watts)*	*Mean Measured Reduction (watts)*	*Mean Measured Reduction (%))*
1	9.48	9.75	8.9
2	8.41	22.21	24.7
3	9.75	15.35	18.5
4	10.04	8.50	10.8
5	6.93	15.34	26
Average	8.92	14.23	17.80

Table 6.9: Summary of experiments' mean real power reductions

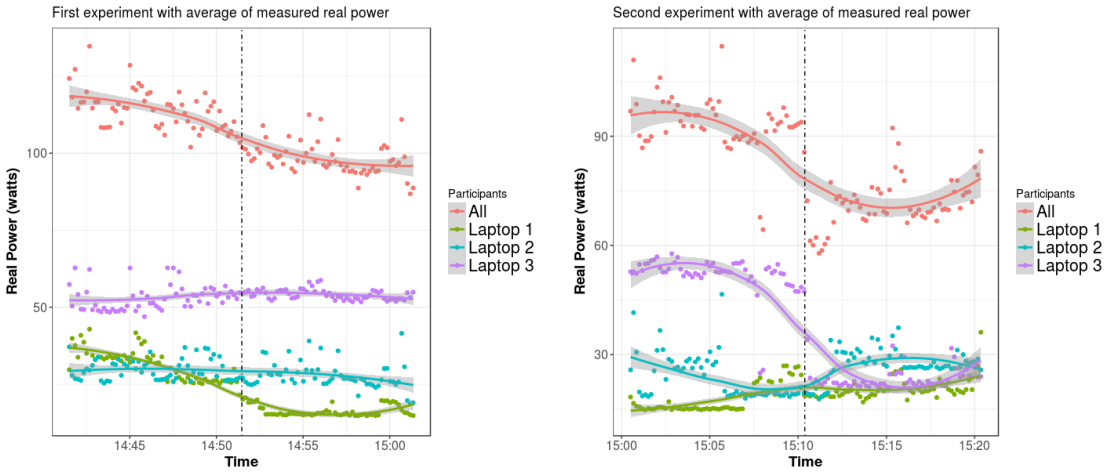
Index	*Estimated Reduction (watts)*	*Median Measured Reduction (watts)*	*Median Measured Reduction (%))*
1	9.48	9.22	8.5
2	8.41	21.37	23.7
3	9.75	14.80	18
4	10.04	9.28	11.8
5	6.93	15.21	26
Average	8.92	13.98	17.61

Table 6.10: Summary of experiments' median real power reductions

Index	*Duration of Scheduling (ms)*	*Length of All Laptops Participation (ms)*	*Total Time (ms)*
1	654	103	757
2	529	181	710
3	726	217	943
4	682	184	866
5	698	135	833
Average	658	164	822

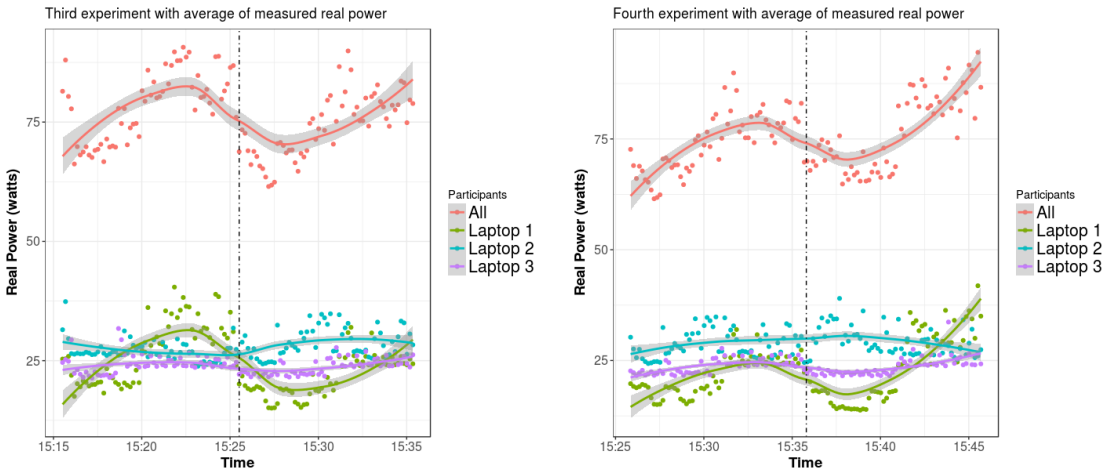
Table 6.11: Summary of experiments' scheduling operation's length

moment power control is activated. Every plot includes the demand load of every laptop separately, named as *Laptop 1*, *Laptop 2* and *Laptop 3* as well as the accumulated demand loads of all laptops, called as *All*. Each plot demonstrates the demand load for ten minutes before and after the activation of power control independent of the length of DR event.



(a) First DR event **(b)** Second DR event

Figure 6.3: Demand loads of first and second DR events

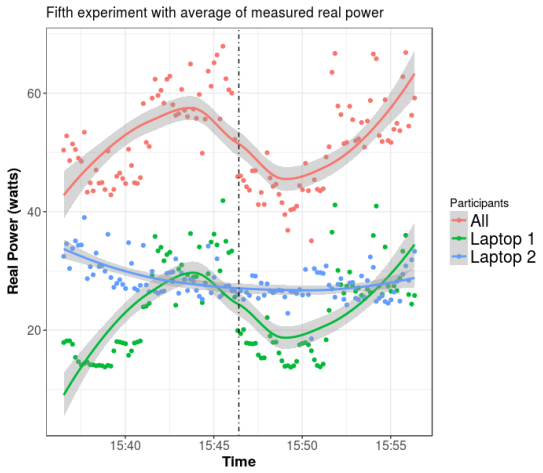


(a) Third DR event **(b)** Fourth DR event

Figure 6.4: Demand loads of third and fourth DR events

Findings

Based on the presented results, we verify that i13DR is capable of reducing the real power consumption of a group of laptops in a matter of milliseconds. Furthermore, table 6.12 summarizes the accuracy levels of the estimated demand reduction against the arithmetic mean and median of measured power consumption reduction, which we calculated according to the 6.2 and 6.1 equations. In general, we estimated an 8.92 watts demand reduction, and we measured a 14.23 and a 13.98 watts according to mean and median of power meter readings, indicating the effectiveness of our design. However, the



(a) Fifth DR events

Figure 6.5: Demand loads of fifth DR event

accuracy of our predictions is relatively low which we discuss the reasons in next section.

	Min/Max Accuracy (%)	*MAPE (%)*
Arithmetic Mean	66	72
Median	68	67

Table 6.12: Demand cut's estimation accuracy

According to the measured time in table 6.11, we observe that i13DR is capable of selecting the participating laptops, scheduling a DR event and activating the power control in less than a second. The average amount of time required for selecting candidate laptops, estimating their offered demand reduction and finally creating and publishing DR schedules is 658 milliseconds. On the other side, the average amount of time required for all the participating i13DMs to activate power controls and join the DR event is 164 milliseconds, pointing out that time required for creating a DR schedule is significantly larger than the necessary time for laptops to join a DR event.

An interesting side effect of power control activation is the brightness reduction of any connected stand-alone screens which subsequently reduces the demand load on the electricity grid. However, since investigating the power consumption of such components are not the focus of this thesis we do not include the results here.

Discussion

Our findings show that the designed i13DR is capable of successfully scheduling and performing a DR event targeting a group of laptops. However, they also highlight some remarks and potentials pitfalls of our design which we are explaining in the following section.

Although, on the average, the measured demand load reduction exceeds the estimated reduction, the accuracy of our estimation is rather low with 66 percent and 68 percent against the arithmetic mean and median of the measured power respectively. The main reason for this rather low accuracy is the low accuracy of deployed energy models used for creating the power consumption profiles. All participating laptops make use of identical power models for estimating the power consumption, which is fitted based on the data collected from a specific laptop (Laptop 1). Even though the power model yields relatively high accuracy for predicting real power consumption on Laptop 1, it fails to measure the power on other laptops accurately. Therefore, we conclude that it is significantly important to take different hardware specification into consideration when fitting the energy consumption models. We ignore this factor in our design because of the reason that it requires additional measurement devices installed on the demand side for collecting necessary data for constructing highly accurate models. These devices are a financial burden for demand side participates. Therefore and because one of our primary objectives is zero initial costs for participants, we sacrifice the accuracy for the sake of reaching our financial goals.

As described in section 6.3, the used power control techniques, offer 26.46 percent and 6.95 percent reduction on Windows and Ubuntu respectively according to the workloads we executed on the testbed laptop. For this reason and because we used two Windows laptops and one Ubuntu laptop during the experiments, we expect to observe a 19.96 percent reduction on average which we calculate according to the arithmetic mean of two instances of 26.46 percent reduction and one instance of 6.96 percent reduction. However, we observe a 17.80 percent and a 17.61 percent reduction according to the arithmetic mean of measured power consumption reductions, indicating roughly 89 percent accurate estimate of the effectiveness of energy control approaches.

During the experiments, we observe that i13DR is capable of fast responses to the immediate changes of RES with an average delay of 822 milliseconds from the moment a DR event is requested till the moment all participating laptops joined the DR event. The results also imply that i13DR significantly spends more time on selecting the participating laptops and scheduling the DR events than bringing all the laptops to the

DR event. The main cause for this is the quality of the mock-up component we develop for this experiments. The mock-ups are created with AngularJS and are executed on administrator's Google Chrome web browser. Therefore, the performance is limited to administrator's laptop's resources (Laptop 1) as well as the native applications running on administrator's laptops besides the web browser. However, we did not execute any performance heavy applications during the experiments. Finally, we suggest developing the scheduling components as a server application which can offer to schedule a DR event as a service to the administrator.

Limitations

In general, according to our evaluations, i13DR is arguably capable of performing effective DR events. However, since one of our primary objectives is eliminating the initial cost for demand side participants, we depend heavily on a pure software approach for implementing all required demand side functionalities. As a result, our design yields low estimation accuracy and subsequently, low reliability, highlighting the trade-off between cost and precision, where decreasing initial cost reduces the accuracy.

Finally, we notice that the amount of energy reduction that laptops can contribute to the power grids using operating systems' built-in power management features is relatively low. For that reason, trying to compensate for a substantial loss of RES supplies, solely based on a limited number of participating laptops may not be feasible.

Chapter 7

Summary

In the previous chapters, we extensively described and discussed our approach to a Demand Response Infrastructure targeting laptops. In the following chapter, we present the final remarks of our work. First, we mention the status of the thesis including the realized goals of the i13DR as well as the open goals which the system have not reached over the course of this thesis. Then, we present a final overview and concluding remarks of i13DR and finally, we provide and outlook about the future work.

Status

Over the course of this thesis, we successfully designed and implemented most of the realized requirements described in chapter 4. i13DR, consisting of two main components of i13DM and i13DRP, effectively organizes a DR event for a number of laptops in response to immediate changes of RES supply to curtail the demand load of participating laptops.

On the DR provider side of the system, i13DRP successfully maintains the registered laptops, their power consumption, and location profiles. However, it lacks a proper scheduling component for creating DR events based on the optimization techniques meant for minimizing the energy consumption on the demand side. For the sake of demonstration, we created a mockup scheduling component which performs simple scheduling tasks based on the current location of currently online laptops as well as some parts of power profiles, without taking the extensive amount of information we gathered in location and energy consumption profiles into consideration.

On the other hand, i13DR successfully performs the essential activities for executing

effective DR events, such as monitoring the actual power consumption, power control techniques and communicating with i13DRP via real-time connections. Furthermore, it performs the specified location and energy consumption profiling based on the deployed power models as described in our requirements analysis. It also executes the activities for maintaining the health and life cycle of the application such as automatic updates and system crash reports. However, the main flaw of i13DM is the low accuracy of real-time power consumption due to the low accuracy of deployed models.

In summary, we verify the design and implementation of the i13DR infrastructure to be effective. Our approach addresses several concerns about responsively, scalability, security, and affordability, despite the rather low accuracy and reliability due to the pure software approach of our design. However, we argue that low accuracy is the trade-off between eliminating the demand side costs and reliability, and because lowering the initial cost is a primary constraint for us, we trade the accuracy for dropping the costs.

Conclusion

In this thesis, we presented a novel approach to design and implementation of a Demand Response Infrastructure for laptops based on combining two lines of research on existing DR infrastructures and high-level software-oriented power consumption modeling techniques. Furthermore, in chapter 2 we explained the tools and technologies required for understating the flow of the thesis including the concept of Demand Response and its essential components as well as the goals and techniques of constructing and utilizing power consumption models. Afterward, in chapter 3, we reviewed the previous studies in the area of DR and energy modeling, emphasizing on our scientific contributions and their contrast to the existing work. We also extensively discussed the functional and nonfunctional requirements and expected use cases of our design. We explained that we treat initial cost reduction of the demand side consumers as a first order requirement and we facilitated integrating fluctuating RES supplies to the power grid. Furthermore, we addressed several issues raised from designing such a complex system including but not limited to reliability, responsively, and security. Finally, we described the ways we decompose i13DR into smaller subsystems and components which are more understandable and easier to implement. Moreover, we introduced the technologies and frameworks used for implementing i13DR.

Finally, to evaluate the effectiveness and performance of i13DR, we conducted a number of experiments, and we evaluated the resulting empirical measurements which we discussed

in chapter 6. We verified that the constructed power models predict the current energy consumption in real-time with accuracy up to 95 percents on Ubuntu and 85 percent on Windows. Furthermore, we observe that i13DR is able of performing effective DR events in real-time with minimal delay. However, the accuracy of our estimation of demand load reduction is about 67 percents indicating rather low reliability. The main reason for the low accuracy is our pure software approach which sacrifices high accuracy for the sake of eliminating initial DR costs.

Future Work

Over the course of this thesis, we realized a few areas remained out of our reach, which we offer here as parts of our future work. First, a large number of laptops are running on *MacOS*, and i13DM is not compatible with this platform. Therefore, to extend the reach of i13DR to all sort of participants, we suggest developing a new version of the i13DM executable on MacOS. The cross-platform Electron framework that we use for developing i13DM is capable of building and packaging MacOS ready applications. However, further research on power monitoring and controlling on MacOS needs to be done.

Moreover, we suggest performing improvement and optimization on some aspects of our design, including power modeling techniques, energy consumption control methods and DR event scheduling and optimization approaches. Our approach toward energy modeling is solely based on linear regression techniques. However, there are several other machine learning approaches which worth further studies, such as *k-nearest neighbors algorithm*. Furthermore, we require to implement a production-ready scheduler and optimizer for minimizing the electrical load consumption on i13DRP, preferably based on optimization algorithms such as *Mixed Integer Linear Programming*, *Convex Optimization Problem* and *Particle Swarm Optimization*. Finally, as previously described, our power control approaches heavily depend on built-in power management features of the operating system. Although Windows offers extensive features in that regard, Ubuntu suffers from insufficient power management options. Therefore, we suggest investigating further on different techniques for managing power consumption on Ubuntu.

Appendices

Appendix A

Appendix

Snapshots of i13DR

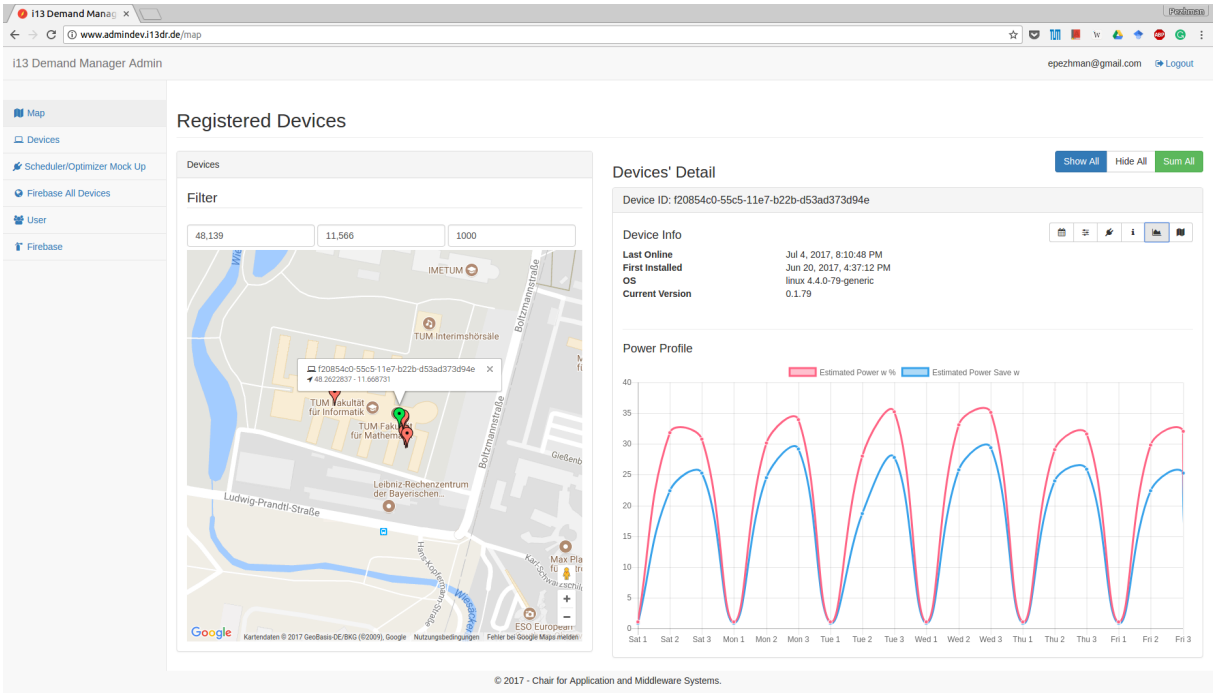


Figure A.1: A snapshot of i13DRP administrator panel with power profile of one laptop

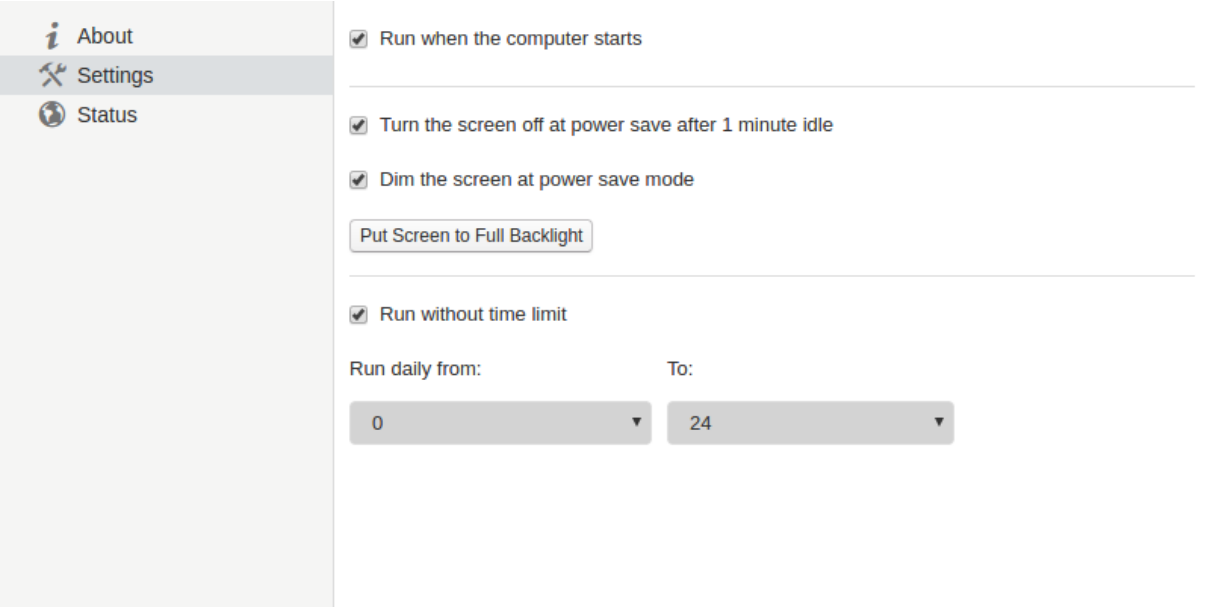


Figure A.2: A snapshot of i13DM's settings window on Ubuntu

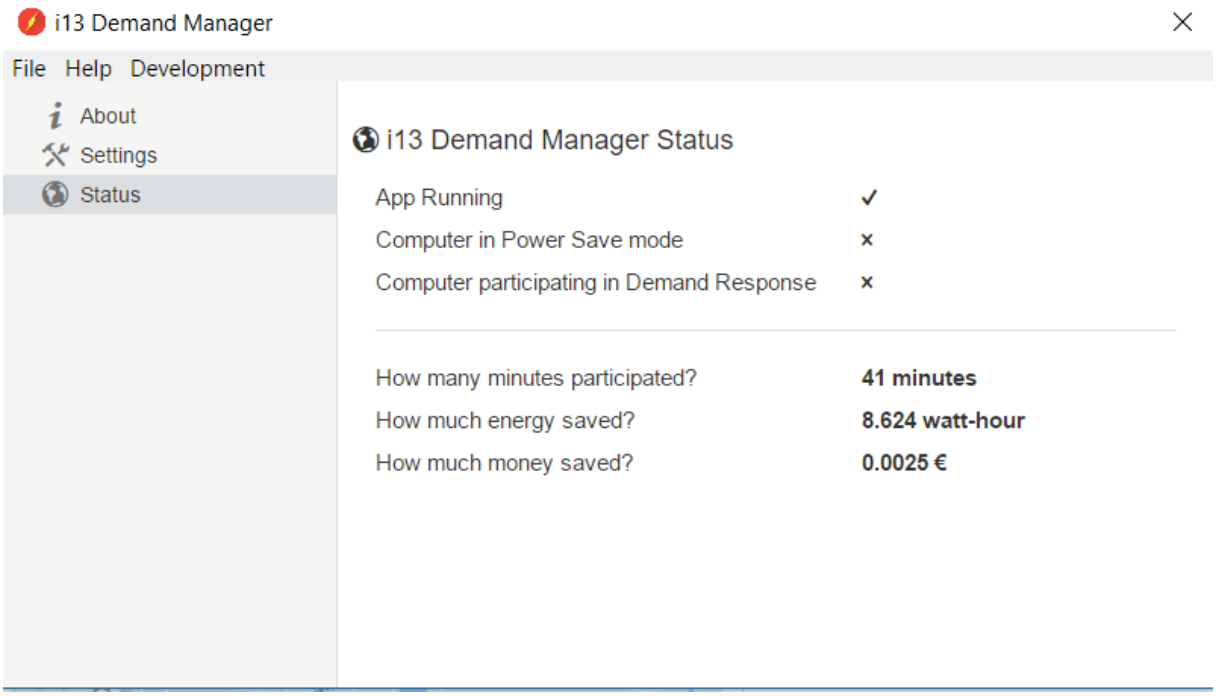


Figure A.3: A snapshot of i13DM's status window on Windows

All Subsets Regressions of Power-related Data

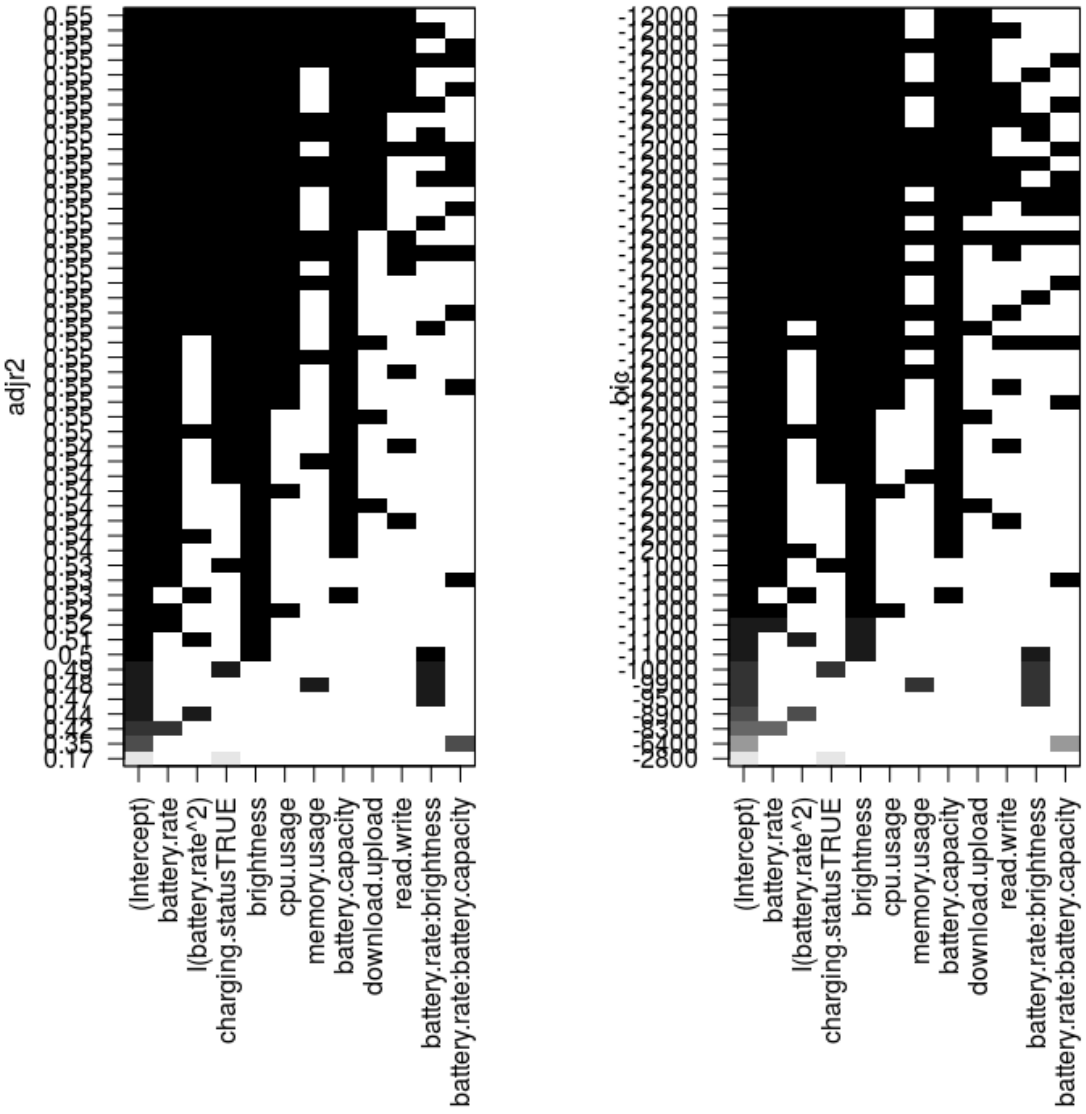


Figure A.4: All subsets regressions for Windows in normal power mode

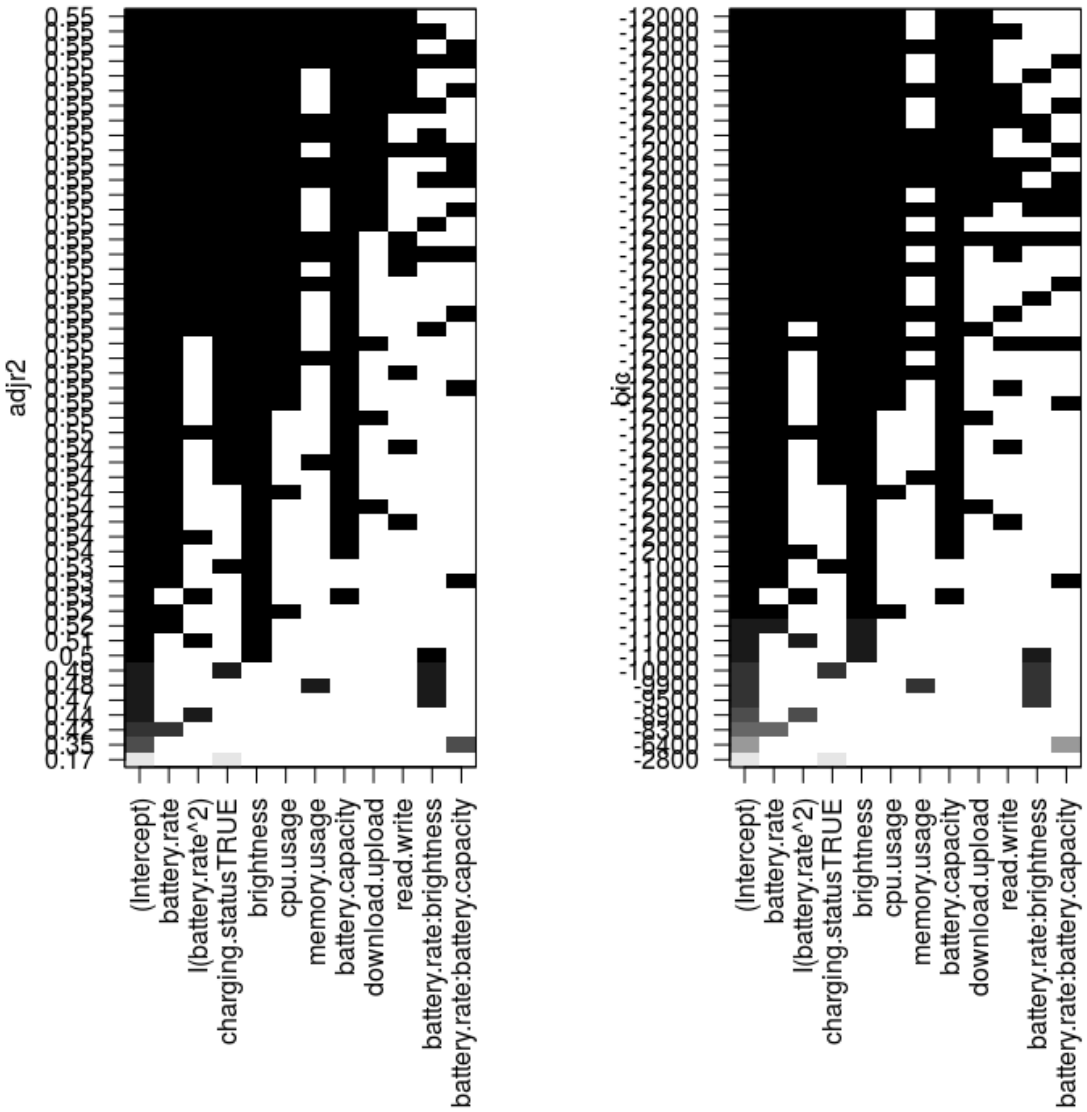


Figure A.5: All subsets regressions for Windows in power save mode

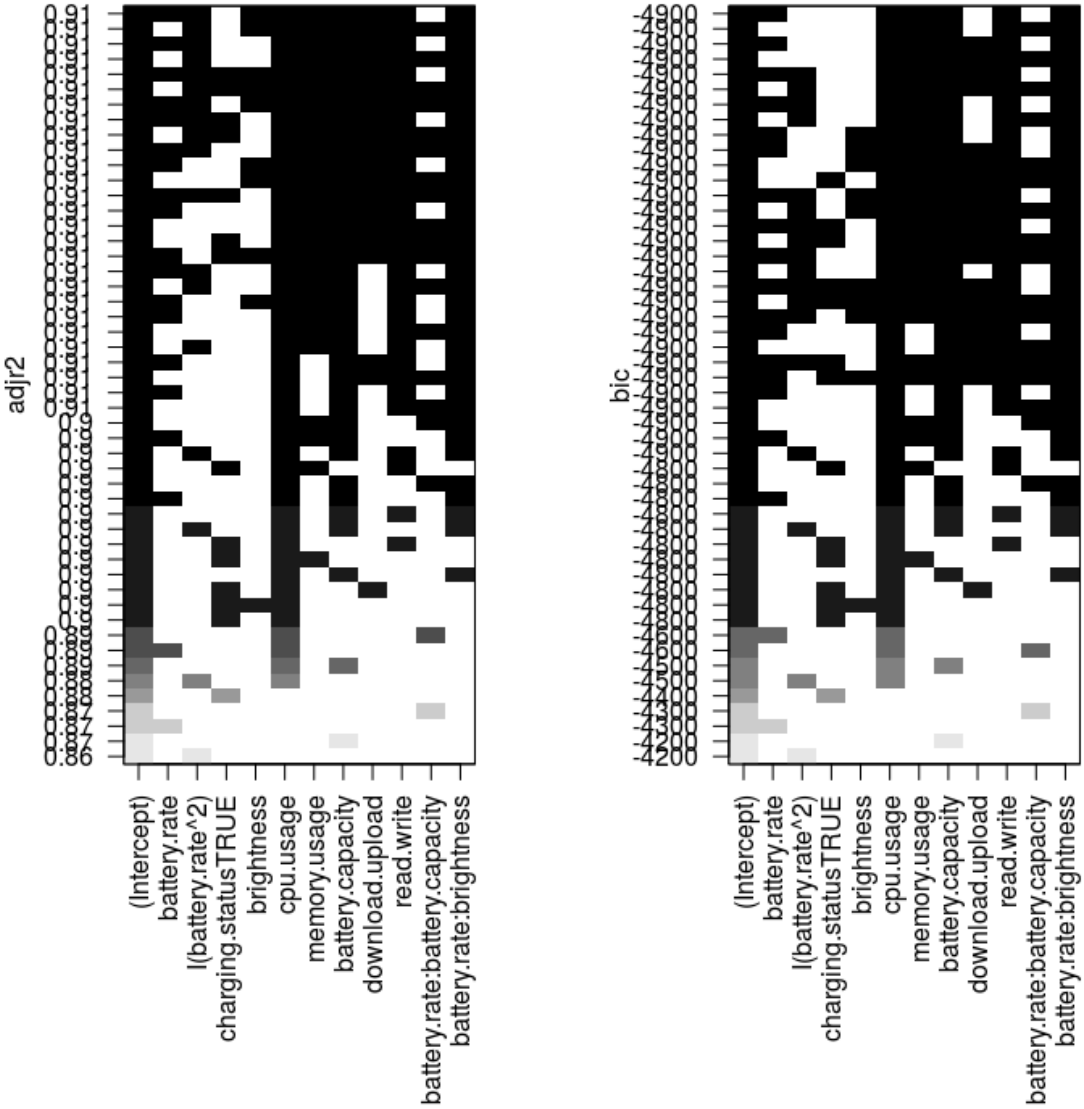


Figure A.6: All subsets regressions for Ubuntu in normal power mode

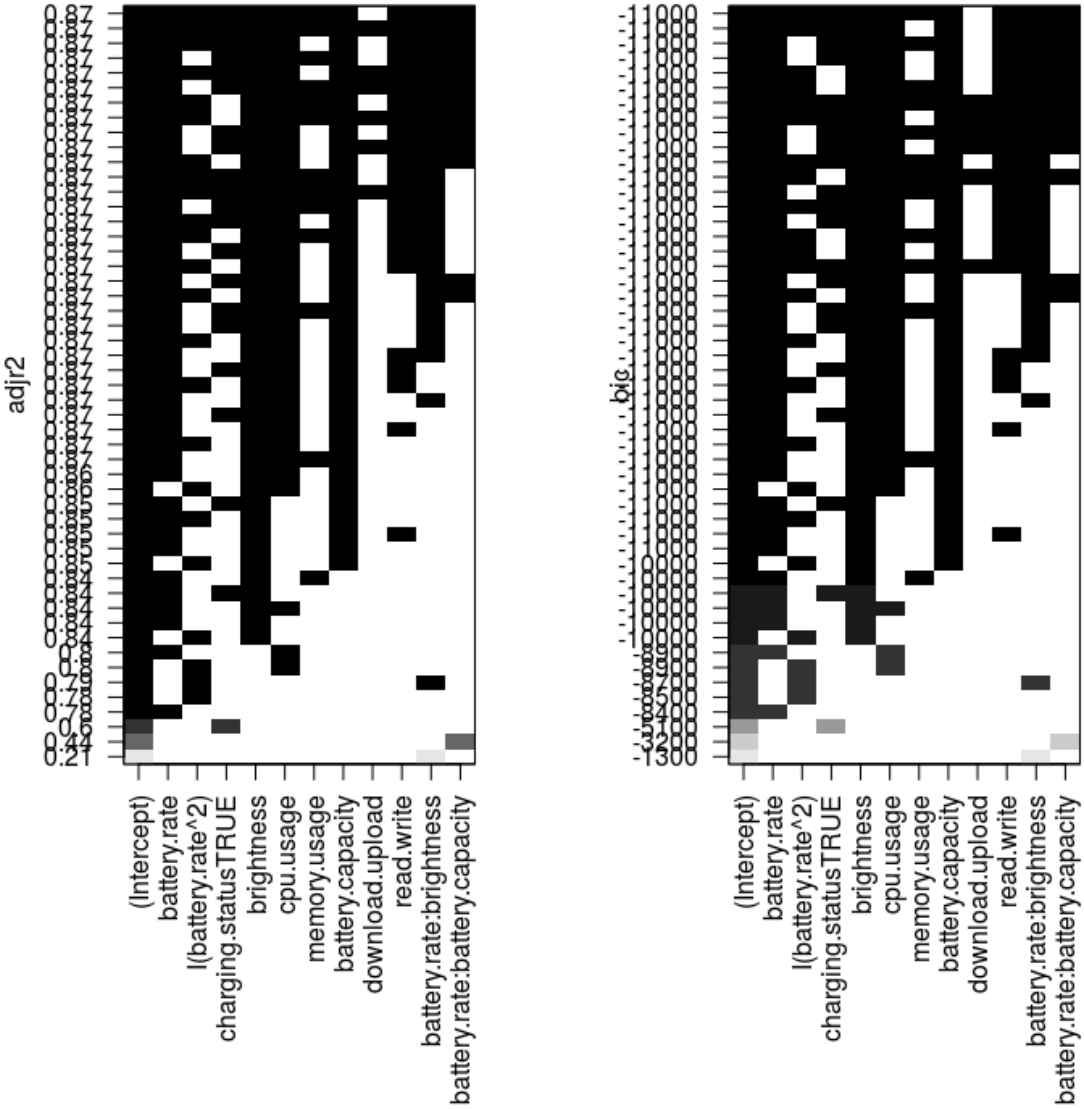


Figure A.7: All subsets regressions for Ubuntu in power save mode

Power Models Summary

```
Call:
lm(formula = real.power ~ battery.rate + I(battery.rate^2) +
    charging.status + battery.rate:brightness + battery.rate:battery.capacity +
    cpu.usage + memory.usage + battery.capacity + download.upload +
    read.write, data = train.win.nr)

Residuals:
    Min      1Q  Median      3Q     Max
-22.479  -4.886   1.269   5.251  32.207

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                    28.9554920  0.4478350  64.657  < 2e-16 ***
battery.rate                    1.9645655  0.0924785  21.243  < 2e-16 ***
I(battery.rate^2)              -0.0449502  0.0036351 -12.366  < 2e-16 ***
charging.statusTRUE            -8.9212710  0.3985847 -22.382  < 2e-16 ***
cpu.usage                       0.0099988  0.0026797   3.731 0.000191 ***
memory.usage                    0.0552145  0.0085994   6.421 1.41e-10 ***
battery.capacity                0.0339203  0.0059349   5.715 1.12e-08 ***
download.upload                 0.0244233  0.0040584   6.018 1.82e-09 ***
read.write                      0.0075262  0.0074644   1.008 0.313340
battery.rate:brightness         0.0114965  0.0004941  23.267  < 2e-16 ***
battery.rate:battery.capacity -0.0086603  0.0005941 -14.577  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.366 on 12087 degrees of freedom
Multiple R-squared:  0.5223,    Adjusted R-squared:  0.5219
F-statistic:  1321 on 10 and 12087 DF,  p-value: < 2.2e-16
```

Figure A.8: Summary of power model for Windows on normal power mode

```
Call:
lm(formula = real.power ~ battery.rate + I(battery.rate^2) +
    charging.status + battery.rate:brightness + battery.rate:battery.capacity +
    cpu.usage + memory.usage + battery.capacity + download.upload +
    read.write, data = train.win.sv)

Residuals:
    Min      1Q  Median      3Q     Max
-25.006  -3.705  -0.519   3.938  31.998

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                    18.6205324  0.1320492 141.012  < 2e-16 ***
battery.rate                    1.0798484  0.1052582  10.259  < 2e-16 ***
I(battery.rate^2)               0.0093846  0.0037028   2.534  0.01130 *
charging.statusTRUE            -3.4060935  0.5609782  -6.072 1.37e-09 ***
cpu.usage                       0.0474625  0.0022351  21.235  < 2e-16 ***
memory.usage                    0.0315672  0.0042431   7.440 1.21e-13 ***
battery.capacity               -0.0243552  0.0027851  -8.745  < 2e-16 ***
download.upload                 0.0342541  0.0118742   2.885  0.00394 **
read.write                      0.0865313  0.0115149   7.515 6.86e-14 ***
battery.rate:brightness         0.0058695  0.0003677  15.965  < 2e-16 ***
battery.rate:battery.capacity -0.0027629  0.0002496 -11.070  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.648 on 4441 degrees of freedom
Multiple R-squared:  0.8248,    Adjusted R-squared:  0.8244
F-statistic:  2091 on 10 and 4441 DF,  p-value: < 2.2e-16
```

Figure A.9: Summary of power model for Windows on power save mode

```
Call:
lm(formula = real.power ~ battery.rate + I(battery.rate^2) +
    charging.status + battery.rate:battery.capacity + cpu.usage +
    memory.usage + battery.capacity + download.upload + read.write,
    data = train.ub.nr)

Residuals:
     Min       1Q   Median       3Q      Max
-13.4318  -2.4211  -0.5811   1.3556  17.7642

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 76.168538   6.792201  11.214  < 2e-16 ***
battery.rate                -1.878025   0.409972  -4.581 4.98e-06 ***
I(battery.rate^2)            0.021265   0.013011   1.634   0.1024
charging.statusTRUE          1.212296   5.025478   0.241   0.8094
cpu.usage                    0.141621   0.007467  18.967  < 2e-16 ***
memory.usage                 0.080845   0.019157   4.220 2.57e-05 ***
battery.capacity            -0.566635   0.067597  -8.383  < 2e-16 ***
download.upload              0.011157   0.005050   2.209   0.0273 *
read.write                  -0.046520   0.008782  -5.298 1.33e-07 ***
battery.rate:battery.capacity  0.025717   0.003221   7.984 2.61e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.25 on 1663 degrees of freedom
Multiple R-squared:  0.9065,    Adjusted R-squared:  0.906
F-statistic:  1791 on 9 and 1663 DF,  p-value: < 2.2e-16
```

Figure A.10: Summary of power model for Ubuntu on normal power mode

```
Call:
lm(formula = real.power ~ battery.rate + I(battery.rate^2) +
    charging.status + battery.rate:battery.capacity + cpu.usage +
    memory.usage + battery.capacity + download.upload + read.write,
    data = train.ub.sv)

Residuals:
     Min       1Q   Median       3Q      Max
-31.0842  -0.9185  -0.2756   0.4500  24.7904

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 1.142e+02  2.212e+00  51.621  < 2e-16 ***
battery.rate               -2.307e+00  1.103e-01 -20.920  < 2e-16 ***
I(battery.rate^2)           7.617e-03  9.715e-04   7.840 4.90e-15 ***
charging.statusTRUE        -8.171e-01  1.215e-01  -6.724 1.86e-11 ***
cpu.usage                   2.617e-01  2.785e-03  93.971  < 2e-16 ***
memory.usage               -3.303e-01  1.919e-02 -17.215  < 2e-16 ***
battery.capacity           -8.679e-01  2.149e-02 -40.388  < 2e-16 ***
download.upload             1.964e-02  2.410e-03   8.152 3.97e-16 ***
read.write                 -1.188e-01  1.039e-02 -11.433  < 2e-16 ***
battery.rate:battery.capacity  3.274e-02  9.831e-04  33.300  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.849 on 11263 degrees of freedom
Multiple R-squared:  0.968,     Adjusted R-squared:  0.968
F-statistic: 3.785e+04 on 9 and 11263 DF,  p-value: < 2.2e-16
```

Figure A.11: Summary of power model for Ubuntu on power save mode

Bibliography

[1] W. Commons, "The power triangle," 2011.

[2] B. Yang, K. V. Katsaros, W. K. Chai, and G. Pavlou, "Cost-efficient low latency communication infrastructure for synchrophasor applications in smart grids," *IEEE Systems Journal*, vol. PP, no. 99, pp. 01–11, 2017.

[3] P. Siano, "Demand response and smart grids—a survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461 – 478, 2014.

[4] J. S. Vardakas, N. Zorba, and C. V. Verikoukis, "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 152–178, Firstquarter 2015.

[5] "Federal energy regulatory commission assessment of demand response and advanced metering." `https://www.ferc.gov/legal/staff-reports/12-08-demand-response.pdf`, 2008. Accessed: 2017-05-13.

[6] C. Cecati, C. Citro, A. Piccolo, and P. Siano, "Smart operation of wind turbines and diesel generators according to economic criteria," *IEEE Transactions on Industrial Electronics*, vol. 58, pp. 4514–4525, Oct 2011.

[7] C. Cecati, C. Citro, and P. Siano, "Combined operations of renewable energy systems and responsive demand in a smart grid," *IEEE Transactions on Sustainable Energy*, vol. 2, pp. 468–476, Oct 2011.

[8] C. Joe-Wong, S. Sen, S. Ha, and M. Chiang, "Optimized day-ahead pricing for smart grids with device-specific scheduling flexibility," *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 1075–1085, July 2012.

[9] E. Shayesteh, A. Yousefi, and M. P. Moghaddam, "A probabilistic risk-based approach for spinning reserve provision using day-ahead demand response program," *Energy*, vol. 35, no. 5, pp. 1908 – 1915, 2010.

[10] C. A. Goldman, N. C. Hopper, R. Bharvirkar, B. Neenan, R. Boisvert, P. Cappers, D. Pratt, and K. Butkins, "Customer strategies for responding to day-ahead market hourly electricity pricing," p. 107, 08/2005 2005.

[11] J. L. Berral, R. Gavalda, and J. Torres, "Adaptive scheduling on power-aware managed data-centers using machine learning," in *2011 IEEE/ACM 12th International Conference on Grid Computing*, pp. 66–73, Sept 2011.

[12] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 732–794, Firstquarter 2016.

[13] J. Davis, S. Rivoire, M. Goldszmidt, and E. K. Ardestani, "No hardware required: Building and validating composable highly accurate os-based power models," tech. rep., July 2011.

[14] G. Xiong, C. Chen, S. Kishore, and A. Yener, "Smart (in-home) power scheduling for demand response on the smart grid," in *ISGT 2011*, pp. 1–7, Jan 2011.

[15] Z. Zhao, W. C. Lee, Y. Shin, and K. B. Song, "An optimal power scheduling method for demand response in home energy management system," *IEEE Transactions on Smart Grid*, vol. 4, pp. 1391–1400, Sept 2013.

[16] F. Saffre and R. Gedge, "Demand-side management for the smart grid," in *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*, pp. 300–303, April 2010.

[17] J. Medina, N. Muller, and I. Roytelman, "Demand response and distribution grid operations: Opportunities and challenges," *IEEE Transactions on Smart Grid*, vol. 1, pp. 193–198, Sept 2010.

[18] N. Motegi, M. A. Piette, D. S. Watson, S. Kiliccote, and P. Xu, "Introduction to commercial building control strategies and techniques for demand response – appendices, report, may 1, 2007; berkeley," *California. (digital.library.unt.edu/ark:/*, vol. 6753, July 2017.

[19] P. Samadi, A. H. Mohsenian-Rad, R. Schober, V. W. S. Wong, and J. Jatskevich, "Optimal real-time pricing algorithm based on utility maximization for smart grid," in *2010 First IEEE International Conference on Smart Grid Communications*, pp. 415–420, Oct 2010.

[20] T. Kriechbaumer, A. Ul Haq, M. Kahl, and H.-A. Jacobsen, "Medal: A cost-effective high-frequency energy data acquisition system for electrical appliances," in *Proceedings of the Eighth International Conference on Future Energy Systems*, e-Energy '17, (New York, NY, USA), pp. 216–221, ACM, 2017.

[21] "Germany's energy consumption and power mix in charts." `https://www.cleanenergywire.org/factsheets/germanys-energy-consumption-and-power-mix-charts`. Accessed: 2017-07-03.

[22] J. Aghaei and M.-I. Alizadeh, "Demand response in smart electricity grids equipped with renewable energy sources: A review," *Renewable and Sustainable Energy Reviews*, vol. 18, pp. 64 – 72, 2013.

[23] M. H. Albadi and E. F. El-Saadany, "Demand response in electricity markets: An overview," in *2007 IEEE Power Engineering Society General Meeting*, pp. 1–5, June 2007.

[24] S. Gottwalt, W. Ketter, C. Block, J. Collins, and C. Weinhardt, "Demand side management—a simulation of household behavior under variable prices," *Energy Policy*, vol. 39, no. 12, pp. 8163 – 8174, 2011. Clean Cooking Fuels and Technologies in Developing Economies.

[25] K. Herter, "Residential implementation of critical-peak pricing of electricity," *Energy Policy*, vol. 35, no. 4, pp. 2121 – 2130, 2007.

[26] S. Shao, T. Zhang, M. Pipattanasomporn, and S. Rahman, "Impact of tou rates on distribution load shapes in a smart grid with phev penetration," in *IEEE PES T D 2010*, pp. 1–6, April 2010.

[27] Q. Zhou, W. Guan, and W. Sun, "Impact of demand response contracts on load forecasting in a smart grid environment," in *2012 IEEE Power and Energy Society General Meeting*, pp. 1–4, July 2012.

[28] G. Webber, J. Warrington, S. Mariéthoz, and M. Morari, "Communication limitations in iterative real time pricing for power systems," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 445–450, Oct 2011.

[29] P. Luh, Y. Ho, and R. Muralidharan, "Load adaptive pricing: An emerging tool for electric utilities," *IEEE Transactions on Automatic Control*, vol. 27, pp. 320–329, Apr 1982.

[30] "Ausstattung privater haushalte mit informations- und kommunikationstechnik - deutschland." `https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsguetern/Tabellen/Infotechnik_D.html`. Accessed: 2017-07-03.

[31] N. Murthy, "Energy-agile laptops: Demand response of mobile plug loads using sensor/actuator networks," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 581–586, Nov 2012.

[32] "Integration of demand side management."

[33] A. Papavasiliou and S. S. Oren, "Coupling wind generators with deferrable loads," in *2008 IEEE Energy 2030 Conference*, pp. 1–7, Nov 2008.

[34] J. Rivera, M. Jergler, A. Stoimenov, C. Goebel, and H.-A. Jacobsen, "Using publish/subscribe middleware for distributed ev charging optimization," *Computer Science - Research and Development*, vol. 31, pp. 41–48, May 2016.

[35] M. Marwan, F. Kamel, and W. Xiang, "A demand-side response smart grid scheme to mitigate electrical peak demands and access renewable energy sources," in *48th Annual Conference of the Australian Solar Energy Society*, 2010.

[36] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, (Berkeley, CA, USA), pp. 3–3, USENIX Association, 2008.

[37] T. Kriechbaumer, "Optimisation and analysis of full-system power models for servers in data centres." Master's Thesis, 2015.

[38] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, (New York, NY, USA), pp. 335–348, ACM, 2011.

[39] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," 2006.

[40] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd ed., 2009.

[41] J. M. Carroll, ed., *Scenario-based Design: Envisioning Work and Technology in System Development*. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[42] "Unified modeling language." `https://en.wikipedia.org/wiki/Unified_Modeling_Language`. Accessed: 2017-07-04.