Self-Adaptation for Energy Efficiency in Software Systems

Fahimeh Alizadeh Moghaddam

Self-Adaptation for Energy Efficiency in Software Systems

2019

Self-Adaptation for Energy Efficiency in Software Systems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het
College voor Promoties ingestelde
commissie, in het openbaar te verdedigen
in de Agnietenkapel
op woensdag 17 april 2019, te 14.00 uur

door

Fahimeh Alizadeh Moghaddam

geboren te Teheran

Promotiecommissie:

Promotors:	Prof. dr. ir.	C.T.A.M de Laat	Universiteit van Amsterdam
	Prof. dr.	P. Lago	Vrije Universiteit Amsterdam
Copromotor:	Dr.	P. Grosso	Universiteit van Amsterdam
Overige leden:	Prof. dr.	H. Afsarmanesh	Universiteit van Amsterdam
	Prof. dr. ir.	A. Osseyran	Universiteit van Amsterdam
	Dr.	G.A. Lewis	Carnegie Mellon University
	Prof. dr. ir.	H.E. Bal	Vrije Universiteit Amsterdam
	Prof. dr. ir.	P. Avgeriou	Rijksuniversiteit Groningen
	Prof. dr.	S. Klous	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

SIKS Dissertation Series No. 2019-09
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover depicts Shushtar Historical Hydraulic System, an ancient engineering masterpiece for optimal use of water.

Cover image credits to Razie Amirian

ISBN: 978-94-028-1423-1
NUR code: 986

Dedicated to Sepehr and Ava

Contents

1 Introduction

1.1 The urgency for energy efficiency

Information and communication technology (ICT) is embedded in human lives more than ever. That has resulted in a growing demand for energy in the ICT sector. The increasing number of mobile users and desktop users confirms this new trend. Statistics show that the number of ICT users reached 3.6 billion in 2015. On top of that, the average time that each user spends every day to use electronic devices, which is around 5.6 hours [54], almost doubled from 2008 to 2015. In 2012, ICT alone consumed about 900TWh of electricity, which is higher than the electricity consumption of Japan in the same year [1, 244]. Researchers estimated that ICT will account for about 2.5% of the total energy consumption by 2020 [66].

Figure 1.1 represents the breakdown of the electricity consumption for the main categories of the ICT sector in the years 2012 and 2017 [64]. The categories that contribute the most to the total energy consumption of the ICT sector are namely communication networks, data centers, personal devices and infrastructure manufacturing. The figure highlights the trend on the growth of contribution of data centers and communication networks over the years. This confirms the fact that many service providers have moved to the cloud.

Today, mobile devices are acting as interfaces to the cloud. Mobile users benefit from various software services running in the cloud data centers with their portable devices, which do not necessarily have high computational power. Higher energy efficiency is the target for both the cloud computing and the mobile computing fields. In mobile computing, the aim is to extend the battery life of the portable devices as they have limited energy stored in their batteries. On the other hand, the primary concern for the cloud is the amount of consumed energy. Cloud providers try to maximize the total energy efficiency of their data centers. For example, U.S. Environmental Protection Agency reported that the

cost of data centers alone was $4.5 billion in 2006 [45].

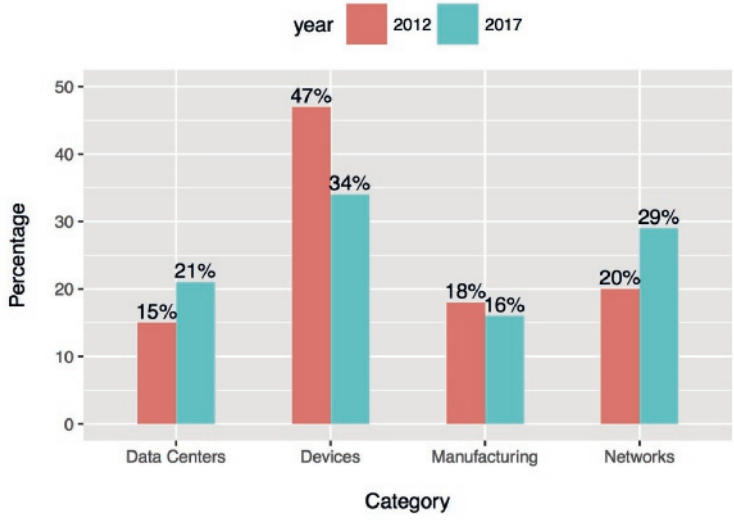


Figure 1.1: The energy consumption breakdown for the ICT sector in 2012 and 2017 based on [64]

Different solutions have been proposed regarding improving energy efficiency in various fields (including mobile computing and cloud computing). However, field-specific solutions do not always improve the total energy efficiency of systems that are composed of different technologies in different fields. Suppose we have a cyber-foraging system[1]. If we implement a solution that focuses only on the components in the mobile, then we can not necessarily globally ensure improvements for the system. Globally optimal solutions are more influential than locally optimal ones as they take the whole life-cycle of the target system into account.

1.2 The role of software architecture

Software defines how systems infrastructures should be utilized. Accordingly, in the case of inefficiencies in infrastructures utilization, one should investigate the running software. It is important to note that infrastructures themselves have a role in the energy efficiency of software systems, and many vendors and

[1]Cyber-foraging is a technique to extend the resource limitations of mobile devices. In cyber-foraging systems, mobile computing techniques and cloud computing techniques are combined either to perform computation-intensive tasks or to store large amounts of data [260]. More information is provided in Chapter 5.

researchers have introduced new techniques to save energy on hardware [125] [128]. However, the influence of inefficient software can spread to the whole (even energy efficient) infrastructure, which can negate the total efficiency level. Therefore, energy efficiency solutions should be applied to the software. Software architecture is the right instrument for this purpose because it can primarily carry over the design decisions that empower the software towards ensuring energy efficiency at runtime.

A wide range of studies approaches the problem of energy efficiency at the infrastructure level. For instance, [238, 240] and [173] focus on optimizing the routing strategies to minimize the energy consumption of network connections in data centers. In addition, dynamic voltage scaling (DVS) and vary-on/vary-off (VOVO) mechanisms are commonly deployed in server clusters [81, 120] and data center networks [223]. Although the provided solutions show improvements in energy efficiency, they are technology-specific and system-specific. These solutions remain disconnected from the software design aspects. Software design must embed the software with the intelligence to react to runtime changes regarding energy efficiency. Software must be able to deploy energy efficiency solutions, assess their impact on system qualities at runtime, and if needed, decide on infrastructural or architectural reconfiguration. In this thesis, we argue and ultimately demonstrate that enabling self-adaptability of software can help steering the infrastructure-level solutions to meet quality requirements, such as energy efficiency.

1.3 Self-adaptability as the key solution

Self-adaptability is the ability of a system to function as expected in the presence of runtime changes in user requirements and operating conditions. A change-proof software architecture can self-adapt itself and its underlying infrastructure based on new/emerging situations. Salama *et al.* propose a taxonomy to describe various facets of architectural stability [213]. One dimension of architectural stability is behavioral stability, which corresponds to the self-adaptability property. The challenge is to create software systems that are externally stable in their behavior and are internally self-adaptive. Self-adaptability could be enabled for both functional requirements and quality requirements. Self-adaptation can in particular help in situations where requirements have potential conflicts by making runtime trade-offs.

Some approaches have been presented to allow runtime self-adaptation in different fields. For instance, in the mobile computing field, Floch *et al.* introduce MADAM (mobility- and adaptation-enabling middleware). MADAM can replace components of mobile applications at runtime [95]. In another example, the SAFDIS (Self-Adaptation For DIstributed Services) framework focuses on

runtime adaptation of service-based applications running in distributed environments [99]. The existing approaches mostly implement the MAPE-K (Monitor-Analyze-Plan-Execute) model [46], which presents the self-adaptation functionalities at runtime.

Although there are studies that optimize energy efficiency by runtime self-adaptation, there is no guideline for software architects on how to deploy them at design time. Software architects need reusable architectural mechanisms. They should be able to compare possible architectural mechanisms qualitatively and quantitatively to make the best fitting design decisions.

1.4 Research Questions

As said before, improvements on energy efficiency should and will attract more of the effort of software architects and software engineers. To ensure energy efficiency of software systems over time (a.k.a environmental sustainability), self-adaptability must be enabled, such that unexpected changes to energy consumption can be recovered at runtime. In this thesis, we target the relationship between energy efficiency of software systems and their self-adaptability with our main research question (RQ) as follows:

RQ: How can we improve energy efficiency of software systems by enabling self-adaptation?

To answer this question, we first need to understand the state-of-the-art in the field. We explore the existing approaches that investigate the link between energy efficiency and self-adaptability, i.e. they enable self-adaptation of software systems for the purpose of higher energy efficiency. The categorization of the existing approaches will reveal a set of reusable approaches and techniques as a reference guide for selection in various types of software systems. Additionally, we can spot the gaps in improving energy efficiency through self-adaptation. Therefore, we define our first sub-question as follows:

RQ1: What are the emerging approaches in the field of energy-efficient self-adaptation?

Once we identified a reusable set of approaches highlighting self-adaptability and energy efficiency of software systems, we explore both bottom-up and top-down approaches to guide software architects and system engineers. The former highlights the impact of infrastructure-specific solutions on the energy efficiency of the running software systems, while the latter considers energy efficiency solutions that target the software design. With answering our next sub-questions, we aim to position the role of each approach type in the big picture of energy efficient self-adaptive software systems.

As for our bottom-up approach, we turn our focus on systems infrastructures, in particular software-defined infrastructure, which can realize self-adaptability with the means of programmability, if utilized to their optimal potential. With the help of programmable infrastructures, software systems can be fully in charge of their resources consumption. Computer networks have shown high potential for improvements in energy savings of software systems. Therefore, we first gather insights on existing networking solutions in cloud-based environments. Our findings will characterize optimization algorithms that are employed to program the systems infrastructure at runtime. Optimization algorithms perform as connectors between software and hardware, in which they schedule the utilization of the infrastructure in the most energy efficient way. As our next step, we develop an optimization algorithm, which we evaluate its quantifiable impact in a controlled experimental environment.

RQ2: How can software-defined infrastructure help increasing energy efficiency of software?

- RQ2.1: What are energy efficient solutions for networking in the cloud?
- RQ2.2: How can optimization algorithms utilize infrastructure in an energy efficient way?

As for our top-down approach, we aim our attention at software design. We look at design decisions that consider energy efficiency of software systems. For that purpose, we identify self-adaptation architectural tactics that can be employed in different usage contexts. We design a number of simulation-based and empirical experiments to quantitatively measure the effectiveness of self-adaptation tactics on improving energy efficiency. Our results can serve software architects as reference guides. We show in a domain model which concepts should be taken into account to build a self-adaptive software system, and how such concepts are related.

RQ3: Can we guide architectural tactics to ensure energy efficiency of software systems?

- RQ3.1: How can we evaluate the effectiveness of the self-adaptation architectural tactics in different usage contexts?
- RQ3.2: Which architectural tactics can ensure energy efficiency of software systems?

1.5 Research Methods

To answer our research questions we use a number of commonly-applied research methods:

- *Systematic Literature Review*: This research method focuses on collecting and analyzing relevant studies in a systematic procedure [141]. We use this method to find the research gaps and the emerging approaches objectively in Chapters 2 and 3. We start from the research questions to define our search queries. Executing the search query in search engines and filtering out irrelevant studies result in the list of primary studies, which are further analyzed.
- *Simulation-based and Empirical Experimentation*: This research method is based on *quasi-experimentation* [31], in which subjects are pre-selected with no randomization. We plan our experiments in terms of variable selection and hypothesis formulations. To study the objects of our experimental studies, we benefit from both simulation-based and empirical experimentations. in Chapter 4 we run simulation-based experiments to evaluate our optimization algorithms in terms of energy efficiency of the networking component. Chapter 5 uses empirical experimentation to investigate cyber-foraging architectural tactics for surrogate provisioning in terms of their energy efficiency and resilience. In Chapter 6, we employ both types of experimentation namely, simulation-based and empirical to assess our self-adaptation framework, which combines software architectural tactics with optimization algorithms. With empirical experimentation, we study the patterns of resource consumption in some well-known mobile applications. Then, we use the collected data as input in our simulation-based framework to evaluate energy efficiency of mobile apps.
- *Conceptual Modeling Method*: We partially follow the steps introduced in the KISS method for object orientation [143]. This method aims at modeling the relevant concepts for a specific domain. We apply this method to scope the domain of self-adaptation, as described in Chapter 7.

1.6 Thesis at a Glance

Figure 1.2 represents the overview of the thesis, including the correspondence between the research questions and the research methods in the chapters. We conduct a systematic literature review to answer RQ1 in Chapter 2. The results of the review lead us to our next research questions. With RQ2 we focus on software-defined infrastructure as an essential element to achieve energy efficiency. Chapter 3 targets RQ2.1 specifically by means of a systematic literature review. It identifies the trending solutions to make cloud-based networks more energy efficient. To answer RQ2.2, we introduce optimization algorithms for cloud-based networks and compare them with the commonly deployed algorithms in Chapter 4.

The results of the two systematic literature reviews help us formulate self-adaptation architectural tactics. In Chapter 5 we evaluate the self-adaptation architectural tactics in the context of cyber-foraging applications. Chapter 6 focuses on mobile applications that are equipped with self-adaptation tactics. We run simulation-based experiments to compare such mobile applications in terms of energy efficiency and other quality requirements. Finally, in Chapter 7, we provide a domain model for self-adaptive software systems. We show all the relevant concepts and their relations that are essential for a seamless adaptation for the purpose of ensuring energy efficiency in any software system.

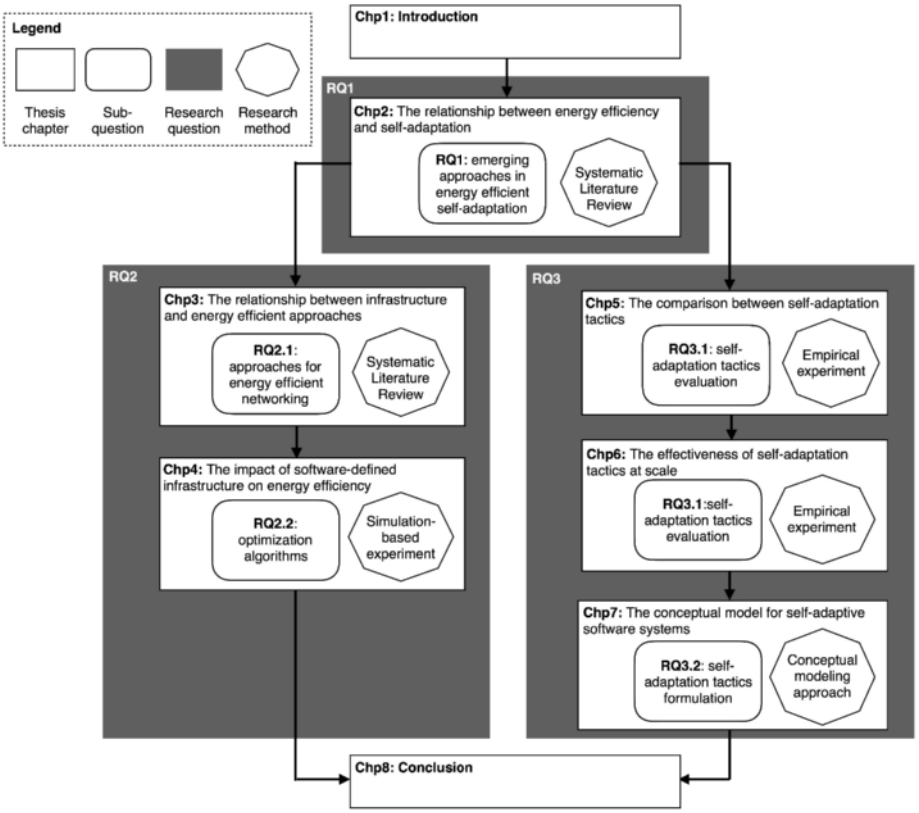


Figure 1.2: The overview of the thesis, including the chapters, the research questions and the research methods

1.7 Publications

All chapters of this thesis are peer-reviewed publications. The first author has designed, implemented and performed the studies. The interpretation of the results are done with the help of the other contributing authors.

- **Chapter 2** (published): Alizadeh Moghaddam, F., Lago, P., Christina Ban, I., "Self-adaptation Approaches for Energy Efficiency: a Systematic Literature Review", In 2018 IEEE/ACM 6th International Workshop on Green And Sustainable Software (GREENS) (pp. 35-42), IEEE, 2018.
- **Chapter 3** (published): Alizadeh Moghaddam, F., Lago, P., Grosso, P., "Energy-efficient Networking Solutions in Cloud-based Environments: A Systematic Literature Review.", ACM Computing Surveys (CSUR) 47.4: 64, 2015.
- **Chapter 4** (published): Alizadeh Moghaddam, F., Grosso, P., "Linear Programming Approaches for Power Savings in Software-defined Networks.", NetSoft Conference and Workshops (NetSoft), IEEE, 2016.
- **Chapter 5** (published): Alizadeh Moghaddam, F., Procaccianti, G., Lewis, G. A., Lago, P., "Empirical Validation of Cyber-Foraging Architectural Tactics for Surrogate Provisioning", Journal of Systems and Software, 2017.
- **Chapter 6** (published): Alizadeh Moghaddam, F., Simaremare, M., Lago, P., Grosso, P., "A Self-adaptive Framework for Enhancing Energy Efficiency in Mobile Applications", Sustainable Internet and ICT for Sustainability (SustainIT), 2017.
- **Chapter 7** (published): Alizadeh Moghaddam, F., Deckers, R., Proccacianti, G., Grosso, P., Lago, P., "A Domain Model for Self-adaptive Software Systems", Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings, ACM, 2017.

2

Energy-Efficient Self-Adaptive Software Systems - A Systematic Literature Review

The increasing energy demands of software systems have set an essential software quality requirement: energy efficiency. At the same time, the many contextual changes faced by software systems during execution can hamper their functionality and overall quality. To address both problems, self-adaptation approaches can empower software systems, at both design-time and runtime, to adapt to dynamic conditions. In this way, software systems can be more resilient to failure, hence more trustful to satisfy the demands of modern digital society. In this chapter, we perform a systematic literature review to study the state-of-the-art on existing self-adaptation approaches for energy efficiency. We analyze the identified approaches from three different perspectives, namely publication trends, application domains, and types of software systems. Our findings can help solution providers to make guided decisions to enable self-adaptability in designing and engineering software systems. The detailed analysis of the existing self-adaptation strategies and the emerging approaches in this field provide us with the answers to RQ1.

2.1 Introduction

Energy efficiency is attracting increasing attention. The ever-increasing demand for more energy has urged the European Commission to identify energy efficiency as the primary quality requirement for achieving sustainability objectives, which is pointed out in its energy efficiency plan 2011 [61]. Information and communication technology (ICT) contributes to the total energy consumption significantly. Only in 2012 ICT consumed approximately 920 TWh, which is around 4.7% of the total produced electricity [62]. Software and hardware technologies as the main ingredients of ICT play an essential role in defining the level of energy efficiency of ICT. To improve energy efficiency both software and hardware technologies

can be targeted. However, software technologies determine the way hardware is exploited [200], and as such can have a more dominant influence on reaching efficiency goals compared to hardware technologies.

Modern software systems cope with many contextual changes during execution, such as changes in environmental conditions and user requirements. Self-adaptability must be enabled for such systems to guarantee the achievement of functional and non-functional requirements. With self-adaptability, software systems can detect unexpected runtime changes and recover from those with available adaptation configurations. Runtime self-adaptation has been described in the MAPE model in the form of four main functionalities [46]: *Monitor*, *Analysis*, *Plan*, and *Execute*. These functionalities must be realized in a loop to enable an autonomic adaptation to a runtime change. There are a number of studies exploring self-adaptation approaches for software systems [146, 167, 263]. In this work, however, we are especially interested in self-adaptation approaches aimed to improve energy efficiency. Accordingly, we study pre-existing approaches to uncover the link between self-adaptability and energy efficiency in software systems. We perform a systematic literature review (SLR). The resulting primary studies are analyzed, and the findings are presented as a guideline including categorizations along various perspectives. As a result, we believe that a wide range of solution providers (from software architects to system engineers) can benefit from this guideline by adopting the best-fitting approaches depending on their own specific requirements. We gather our findings around the notion of *approaches*, which are reusable generic sets of operations. Approaches, in turn, can be adopted and specialized into *solutions*, which are technology-oriented and system-specific. Our results reveal the current and the possible future focus of software solution providers on different approach types.

The rest of the chapter is organized as follows. Section 2.2 describes the research method we follow. In Section 2.3, we present the analysis of the identified primary studies. This relies on a meta-model illustrating the relevant information extracted from the primary studies. Section 2.4 introduces the findings and their categorization. We discuss our main observations in Section 2.5, and the threats to validity are explained in Section 2.6. Related work on self-adaptation approaches and energy efficiency improvements is presented in Section 2.7, and Section 2.8 concludes the chapter.

2.2 Research Method

We conducted an SLR to objectively select the primary studies to base our analysis on. According to [141], an SLR can be organized in four main steps:

1. *Research Question Identification*: The main goal of this SLR is to review

the existing self-adaptation approaches aimed to improve energy efficiency of software systems. We achieve this goal by formulating the following research questions:

- **RQ1**. What are the types of approaches emerging over time? By answering this research question we aim at characterizing the intensity of scientific works targeting self-adaptation approaches over the years. This may help evaluate possible trends in the coming years.
- **RQ2**. How can we categorize the application domains of the approaches? The answer to this question will help identify which types of approaches can be more suitable for which domains.
- **RQ3**. How can we categorize the types of software systems targeted by past research studies? The answer to this question maps the approaches to the types of software systems, namely data-intensive or computation-intensive. This will help selecting better fitting approaches for a given type of software system and with specific quality requirements.

The research questions should be answered by extracting the relevant information from the primary studies. Identifying the research questions helps us with the next steps, to select and assess the primary studies.

2. *Search Strategy Planning*: Thanks to its accessibility, we have selected Google Scholar as the source of relevant studies in our study. To run the search, we have constructed a search query with specific syntax suitable for Google Scholar:

   ("(energy OR power) (efficiency OR efficient OR saving)") AND ("self-adaptive (software OR system OR framework)" OR "(software OR system OR framework) self-adaptation" OR "self-adaptive application")

   We started with 687 studies as the result of the search query[1]. The relevant studies are retrieved if they contain the keywords set up to cover the research question. To ensure the effectiveness of our query, we checked the results against a list of pilot studies that we identified based on our knowledge of the field. Pilot studies are in fact primary studies that should appear among the results of the query execution.

3. *Study Selection*: We assessed the retrieved studies according to our inclusion and exclusion criteria, which are summarized in Table 2.1. For efficiency reasons, we performed the assessment in three rounds: title-based, abstract-based, and body-based. In each round, a number of irrelevant

[1]The search query was executed in Google Scholar on January 29, 2018.

studies were excluded, and the remaining studies were used as input for the next round. The last round resulted in 95 primary studies, which we have used for analysis.

Table 2.1: Our inclusion and exclusion criteria to assess the retrieved studies

#	*Criterion*	*Description*
	Inclusion Criteria	
I1	The study is written in English.	For readability purposes, we include only studies that are in English.
I2	The study is peer-reviewed.	We ensure the quality of the primary studies with selecting only from peer-reviewed studies.
I3	Self-adaptability is investigated in the study.	Self-adaptation approaches are the focus of our study. We investigate approaches enabling self-adaptability in software systems at different levels (from software design to system engineering)
I4	The main focus of the study is energy efficiency.	The objective of the studies is to improve on energy efficiency. We also take into account other energy-related concepts, such as power efficiency and energy consumption.
	Exclusion Criteria	
E1	The body of the study is not available.	The body text of the studies must be available. We exclude the studies that cannot be retrieved publicly. It should be mentioned that with our university license, we can access top journal publications and conference proceedings.
E2	The study does not provide a solution.	The studies must contribute a solution to the field. For instance, we exclude discussion studies that do not provide self-adaptation solutions for energy efficiency.
E3	The study does not contribute to the link between energy efficiency and self-adaptability	We exclude the studies that do not investigate the link between energy efficiency and self-adaptability.

4. *Primary Studies Management*: During the study selection phase, we used Google Scholar, in which we could assign customized labels to each retrieved

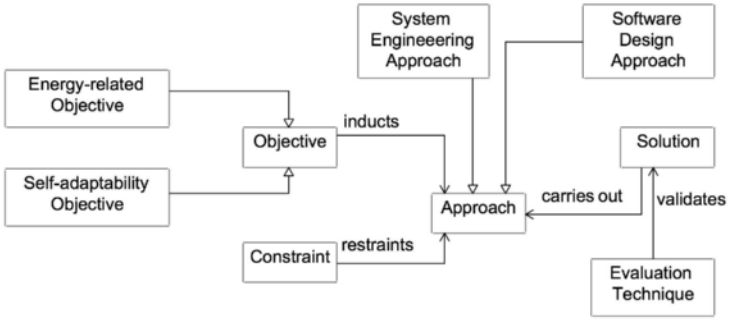


Figure 2.1: The metamodel presenting the concepts and their relations extracted from the primary studies

study. During the analysis phase, we used an Excel sheet to store relevant information extracted from each primary study.

2.3 Data analysis

We used the coding method [236] to systematically analyze the primary studies. Accordingly, one assigns codes that emerge from the text in each primary study. The codes help compare and discover the similarities between the primary studies. We categorize the extracted codes further to identify more generic concepts. This process resulted in the metamodel (see Figure 2.1) that illustrates the uncovered concepts and their relations. Among them we identify five main classes:

- *Objective*: This is the goal that each study aims to achieve, whether it is linked to self-adaptability, energy efficiency, or both. *Energy-related objectives* and *self-adaptability objectives* indicate the two types of objectives that have been addressed. Objectives must be specified before initiating the process of forming approaches. At a higher level, objectives can arise from a financial issue. For instance, an organization wants to lower the energy bills. Thus, its objective can be minimizing the energy consumption of its infrastructure, which is an energy-related objective. In another example, for resource-scarce environments, the efficient utilization of computing resources is a necessity. In these cases, software systems have the self-adaptability objectives. It should be noted that as a by-product the energy efficiency of such systems improves.

- *Constraint*: This is a limitation to customize approaches due to environmental conditions and user requirements. For example, financial limitations

can narrow down the range of possible approaches to be adopted. Constraints can be zoomed in to more details and restrict a selected approach. For instance, if the users of an application have strict requirements on the output performance, then the energy-related solutions can only hinder performance up to a certain level.

- *Approach*: This is a viewpoint for solving the stated problem in the primary studies and achieving the objectives. An approach comprises an ordered set of (usually high-level) operations designed to deliver the proposed solution. It is a process of working through the details of a problem to realize the proposed solution. With an example we can describe an approach and its relation to a corresponding solution. Assume there is a data center framework that aims to optimize allocation of resources for input workload, which can have a varying pattern during execution. A possible approach to enable self-adaptability for such framework is to implement pattern recognition models that can identify the pattern of each workload, and dynamically select the optimum allocation plan. A solution for this example could be the specific recognition algorithms implemented in the framework.

  We identify two types of approaches in the primary studies: *software design approaches* are concerned with the design of software systems throughout their entire life-cycle; *system engineering approaches*, instead, focus on runtime system-level adaptation configurations.

- *Solution*: This is a specific low-level method to solve a problem. Solutions realize approaches, which are generic and applicable for target software systems. For instance, in cyber-foraging an implemented code partitioning algorithm is a self-adaptation solution, while the approach is "to offload the computation".

- *Evaluation Technique*: This is a technique adopted to evaluate the outcome of the proposed solution in each primary study. The evaluation technique determines the degree of improvement on energy efficiency when the self-adaptation solution is applied. A wide range of techniques has been used in our primary studies, such as qualitative discussions, mathematical formulations, empirical experiments, and simulation experiments.

2.4 Results

The analysis of the primary studies helps answer our research questions. The answer to the research questions will help software architects and software engineers to choose the best fitting approaches for their software systems.

RQ1. What are the types of approaches emerging over time?
As mentioned before, we have identified two types of approaches, namely software design approaches (SDA) and system engineering approaches (SEA). Some primary studies propose solutions that realize both types of approaches (SEA+SDA). Figure 2.2 shows the distribution of each type of approaches in the list of our primary studies throughout the years.

As shown in the figure, there is a growing trend for the both SEA and SDA approach types until 2014, when we witness a peak in adoption of the SEA and SDA types. The second peak is in 2017, in which all the types of approaches have been investigated.

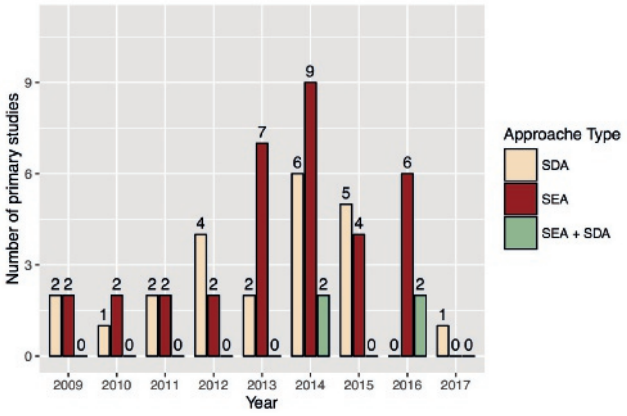


Figure 2.2: The emerging trend of the adopted approaches throughout the years. Different colors indicate different types of approaches, namely software design approaches (SDA), system engineering approaches (SEA), and both (SEA+SDA).

Mostly, the approach type SEA is adopted more than other types in each year. We see that only in 2012 and 2015, the primary studies turn their attention more towards the type SDA. This means that a lot of effort has been devoted to system engineering approaches in order to realize runtime adaptation. Many scheduling algorithms and infrastructure configuration mechanisms are implemented at this stage. We expect to see an increase in growth for all types of approaches in the coming years.

An example for type SEA is introduced in [185]. They provide a self-adaptive framework that adjusts cluster configurations during runtime, to cover heterogeneity and cluster status variations. The framework has built-in learning algorithms to compare current and previous configurations, and predict cluster performance. All the MAPE functionalities are realized by including essential components such as a monitoring network, a dynamic predictive model, and a runtime scheduler.

Differently, Perez-Palacin *et al.* [195] present an approach of type SDA. They introduce a reference architecture, or an adaptation framework, based on a three-layer software architecture for self-managed systems. The reference architecture

uses model-driven techniques to transform design patterns into different Stochastic Petri Nets subnets.

We have identified a few studies of type SEA+SDA. For instance, Cioara *et al.* [60] propose a methodology for energy-aware management in data centers. They have defined an ontological model for representing the key indicators in data centers, namely energy and performance. The model helps calculate the level of energy efficiency in data centers at runtime and apply reconfigurations when needed.

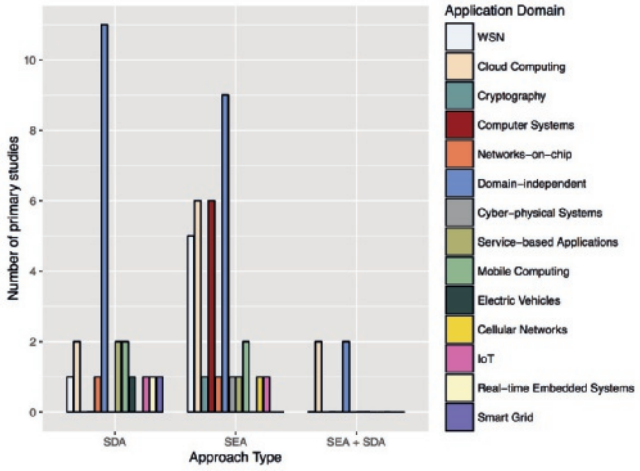


Figure 2.3: The distribution of the types of the self-adaptation approaches in different application domains.

RQ2. How can we categorize the application domains of the approaches? The self-adaptation approaches are generic guidelines that can be applied to different domains. From the primary studies we identify several application domains the approaches are customized for. Figure 2.3 shows the popularity of each type of self-adaptation approaches for different application domains. The cloud computing domain has benefited the most from the all types of approaches. The second popular domains are computer systems and wireless sensor networks (WSN), in which many

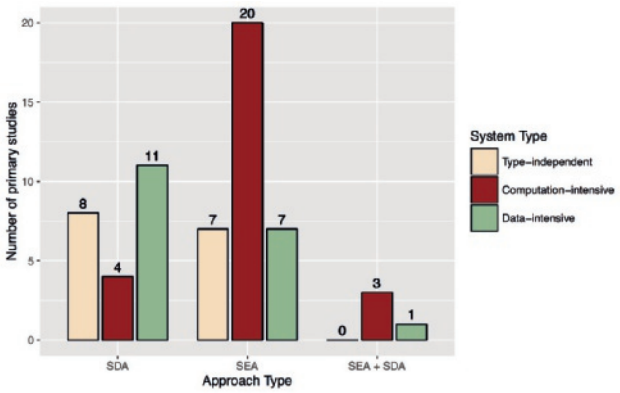


Figure 2.4: The distribution of types of software systems targeted by the primary studies.

approaches focus on runtime resource optimizations. In particular, the limited battery life of sensors in WSN has motivated a lot of effort on runtime adaptations. In general, we can see from the figure that the approaches of the type SEA are usually proposed for specific domains. The reason is that these approaches are mostly technology-driven, and hence domain-dependent.

Another observation is that many of the proposed approaches are not proposed for a specific domain. We label these approaches as *Domain-independent* that enable self-adaptability from a general-purpose point of view. For instance, a primary study that introduces an online learning algorithm to make dynamic adaptation decisions falls under this category. The domain-independent category itself can be sub-categorized into more detailed categorizations such as requirements engineering, software design, and optimization algorithms. It should be noted that general-purpose approaches are mostly from type SDA focusing on self-adaptation software architecture.

Type SEA+SDA has been specified for the cloud computing domain. This domain has attracted the highest number of approaches, which is an indicator of the the maturity of the cloud computing domain in general. Table 2.2 illustrates the primary studies focusing on each application domain.

RQ3. How can we categorize the types of software systems targeted by the primary studies?
The ratio between the size of computed data and the size of computing power can show the types of software systems, which are namely, data-intensive and computation-intensive. Most primary studies do not specify a system type, classified as *Type-independent*. Figure 2.4 shows the classification of the primary studies based on the types of software systems. From an overall perspective we see that computation-intensive tasks are mostly the target for self-adaptation approaches. We see this pattern for the types SEA and SEA+SDA. However, the same pattern does not apply to the type SDA, in which approaches are mostly focused on data-intensive tasks rather than computational-intensive ones. Table 2.3 shows the primary studies that are proposed for the two system types.

2.5 Discussion

By zooming into the self-adaptation approaches, we can add more details to each approach type as listed in Table 2.4. In particular, *software design approaches* (SDA) target the software design at different stages ranging from software development life-cycle to execution time. We have identified three categories of SDA approaches on *requirements engineering*, *architecture modeling*, and *architecture reconfiguration*. For instance, Bencomo *et al.* present a requirement engineering approach that systematically and incrementally builds software architectures from system goals [35]. They use goal-based requirements specification for self-

Table 2.2: The application domains identified in the primary studies

Application Domain	***Approach Types***	***Primary Studies***	*#*
Domain-independent	SEA / SDA / SEA+SDA	[7, 13, 22, 35, 38, 48, 50, 56, 79, 83, 91, 102, 106, 107, 118, 163, 175, 176, 180, 195, 205, 207, 210, 219, 235, 253, 256, 261, 262, 270]	30
Cloud Computing	SEA / SDA / SEA+SDA	[18, 20, 21, 23, 60, 72, 74, 75, 89, 90, 133, 221]	12
WSN	SEA / SDA / SEA+SDA	[5, 11, 109, 126, 135, 136, 139, 185, 198, 216]	10
Computer Systems	SEA	[29, 80, 105, 117, 208, 232, 243, 251]	8
Networks-on-chip	SDA / SEA	[10, 87, 124, 183, 266]	5
Mobile Computing	SEA / SDA / SEA+SDA	[69, 96, 108, 179, 203]	5
Service-based Applications	SEA / SDA	[70, 73, 174]	3
Smart Homes	SDA / SEA / SEA+SDA	[186, 206, 222]	3
IoT	SEA / SDA	[67, 181, 255]	3
Cyber-physical Systems	SDA / SEA	[202, 218]	2
Electric Vehicles	SDA / SEA	[123, 217]	2
Real-time Embedded Systems	SDA	[212]	1
Smart Grid	SDA	[151]	1
Micro-architecture	SDA	[258]	1
FPGA	SDA	[88]	1
Cyber-foraging Systems	SEA+SDA	[16]	1
NLP	SEA	[8]	1
Micro-electronics	SEA	[242]	1
Cryptography	SEA	[246]	1
Cellular Networks	SEA	[119]	1
Lighting Systems	SEA	[138]	1
HPC	SEA	[233]	1
Embedded Systems	SEA	[100]	1

Table 2.3: The types of system tasks identified in the primary studies

Software System Type	***Approach Types***	***Primary Studies***	*#*
Computation-intensive	SEA / SDA / SEA+SDA	[5, 18, 21, 23, 60, 69, 72, 80, 87, 89, 90, 96, 105, 106, 107, 119, 133, 135, 174, 185, 202, 207, 210, 219, 221, 232, 235, 251]	28
Data-intensive	SEA / SDA / SEA+SDA	[7, 50, 67, 70, 73, 74, 79, 83, 91, 102, 124, 175, 176, 179, 181, 198, 212, 216, 253]	19

adaptive software systems. *Architecture modeling* approaches introduce reference architectures that guide software architects in building self-adaptive software systems. The design patterns proposed by Said *et al.* for self-adaptive real-time embedded systems fall under this category of approaches [212]. Our analysis shows that the proposed reference architectures mostly adopt a layered architectural style. The second most common architectural style is the service-oriented style that with an automated composition of software services can seamlessly adapt to runtime changes. Another interesting category of the type SDA is *architecture reconfiguration*, which suggests modifications to the software architecture at runtime. A nice example for this category is the work by Mizouni *et al.* that presents a battery-aware cyber-foraging mobile application [179]. In their work, architecture reconfigurations are introduced in the form of features availability in the application, at both the mobile side and the surrogate side.

System engineering approaches (SEA) target the MAPE model functionalities at the system level. The type SEA includes the specific solutions to realize each of the MAPE functionalities in terms of specific tools and methods. We have identified two categories of SEA approaches namely, *MAPE functionalities* and *system architecture*. Self-adaptation approaches that focus on the seamless adaptation at runtime, usually cover all the MAPE functionalities at once. However, we have seen some approaches that investigate only one functionality. For example, Mishra *et al.* propose a probabilistic graphical model-based learning algorithm to enable energy savings [176]. Another category of the type SEA is *system architecture*, which introduces an architecture for tools and technologies to enable self-adaptability. The work presented by Alzamil *et al.* is an example of this type [18]. They propose a system architecture for profiling and assessing the energy efficiency of cloud infrastructure resources.

Table 2.4 lists the primary studies relevant for each category of approaches. As shown in the table, some primary studies fall under more than one category. Most of the primary studies from the type SEA focus on the *MAPE functionalities* rather than their *system architectures*. Differently, the primary studies from the

type SDA focus more on *architecture modeling* rather than other categories. The few number of primary studies on *architecture reconfiguration* indicates that it is a young field, and more advances in this category will emerge in the following years.

Table 2.4: The categorization of self-adaptation approaches identified in the primary studies

Approach Type	***Approach Category***	***Primary Studies***	*#*
Software Design (SDA)	Architecture Modeling	[10, 13, 22, 23, 48, 50, 56, 60, 74, 75, 88, 96, 102, 108, 124, 139, 151, 163, 174, 181, 195, 198, 206, 207, 212, 217, 218, 235, 258]	29
	Requirement Engineering	[35, 73, 90, 180, 205, 222, 253, 261, 262]	9
	Architecture Reconfiguration	[16, 83, 91, 179, 270]	5
System Engineering (SEA)	MAPE Functionalities	[5, 7, 8, 11, 20, 21, 29, 38, 48, 60, 67, 69, 70, 72, 75, 79, 80, 87, 89, 90, 102, 105, 106, 107, 108, 109, 117, 118, 119, 126, 133, 135, 136, 138, 139, 175, 176, 183, 185, 186, 202, 203, 207, 208, 210, 216, 219, 221, 222, 232, 233, 242, 246, 251, 255, 256, 266]	57
	System Architecture	[18, 20, 29, 69, 80, 89, 100, 118, 123, 126, 135, 243]	12

2.6 Threats to Validity

In the following, we identify a number of threats to the internal and external validity of our work, and explain our mitigation strategy.

The threats to **internal validity** target the design and execution of the review. To prevent possible design errors, we have followed the steps described in the systematic literature review protocol summarized in Section 2.2. In addition, to mitigate a possible subjective execution of our review, three researchers independently followed the inclusion/exclusion criteria used to select the primary studies.

The threats to **construct validity** target the alignment between the research question and the measurements of the review. We evaluate the effectiveness of

our search query with a list of pilot studies that has proven to answer the research query based on our knowledge in the field.

The threats to **external validity** target the generalizability of our results. To mitigate this type of threat, we have used generic keywords to construct a search query that can capture as many relevant studies as possible. Furthermore, we have validated the list of obtained primary studies against a list of pilot studies as an objective common ground in the field.

2.7 Related work

The related work can be considered from two different perspectives, namely self-adaptability and energy efficiency. Each perspective has received a lot of attention from researchers in the last few years but we could not find any research paper specifically focusing on the link between the two perspectives.

From the perspective of **self-adaptability**, Macias-Escriva *et al.* review various methods and techniques employed in the design of self-adaptive systems [167]. They evaluate current progress on self-adaptability from the viewpoint of computer sciences and cybernetics, based on the analysis of state-of-the-art strategies reported in the literature. Krupitzer *et al.* present a comprehensive taxonomy for self-adaptation. They further provide a structured overview on approaches for engineering self-adaptive software systems [146]. Yang *et al.* conduct a systematic literature review to explore the modeling methods and the activities regarding requirements engineering for self-adaptive systems [263]. They also specify the application domains and the quality attributes that the primary studies focus on. Mahdavi-Hezavehi *et al.* review existing architecture-based methods to achieve multiple quality attributes in self-adaptive software systems [171]. Muccini *et al.* explore the existing self-adaptive approaches for cyber-physical systems [182]. They identify the self-adaptation concerns and application domains covered by the primary studies. Energy efficiency has been observed as the most dominant application domain for cyber-physical systems.

In turn, from the perspective of **energy efficiency**, Hammadi and Mhamdi [111] conduct a survey on energy efficiency solutions in data centers such as virtualization, energy-aware routing, dynamic voltage/frequency scaling, dynamic power management, renewable energy supply, and energy-aware cooling systems. Orgerie *et al.* survey the solutions to improve the energy efficiency of computing and communication resources in distributed systems [190]. Alizadeh Moghaddam *et al.* carried out a systematic literature review of the energy-efficient networking solutions in cloud-based environments [15]. Pinto and Castor provide an overview on the contributions of the software engineering community to improve energy-related qualities [196]. They mostly focus on the two main issues in this domain, namely: *lack of knowledge* and *lack of tools*.

Differently from all other surveys we found in the literature, the work presented in this paper focuses on the link between self-adaptability and energy efficiency. In our study, we distinguish between software design approaches and system engineering approaches that can make our findings beneficial to both software architects and system engineers.

2.8 Conclusion

The energy consumption of software systems has been the focus of many research studies. Improvements in energy efficiency can result from applying self-adaptation to software systems. We believe the link between self-adaptability and energy efficiency is a promising target for building energy-aware software systems. Therefore, we carry out a systematic literature review to identify pre-existing approaches that focus on this link. We, in particular, look for *self-adaptation approaches for energy efficiency in software systems* to answer our RQ1, namely "What are the emerging approaches in the field of energy-efficient self-adaptation?". Our findings from the SLR indicate two types of self-adaptation approaches that have been implemented by the primary studies, namely software design approaches (SDA) and system engineering approaches (SEA). The publication trends of the primary studies show an increasing growth in the adoption of the self-adaptation approaches over the years. In particular, more attention has been paid to the type SEA that focuses on runtime realization of the MAPE (monitor-analyze-plan-execute) model functionalities.

Our results show that most approaches are application domain-independent, i.e. can be applied to any domain. Cloud computing is the most common application domain as the target of the approaches. Also, our findings show that computation-intensive software systems have attracted the highest attention of research work so far.

Our analysis shows that four primary studies address runtime *software architecture reconfiguration*, which we identify as one of the categories of the type SDA. In the coming years, we expect to see more attention dedicated to this category of approaches as a response to the increasing maturity of this field. As our future work, we aim to examine the effectiveness of software architecture reconfiguration combined with runtime adaptation mechanisms.

Acknowledgement

We thank Sarah Haddou for her assistance in collecting and selecting the primary studies.

3

Energy-Efficient Networking Component in the Cloud - A Systematic Literature Review

This chapter answers RQ2.1 with the focus on energy-efficient solutions in the networking component of cloud-based environments. The energy efficiency improvements on the networking component can provide both mobile applications and cloud-based software systems with higher efficiency qualities. We follow a systematic literature review method to identify adopted solutions in this field. We categorize the solutions into types: Device, Network Architecture, Routing/Switching Protocol and Decision Framework. Amongst all, decision frameworks have been investigated the most in the recent years. Decision frameworks are mostly self-adaptive software systems that monitor the infrastructure (including software-defined networks) and according to the runtime changes, plan the best-fitted configurations to the networking components.

3.1 Introduction

Cloud-based environments consist of a single or multiple operational data centers that provide resources to deploy cloud-based software systems. High electricity bills have turned the attention of data centers providers towards energy consumption, which is increasing rapidly. According to DatacenterDynamics 2012 Global Census [245], there has been 63% increase in the total power consumption of data centers globally between 2011 and 2012 from 24GW to 38GW. This sharp increase shows the growing demand for energy supply in data centers. Energy saving approaches do not necessarily aim to limit the demand for energy but to find potential rooms for improvements. Statistics show that the average data center workload is only 30% of the load in peak hours [142]. Reliability and

availability are some of the primary quality requirements in the context of cloud data centers, which lead to keep the non-utilized infrastructure up and running.

The distribution of energy consumption within a data center is not homogeneous: a significant portion of energy consumption (80-90%) goes to infrastructure, servers, and cooling. The remaining portion (about 10-20%) is spent on networking [115]. Baliga *et al.* determined that these proportions will change in different scenarios [27]. Based on specific network traffic patterns or data computations, energy consumption of each component would vary. Our focus is on the energy efficiency of the networking component, which shows high potential for improvements in energy savings. The networking component is not the major contributor to total energy consumption, but improvement in this area is still important for the following reasons:

1. The network architectures of data centers are typically richly connected to ensure availability with a lot of redundant network devices. Consequently, a large number of network devices remain up and running all the time.

2. As mentioned earlier, the average load on network devices is around 30% of the total capacity. This means that the network infrastructures are idle most of the time that decreases energy efficiency.

3. The currently employed network devices in data centers are not energy proportional. Based on [115], network devices even at low load or in their idle state consume around 90% of the power consumed in the highly utilized state.

In this chapter, we survey existing solutions in cloud-based environments that aim to improve the energy efficiency of the networking component. Contrary to other surveys in this research area, we adopt a systematic manner, namely the "Systematic Literature Review" research method [141], to define the search strategy and identify our primary studies. We then analyze the primary studies using the coding analytical method [236] to collect and extract the results. The codes are analyzed and aggregated into a meta-model, which illustrates the identified classification in the articles.

The rest of the chapter is structured as follows. After discussing related work in section 3.2, section 3.3 describes the steps taken for collecting the primary studies. Section 3.4 details the classification of collected information and indicates a corresponding meta-model. Section 3.5 focuses on the findings from the primary studies. Existing implemented energy efficient networking solutions are further discussed in section 3.6. Section 3.7 explains the threats to validity. The chapter closes with emerging opportunities of research in section 3.8 and our conclusions in section 3.9.

3.2 Related Work

Several studies in the literature have analyzed the existing energy efficient solutions in various domains.

There are surveys focusing only on the network infrastructure and not including cloud-based environments. For instance [269] describes existing work done in the energy efficiency of optical networks and [40] and [42] provide the current perspectives and emerging technologies on the energy efficiency of network infrastructures in general. These studies are interesting from a general viewpoint, but they miss the specific characteristics of networks suitable for cloud operations.

The energy efficiency of cloud-based environments has been researched from different perspectives. Priya *et al.*, [199] concentrate on energy models of different components in data centers and illustrate energy efficient solutions of cloud computing services (SaaS, PaaS and IaaS). On the other hand, Beloglazov *et al.* [34] survey some ongoing projects from different companies on energy efficient solutions in data centers, which include the networking component and its energy efficiency.

Other surveys have collected energy efficiency solutions of cloud-based environments split by different components. Therefore, the networking component has been included as well but partially as one of the influencing factors in the total energy consumption of data centers. General energy efficiency solutions are gathered in [52, 147, 190], which cover "green" metrics applied in data centers, and how energy saving solutions for servers and cooling are improving energy efficiency. Moreover, they discuss networking solutions as part of these solution sets. Differently, we emphasize only energy efficient networking solutions in data centers and provide a deep analysis of existing solutions. Our analysis goes further by including networking solutions investigated in large scale cloud-based environments as well.

The original contribution of our work compared to the research presented above is that we specifically focus on the networking component of the data centers offering cloud services and present the energy efficient networking solutions provided at different scales: intra-, inter- and mixed-data center scales. We are interested in identifying the solutions proposed to reduce energy consumption of the networking component and the granularity at which they are applied. For the first time in this research area, we use a systematic literature review method to select our primary studies in a systematic manner. While this method is quite common in other research fields (like software engineering and knowledge management), it is not many times adopted in the field of computer networks. Using this method, we provide a fresh perspective of the state-of-the-art in this research area.

3.3 Systematic Literature Review

Systematic literature review (SLR) is a research method to objectively collect the relevant studies based on a predefined search query [141]. As the name implies all the steps are taken via a systematic procedure, which provides a more objective process to select relevant studies in comparison to other review methods. There are four major steps in a SLR: *a*) Definition of the research question; *b*) Search strategy; *c*) Study selection; *d*) Primary studies management.

Following the first step (*definition of the research question*), an initial list of studies is created during the *search strategy* step. The list is used as a starting point in the *study selection* step. Each study is examined according to our research question. This requires the definition of selection criteria which acts as an objective guidance in selecting primary studies. It is important to record all the inclusion and exclusion rules. Inclusion criteria determine if one study can be a candidate primary study; they are the minimum set of conditions that have to be met by a primary study. After filtering the studies based on the inclusion criteria, the resulting relevant studies are evaluated according to exclusion criteria. The remaining set of articles are called **primary studies**. Primary studies are the output of the SLR method and the input for further analysis and discussion. With the *primary studies management* step, we facilitate the process of study selection. The following describes in more details each step of the SLR:

Research Question We set out to investigate the current solutions adopted in cloud-based environments to improve the energy efficiency of the networking component. Namely we want to answer the question: *What are the energy efficient networking solutions in cloud-based environments?*

Search Strategy We use *Google Scholar* as input data source thanks to its accessibility and reproducibility by others. We construct the search query based on our research question with a combination of several relevant keywords and boolean AND's/OR's. The keywords are chosen to maintain the proper balance between generality and specificity at the same time. To make sure that the query we define is effective in finding relevant studies we verify its sensitivity using a set of pilot studies.

Pilot Study: Based on our knowledge of the field, we found a list of 15 pilot studies, which we expect to show up as primary studies. We trained the SLR protocol with these pilot studies to make sure that each one of them is retrieved by the query.

The resulting search query is formulated as:

Search Query: *routing "data-center" network cloud (intitle:energy OR intitle:power) -intitle:mobile -intitle:telecom -intitle:wireless -intitle:hoc -intitle:radio*

-intitle:smart

The specific syntax keywords shown in the search query are related to our choice of input data source. For different digital libraries the syntax would vary. In order to make the search query more tangible, hereby we describe the search operators:

- intitle:*term*: It finds the studies that have *term* in their title.
- The - operator: It can be used before any word or operator to give a "NOT" meaning to that part of search query. For example *-intitle:wireless* in our search query targets the studies that do not have the term wireless in their title.
- "*phrase*": Using quoted phrases, it is possible to search for studies containing the exact *phrase*. In our search query we include "data-center" and *Google Scholar* is able to collect studies containing either "data center" or "data-center".
- (): By using parentheses, it is possible to skip conflicts of operators. In our search query all words and operators are put together with an implicit "AND" operator between them. Because we need to use the "OR" operator as well, we put it in the parentheses.

Study Selection After running the above query on Google Scholar we obtained the relevant studies[1]. We assessed each study for its actual relevance through two sets of selection criteria: Inclusion and Exclusion. Table 3.1 lists all the inclusion criteria we used to identify our primary studies. Table 3.2 lists all the exclusion criteria that must not be seen in one of our primary studies. As the end result of this phase, we collect all relevant primary studies.

Instead of reading all the studies at once, we completed the selection process in multiple stages to accelerate the procedure. We used the output set of studies from each stage as

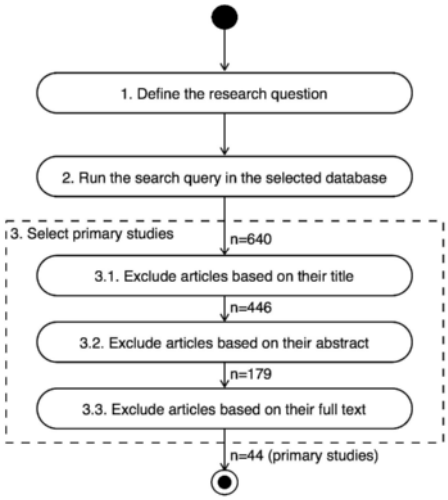


Figure 3.1: The steps taken in our systematic literature review, n being the number of remaining studies in each step

[1]The search query was run on 13 November 2013 in Google Scholar.

Table 3.1: Inclusion Criteria

#	Criterion	Description
1	The study is written in English.	There are some studies written in languages other than English but because of providing an English title or abstract, they show up in our query result. Due feasibility reasons, only studies written in English will be included.
2	The study is peer-reviewed.	To ensure satisfying quality of primary studies, only peer-reviewed studies will be chosen since they are already published by a professional scholarly society.
3	The networking component is considered.	There are plenty of energy efficiency techniques investigated in cloud-based environments, which are not focused on the networking component such as cooling technologies. So it is needed to specify the requirement to find studies that aim to improve energy efficiency of the networking component.
4	Data and services are in cloud-based environments.	In order to investigate the solutions provided specifically in cloud-based environments, we do not consider cases that data and services need to be transferred from customer side to the cloud.

an input set for its next stage. In the first two stages, we processed the title and the abstract of the relevant studies. Finally we assessed the whole body of studies and we considered the output of this stage as our primary studies, which will be used in the data analysis. Figure 3.1 represents the steps of SLR and the number of remaining studies for each step.

We started from 640 studies as the result of our search query and we ended up with 44 primary studies. From the removed 596 studies, around 45% were removed during stage #3 (selection by abstract). In most cases, reading the abstract eases the decision making process since it describes the main goal and the scope of the study.

Primary Studies Management We used two applications in the process of obtaining primary studies and analyzing them: the study selection of different stages is done in *Zotero*[2]; we then import the primary studies in *ATLAS.ti*[3], which makes qualitative analysis of large number of studies much easier.

[2]http://www.zotero.org/
[3]http://www.atlasti.com/index.html

Table 3.2: Exclusion Criteria

#	Criterion	Description
1	The network infrastructure is not wired/optical	We are not interested in wireless network infrastructure deployed in data centers. In the search query some limitations on title are defined for example not having “wireless”, “smart”, “mobile” and “ad hoc” in the title.
2	Energy efficiency is not the primary requirement.	According to our research question, the aim is to find the energy efficient solutions. So the studies investigated to model and monitor power/energy consumption of components in cloud-based environments will not be the case, unless they provide energy efficiency solutions.
3	The main focus of the study is not the networking component.	Energy efficiency of cloud-based environments is dependent on several components; one being as the networking component. Because of the high impact of servers on total energy consumption of data centers, most studies are focused on this part and the energy efficiency issue in the networking component is considered implicitly. For example VM consolidation in data centers will effect the energy consumption of the networking component. But if the study is only covering considerations of potential congestion in the network, then it will be removed from our list.
4	The study does not include data center environment.	As mentioned earlier we consider data centers as cloud-based environment’s infrastructure. Other types of cloud IT infrastructure will not be included.
5	Missing study source.	Some studies release their abstract publicly but the body text can not be found because of either non-public publications or other transfer issues. These studies will be removed in the selection process. However, it should be noted that with our university license we can access most of the major scientific journals.

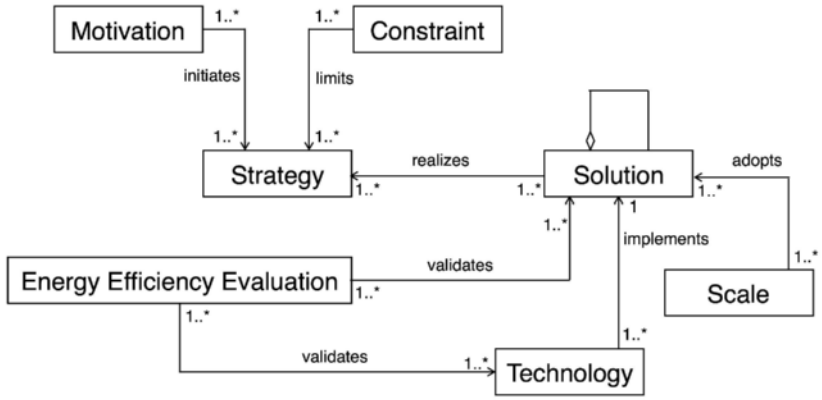


Figure 3.2: The meta-model showing the summary codes extracted from the primary studies

3.4 Data Analysis

We use Coding [236] as our analysis method, where from each study we extract a number of codes that summarize the relevant valuable information with summative words/phrases. After collecting all coded data out of the primary studies, the codes are clustered together based on relations between concepts to make hidden patterns visible. Finally each cluster is mapped to a more general code, which is shown in our meta-model as a class.

The meta-model has emerged from the identified codes in the primary studies. The aggregated codes are shown in the meta-model as classes and they are related to each other based on the identified patterns in the primary studies. Figure 3.2 illustrates our meta-model and the elements necessary for providing energy efficient solutions in all the primary studies. In the following we provide a more detailed description of each class:

1. **Motivation:** It is the reason that makes cloud providers think of energy efficient solutions. The motivation can even be initiated from non-networking triggers but can implicitly impact the energy consumption of the networking component. High electricity costs, using renewable energy and different energy policies are some of the example motivations in data centers.

2. **Constraint:** In the service provisioning process, the main objective has always been to meet quality requirements like performance and network throughput. But due to the rapid growth of energy consumption in cloud-based environments, energy efficiency is also added to the list of efficiency

qualities that have high priorities. Optimum energy efficiency in a system might cause degradation in other efficiency factors. For example aggregating network traffic into fewer number of network devices, improves the energy efficiency of the networking component while it introduces some increase in packet delivery delays. This class puts emphasis on the trade-off that has to be made between the required level of energy efficiency and other efficiency factors such as network throughput.

3. **Strategy:** A strategy introduces a high-level plan to achieve energy efficiency, which will be realized with an implemented solution. For example if the solution focuses on *Devices*, the strategy to improve energy efficiency could be "Replacing current devices with less energy consuming ones", or if the *Routing/Switching* solution is established, the strategy could suggest "Energy aware routes". A Strategy is in essence providing a high level of abstraction in this classification.

4. **Solution:** A cloud-based environment, which is a data center or a network of multiple data centers, consists of a variety of components involved in energy consumption of the networking part. A solution is described as a method that aims to reduce energy consumption in one or more of these components.

5. **Technology:** It is the specification of how a solution is implemented and deployed in order to satisfy the corresponding strategy, which is triggered by motivations and limited by constraints. It is the lowest level of abstraction in this classification, which covers specific technical details.

6. **Scale:** The identified energy efficient networking solutions are deployed in different scales of cloud-based environments. As shown in the meta-model based on the considered scale, the solution will be adopted. We have identified three different scales:

    - *Intra-data center network:* This scale focuses on the solutions proposed within one data center.
    - *Inter-data center network:* In this scale the concentration is on the solutions implemented for data transmission between data centers. For example in case of virtual machine (VM) migration to another data center or moving data to remote data centers, the energy consumption of the networking component should be considered.
    - *Mixed-data center network:* The suggested solutions in this scale not only consider energy consumption within one data center but also take care of energy efficiency factors between data centers. Energy efficient related decisions have to be made while considering all effective factors.

In this scale it has been tried to combine the energy consumption within one remote data center with the energy consumption of transfer routing path to that data center.

7. **Energy Efficiency Evaluation:** In order to assess the effectiveness of the proposed solutions on energy efficiency, various evaluation methods are employed, which benefit from the means of energy efficiency metrics. Each primary study provides theoretical arguments or statistical/testbed experiments to prove the validity of the proposed solution and the deployed technology. Some primary studies use the existing energy models from the literature as a reference to calculate energy consumption values. Some also generalize and simplify the energy models based on their scope and research question. Another way is to apply external power metering tools, which can profile more accurate measurements.

The meta-model not only introduces summarized classes but also their relations with an assigned name and multiplicity values. For example we observed that *Motivation* plays the initiator role for the adoption of a certain *Strategy* while *Constraint* limits it. The *Solution* class has aggregation relationship with itself because of the "Decision framework" type, which can be derived from a set of other solutions. Depending on the *Scale*, one applicable *Solution* is adopted, which is then implemented by a specific *Technology*. The *Energy efficiency evaluation* class is meant to validate the proposed *Solution* and *Technology* in order to assess the improvements they provide.

3.5 Results

Each class presented in the meta-model shows the primary elements for composing an energy efficient solution. Classes *Motivation* and *Constraint* specify the scope of a solution. The diversity of identified items of these classes in the primary studies does not raise specific observations. Therefore, we skip these two and present the analysis of the classes *Strategy*, *Scale*, *Solution*, *Technology* and *Energy efficiency evaluation*.

3.5.1 Results regarding "Strategy"

Table 3.3 lists all the 11 strategies identified in the primary studies. The "Description" column provides a short summary of the intent of the strategy and how energy efficiency as the ultimate objective will be achieved. The primary studies realizing each strategy are listed and are counted in the last column. We use the number of primary studies in the last column of all the tables to sort the table rows.

The main goal with all strategies is to provide a plan for total energy consumption reduction of cloud-based environments although they differ in their perspectives. For example some try to shape the network traffic ("Traffic patterns" strategy) while some consolidate virtual machines in fewer number of physical machines ("Virtual machine consolidation" strategy). There are two types of relations identified between the networking component and energy efficiency achievement:

- One type starts from energy consumption reduction and then reflects that in the networking component. For example in "Heat minimization" strategy, the temperature increase in different regions of the data center is the trigger to execute the energy saving procedure. If one region passes the heat threshold then the networking related energy saving procedures are also executed to switch the network devices from their active mode to their sleep mode.
- The other type starts from the networking component and then influences the total energy efficiency. "Traffic consolidation" is an example of this type. So first aggregating network traffic onto a fewer number of machines takes place. Then the total energy efficiency is improved in other aspects of the data center.

Table 3.3: Strategies identified in the primary studies

#	Strategy	Description	Solutions	*Layer*	Primary Studies	OCR
1	***Sleeping mode/ Switching off***	This strategy aims to improve the energy efficiency by deactivating idle network devices.	Routing/ Switching, Device, Decision framework, Network architecture	*Device*	[47, 51, 76, 84, 85, 114, 115, 122, 127, 148, 152, 166, 168, 169, 172, 173, 184, 211, 224, 225, 226, 228, 230, 237, 241, 248, 250, 257]	28
2	***Traffic consolidation***	The idea with this strategy is to aggregate network traffic into fewer number of links and devices in order to optimally utilize networking resources and result in higher energy efficiency.	Decision framework, Routing/Switching, Device	*Network*	[51, 84, 130, 168, 169, 184, 226, 228, 237, 238, 250]	11
3	***Virtual machine consolidation***	This strategy aims to aggregate virtual machines into fewer number of physical machines in order to reduce the total amount of energy consumption in data centers. The networking component will be involved at the stage of technical specification and implementations.	Decision framework	*App*	[84, 127, 152, 168, 172, 173, 228, 237, 248]	9

4	***Optical devices***	The goal with this strategy is to replace current electrical networking devices with optical devices, which consume less energy and provide more throughput.	Device, Decision framework, Routing/Switching, Network architecture	*Device*	[53, 103, 125, 128, 129, 131, 132, 140, 240]	9
5	***Energy aware routes***	The selection of the networking path is based on the energy consumption of switches. This can be applied in two ways. Either the switches with less energy consumption will be on the path or the total energy consumption of the path would be kept at its minimum level.	Decision framework, Routing/Switching	*Network*	[65, 114, 131, 132, 192, 193, 237, 257]	8
6	***Traffic patterns***	This strategy takes the traffic patterns into account in order to discover behavior of applications and consequently make intelligent decisions.	Decision framework	*App*	[85, 115, 166, 240, 241, 248, 250]	7
7	***Traffic locality***	In order to save the networking resources, one idea is to localize the traffic in some specific parts of data centers. This way less amount of networking devices will be involved in data transmission, which consequently consumes less energy.	Decision framework, Network architecture	*Network*	[110, 122]	2

8	***Energy aware devices***	This strategy uses modified electrical switches, which are able to increase their energy efficiency by aggregating the traffic into fewer number of ports and putting the idle ports into sleep mode.	Device	*Device*	[51, 230]	2
9	***Heat minimization***	This strategy aims to reduce the total heat in data centers, which consequently will improve the energy efficiency of cloud-based environments. To mitigate the temperature growth in data centers, the load distribution takes place, which also needs involvement of the networking component.	Decision framework	*DC*	[211]	1
10	***Traffic minimization***	The smaller network traffic becomes, the less energy will be consumed in the networking component. Using this strategy the amount of traffic in the cloud-based environment will be reduced, which involves either fewer number of networking devices or the same number of devices but for shorter duration.	Decision framework	*Network*	[76]	1

11	***Green energy***	Green energy has received more attention in cloud-based environments as it is produced from renewable and non-polluting resources. There are some studies doing path selections based on availability of this type of energy.	Decision framework	*DC*	[12]	1

Overall it can be said that the former type focuses on the desired effects namely improvement in energy efficiency, while the latter concentrates on the cause, which is the networking component.

As illustrated in table 3.3, "Sleeping mode/Switching off" is the most frequently used strategy. This strategy tries to make the group of the network devices in the data centers behave more energy proportional to the network traffic load. Studies show that the traffic load in data centers follows a relatively known pattern. For instance, at night the traffic load drops dramatically. This can result in considerable number of idle networking devices, which makes this strategy a good choice to be applied. Some strategies progress further and aim to even increase the number of idle network devices. That is why some strategies, such as "Virtual machine consolidation" and "Traffic consolidation", are investigated in combination with "Sleeping mode/Switching off" that deactivates idle network devices to gain further improvements. Wang *et al.* introduce CARPO [250] as an intra data center scale decision framework, which focuses on network traffic engineering to follow both "Sleeping mode/Switching off" and "Traffic consolidation" strategies.

As it is clear from the table, each strategy is realized with one or more of the solution types. The decision framework solution type is able to realize all but one strategy at different levels of granularity. Depending on the target component of the data center, it is possible for other solution types to realize specific strategies. For example "Energy aware devices" is only realized by a "Device" solution.

For the sake of discussion, we classify the strategies into more general groups, which are shown for each strategy in the "Layer" column of the table. This classification is shown in *italic* format to differentiate it from the codes extracted out of the primary studies. Four groups of strategies have been identified based on the granularity of the target components as: *a*) Data Center layer (DC); *b*) Application layer (App); *c*) Network layer; and *d*) Device layer. Within some strategy groups, there is no clear boundary between the solution domains covered by the strategies. For instance in the "Network layer" group,"Traffic minimization" and "Traffic locality" try to improve on the amount of network flows. The former makes static/dynamic decisions on virtual machine allocations based on the network traffic between virtual machines while the latter concentrates more on the network architecture of servers and switches in the data centers.

As shown in the table, both "Optical devices" and "Sleeping mode/Switching off" strategies are realized by all solution types. Replacing electrical switches with optical ones, which are more energy proportional and whose energy consumption is lower, fades out the need to shut down idle networking devices. Therefore, we have seen no primary study realizing these two strategies at the same time. "Energy aware routes" is another frequently studied strategy that puts the emphasis on the energy consumption of routing paths. The energy efficient routing

paths are established by either selecting the switches that consume less energy or selecting the routing path with minimum total energy consumption. The energy related decisions can be made through a centralized controller, or in a distributed manner, in which each standalone router/switch will select the next hop based on energy efficiency metrics. The next interesting strategy is "Traffic patterns" with the idea of adapting the networking component based on the incoming traffic generated by the running applications. Setting the network devices as active/idle is done in a more intelligent way in this case, since it can be estimated for the coming time t what will be the traffic load. Traffic patterns are taken into account in two ways: static and dynamic. The former uses an initial traffic matrix while the latter collects traffic information at runtime.

3.5.2 Results regarding "Scale"

We have identified three different scales in the primary studies. One is intra-data center scale, which is concerned with the solutions and technologies implemented within one data center. The other two require to adopt solutions that are investigated on the data transfer network between data centers. We call them inter- and mixed-data center scales and point them out as large scales in this chapter. This appellation is not a comparison on the number of virtual machines running in the cloud-based environment rather it is referring to the number of data centers deployed in a cloud-based environment.

It is interesting to study the relation between the solution types and each scale. Table 3.4 lists the solutions presented in inter-, intra- and mixed-data center scales. As it illustrates, most of the primary studies (around 82%) have proposed intra-data center network solutions. The *Network architecture* and *Device* solution types are not investigated for larger scales. Since in multi-domain transfer networks, different areas of administration and ownership exist, these solution types can not be deployed in a centralized consistent way. We have identified two primary studies covering large scale cloud-based environments that propose the "Device" solution type. They concentrate on the "IP over WDM (Wavelength-division multiplexing)" optical networks as their backbone transfer network.

Table 3.4: Solutions provided for different data center scales

#	Solution	Primary Studies	Occurrence
Scale: Intra data center (%81.8)			
1	***Decision framework***	[65, 76, 84, 85, 115, 122, 127, 148, 152, 166, 168, 169, 172, 173, 184, 211, 226, 228, 237, 240, 241, 248, 250, 257]	24
2	***Network architecture***	[51, 53, 76, 84, 110, 115, 122, 125, 127, 128, 129, 140, 166, 172, 173, 224, 225, 228, 240, 241, 248]	21
3	***Routing/ Switching protocols***	[65, 76, 85, 103, 114, 140, 152, 172, 224, 225, 226, 230, 237, 241, 248, 257]	16
4	***Device***	[51, 53, 103, 125, 128, 129, 140, 211, 230, 240]	10
Scale: Inter data center (%11.3)			
5	***Decision framework***	[12, 47, 132, 192]	4
6	***Routing/ Switching protocols***	[12, 47, 132, 192, 238]	5
7	***Device***	[132]	1
Scale: Mixed (Inter & Intra) data center (%6.8)			
8	***Decision framework***	[130, 131, 193]	3
9	***Routing/ Switching protocols***	[193]	1
10	***Device***	[131]	1

From table 3.4 it can be concluded that *Decision framework* is the most widely adopted solution type in all the scales. The table also shows how decision frameworks make use of other solutions. For example [248] is an intra-scale decision framework that makes use of both solution types *Network architecture* and *Routing/Switching*. The permutations of applied solution types in decision frameworks vary in different scales. Extracted from the table, a decision framework for an inter-data center scale is more probable to employ the *Routing/Switching* solution type rather than the *Device* type.

3.5.3 Results regarding "Solution"

We have classified the discovered solutions in the primary studies as the following:

- *Device:* This solution type is provided when focus of the solution is to improve energy efficiency of networking devices. Only network switches and links are put in this category, excluding servers since they are not considered as part of the networking component.
- *Routing/Switching protocols:* The main focus of this classification is to answer the question: how can energy efficiency be improved in network connections namely routing and switching? Redesigning routing/switching algorithms or using current protocols but in more intelligent ways are some examples of this type realized in the literature.
- *Network architecture:* This solution type offers the optimum topology for network devices and network links. As mentioned earlier, currently developed data centers are "richly-connected". Although this helps with the achievement of reliability and availability qualities, it does not ensure high energy efficiency, which should be investigated to provide more improvements.
- *Decision framework:* It can be a combination of other solution types or be implemented as a standalone solution. It provides a framework to make static/dynamic decisions based on the information collected from data centers through a centralized/distributed method. We consider decision frameworks as self-adaptive software systems, which can benefit from other solution types to monitor the infrastructure at runtime and recover from runtime changes with reconfiguring the networking component.

3.5.4 Results regarding "Technology"

The ideal situation for a primary study is to propose a generalizable solution that can be applied to any cloud-based environment. Our primary studies mostly have

suggested a general solution. Afterwards, they have evaluated the proposed idea with specific IT infrastructure like a certain type of device or network architecture. We define technology as implemented tools or specifications that are required to realize the solution, and not the details regarding the evaluation phase. Tables 3.5-3.7 detail the technologies that implement the *Device*, the *Routing/Switching* and the *Network architecture* solution types.

Device Technologies

Table 3.5 indicates the technologies implementing solution type "Device". Column "Type" illustrates the methods targeting devices in *italic* to be unlike other columns showing the primary studies codes. Two methods are suggested for the network devices; either replacing the current devices with optical ones or improving the current ones to be energy aware. 8 out of 10 technologies make use of optical devices. Since optical networks are recognized as a remarkable alternative to traditional transfer networks, in the cloud-based environments, it benefits from the energy efficiency features of optical devices. Different kinds of optical routers, optical switches, and other optical devices are examined to explore the potentiality of data centers to be more energy efficient. For example, [53] has suggested space-time interconnection optical devices for data center network architectures. Most primary studies of this type adopt the existing optical technologies in a smart way to minimize the energy consumption of data centers at different scales.

Table 3.5: Technologies implemented for the Device solution type

#	Technology	Scale	Description	*Type*	Primary studies	OCR
1	***Optical interconnect devices***	Intra	This technology refers to all the devices used in the optical networks such as couplers, SOAs, WSSs and MEMS-switches.	*Optical*	[103, 129]	2
2	***Using optical switches for "IP over WDM" backbone network***	Inter, Mixed	This technology is used to reduce the energy consumption of transfer networks. "IP over wavelength-devision-multiplexing" networks make use of optical bypasses, which aggregate the incoming traffic from access routers and send it to light-paths. This technology is suggested for inter-data center networks.	*Optical*	[131, 132]	2
3	***Hybrid optoelectronic router***	Intra	"Hybrid optoElectronic router" performs a combination of optical and electrical switching technologies. It contains a shared buffer, which can be investigated to provide different services in optical packet switched networks such as multicast and QoS. This technology is used in combination with other solution types like *Network architecture* and *Routing/Switching*.	*Optical*	[140]	1
4	***CAWG***	Intra	CAWG stands for cyclic arrayed waveguide grating, which plays the role of aggregation switches in the optical network architecture. It routes the incoming wavelengths to the output ports.	*Optical*	[125]	1

5	***WDM-OFDM ToR Switches***	Intra	These switches are used in optical network architectures as an alternative for conventional switches.	*Optical*	[125]	1
6	***STIA***	Intra	Space-time optical interconnection architectures introduce new optical switching technologies. They are more scalable and more energy efficient than other optical switches. They do the switching procedure using space and time domains.	*Optical*	[53]	1
7	***Using optical devices for WDM PON***	Intra	To create a "passive optical network", extra optical devices are needed in order to keep the network up and running. Optical WDM Transceivers and AWGR (arrayed waveguide grating router) are some examples of deployed optical devices.	*Optical*	[128]	1
8	***OXC***	Intra	In some layers of network architectures in data centers, this technology is used to implement an optical network. OXCs, standing for optical cross connects, are mainly used to reconfigure the wavelengths according to bandwidth demands.	*Optical*	[240]	1
9	***eAware ethernet switch***	Intra	Using this technology, switches and their ports are put in the idle state based on their queue length and utilization percentage of the device in order to save energy.	*Electrical*[230]		1
10	***"Merge network" switch***	Intra	This technology emphasizes on the traffic aggregation into fewer number of output ports based on the utilization rate of ports.	*Electrical*[51]		1

There are two implementations focused on improving the current electrical switches (#9 and #10). The main idea with these technologies is to make use of fewer number of ports in a switch to reduce energy consumption. The incoming traffic load can be distributed into fewer number of output ports according to optimized utilization rates. The idle ports and gradually the switch itself can be put in the sleeping or the low power mode. *Merge Network* switch (#10) not only deactivates idle ports but also tries to increase them by traffic aggregating methods. The advantage with these implementations is that energy awareness is distributed all over the data center network and each switch makes locally optimal decisions (performing as greedy algorithms). Also there is no single point of failure in such implementations. However, from a distributed and holistic perspective these local optimizing solutions might introduce some overhead in throughput and other efficiency qualities. Besides, they can not ensure to utilize the most energy efficient set of active ports in the data center. These two technologies are suggested as standalone solutions but other device technologies can be used in combination with other solution types like decision frameworks. In total we have 14% of primary studies that propose a standalone device solution.

Routing/Switching Technologies

Table 3.6 describes all the technologies regarding the "Routing/Switching" solution type. Primary studies investigating this solution type propose algorithms/protocols by which, data center VMs will be able to transfer data in an energy efficient way. There is a diverse range of ideas displayed in the table that differ in several aspects, such as the target network layer and the purpose of use. For example Green VLAN (#7) is a specific implementation for layer 2 in the OSI model, while ECMP (#2) details a multi-path routing protocol for layer 3 in the OSI model. The major number of technologies goes for layer 3 rather than layer 2 and only 2 out of 8 technologies are implemented for layer 2 (Green VLAN and ExP).

Two types of Routing/Switching technologies are proposed in the primary studies. They are shown in the "Impact" column of the table in the *italic* format in order to be distinguished easier from other columns, which illustrate the codes that emerged from the primary studies:

- **Direct**: The Routing/Switching technology is not used along with other solutions and it provides energy efficiency improvement in the networking component itself as a standalone solution. The improvement may come from either the least energy consuming routes or transferring the traffic through fewer number of network devices.
- **Indirect**: The Routing/Switching technology is used in combination with other solutions and it does not necessarily need to be an energy aware

protocol. For example, the energy consumption related decisions can be delegated to a decision framework and the routing protocol performs as it did before. That is why we see ECMP in the table, which is a non-energy aware routing protocol. Our primary studies show that Routing/Switching technologies have been used in combination with the *Device* and the *Decision framework* solution types.

Another reason to make use of non-energy aware Routing/Switching algorithms is because they are so widely used in cloud-based environments. In this case, the primary study aims to increase the energy efficiency of a cloud-based environment with having existing non-energy aware technologies deployed. Therefore an energy-aware decision making component is added to the system as a decision framework, which manages all the energy efficiency related concerns.

Moreover, the energy efficient route is more effective when it is reconfigured at runtime due to the changes occurring in the incoming load traffic. Some primary studies, such as the one proposed by Fang *et al.* [85], make use of traffic patterns, which include information of the traffic flows in the data center in order to make more accurate energy efficient related decisions.

EAR technology [224] is one of the most frequently deployed ones in our list. This algorithm makes a trade-off between energy efficiency and other efficiency factors based on the quality requirements. The idle networking devices are eliminated gradually in steps and in each step it is checked whether the output network still responds with the expected throughput or not.

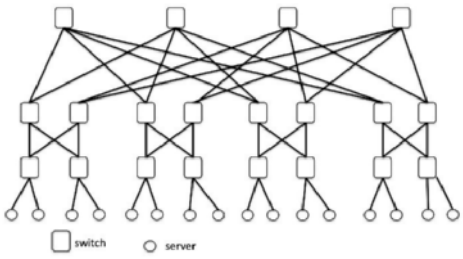


Figure 3.3: A fatTree network architecture with 4-port switches

As the table presents, Routing/Switching technologies are mostly targeted for intra-data center scale; however, inter-data center scale has got the attention of standalone Routing/Switching solutions recently. *Max-Flow Min-Energy* technology [238] is an example of an inter-data center solution.

Table 3.6: Technologies implemented for the Routing/Switching protocols solution type

#	Technology	Scale	Description	*Impact*	Primary studies	OCR
1	***EAR***	Intra	In the "Energy aware routing" protocol, number of switches and links will be minimized. Since the energy efficiency and other efficiency qualities (like performance and network delay) usually have an inverse relationship, in this protocol the eliminating process of network devices goes on until the network throughput decreases to a pre-specified threshold value. So the energy efficiency in the data center networks is achieved while the network throughput threshold is not violated.	*Direct*	[224, 225, 226]	3
2	***ECMP***	Intra	"Equal-cost multi-path" routing protocol's focus is to make use of several routing paths, which have equal costs. The forwarding decisions on sending the data through more output ports, are made by each router. In data centers, which implement richly connected network architectures, ECMP is used to distribute the load over multiple paths and also increase the total bandwidth for that flow.	*Indirect*	[85, 230]	2
3	***Shortest Path***	Intra, Inter	According to this routing algorithm, always the shortest path between two nodes is selected.	*Indirect*	[47, 76]	2

4	***EER***	Intra	The idea behind the "Energy efficient routing" algorithm is to use the minimum number of aggregation/core switches in a data center based on the traffic load at a specific time interval. Also by using multi-path routing, the traffic flows will be load-balanced among chosen network devices.	*Direct*	[248]	1
5	***Max-Flow Min-Energy***	Inter	Using this technology, the data transfer among remote data centers is done by aggregating the traffic flows as "trunks" and then routing them through the least energy consuming paths.	*Direct*	[238]	1
6	***HPR***	Intra	HPR stands for "High performance routing" and its key idea is to provide the highest network throughput for each flow. The procedure is done in steps. First the routing paths for each flow are identified, then the paths with lowest number of assigned flows will be selected. So depending on the order of processed flows, the final set of routing paths would vary.	*Indirect*	[225]	1

7	***Green VLAN***	Intra	This technology tries to organize VLANs in a more energy efficient way. It starts by checking out each VLAN's impact on energy consumption according to some constraints (for example the number of hosts belonging to that VLAN and the load of broadcast traffic in that VLAN). If they do not satisfy the requirements, they will be split into several VLANs. Therefore each two hosts are ranked based on the amount of traffic between them and the path linking them together. According to the calculated rankings, new VLANs are categorized.	*Direct*	[114]	1
8	***ExP***	Intra	Express Path is a routing method introduced for optical networks and the routing process takes place based on flows and not packets. Since all the packets of a same flow contain the same values of identification parameters such as source address, source ports, destination address and destination ports, then only routing the first packet of each flow would be sufficient and the rest of the packets will follow the same route.	*Indirect*	[140]	1

Network Architecture Technologies

Given the high number of servers and switches in data centers, it is important to design a network architecture in a flexible way to be functional in the presence of quick bursty incoming load. Table 3.7 presents the network architectures studied in our primary studies.

Currently many richly-connected network architectures are deployed in data centers such as FatTree, BCube, VL2 and DCell. As the table shows, FatTree is the most widely investigated topology in our primary studies. It is a popular, regular, symmetric and scalable network architecture. The organization of network devices in FatTree is based on a switch-centric approach, providing 3 layers of switches: core, aggregation and ToR. Figure 3.3 shows a FatTree network architecture with 4-port switches.

The primary studies illustrate that both "Device" and "Decision framework" solution types are implemented in combination with the "Network architecture" solution. FlattenedButterfly and Hybrid WDM PON network architectures [51, 53, 128] are examples of the network architecture solution types used in the device solution type. The network architecture solution type is used mostly as a supporting role along with the decision framework solution type. Only two primary studies focus on network architecture as a standalone solution [53, 125]. Extracted from the table, 6 out of 14 network architectures are specialized for optical interconnections. Although this number is almost half of the total number of identified technologies for this solution type, this ratio is not seen in the number of primary studies providing optical technologies. It is because the energy efficiency in optical data center networks is nearly optimum and it is hard to get extensive improvements on that. The network architecture technologies are proposed only for intra-data center scale, because of different ownerships and maintenance methods in each domain.

Table 3.7: Technologies implemented for the Network architecture solution type

#	Technology	Scale	Description	Primary studies	OCR
1	***FatTree***	Intra	FatTree network architecture is a switch-centric physical topology, richly-connected and scalable. In this architecture, the network switches are classified in 3 tiers: core, aggregation and ToR. If n is the number of ports in a switch, in a FatTree architecture there will be $(n/2)^2$ core switches with n ports, n pods with n switches having n ports ($n/2$ aggregation switches + $n/2$ ToR switches).[110]	[53, 76, 84, 110, 115, 127, 166, 173, 224, 225, 228, 241, 248]	13
2	***BCube***	Intra	BCube network architecture is a server-centric physical topology, which can be easily extended in a recursive manner. If k is the level number and n shows the number of ports in a switch, BCube(k) consists of n BCube(k-1) architectures, which are connected by n switches having n ports.[162]	[110, 122, 225]	3
3	***VL2***	Intra	VL2 network architecture is a switch-centric physical topology. There are bipartite-like connections between core and aggregation switches in this topology. VL2 uses load balancing techniques (Valiant Load Balancing) to distribute the load from aggregation switches to core switches.	[84, 172]	2
4	***Flattened-Butterfly***	Intra	The main idea with this network architecture is to minimize the number of hops for each route. It is scalable and can be extended in a recursive manner. As it is studied before, this topology is recognized to be more energy efficient than FatTree. [51]	[51, 53]	2

5	***DCell***	Intra	DCell is an example of server-centric data center network architectures, which can be extended in a recursive way. If k shows the level number and n(i) shows the number of servers at i(th) level, DCell(k) consists of (n(k -1) + 1) DCell(k-1) architectures that are connected only through servers.	[110]	1
6	***BalancedTree***	Intra	In a balancedTree, there is only one switch as root having n ports. The idea with this architecture is to distribute the servers between switches uniformly, which all are similar in number of ports. The resulted topology looks regular and symmetric but it has the possibility of single point of failure.	[53, 110]	2
7	***VL2N-Tree***	Intra	VL2N-Tree network architecture is a combination of traditional network architecture 2N-Tree and VL2. So there is a bipartite graph between core switches and aggregation switches and the rest of connections (among aggregation switches and ToR switches and servers) follow the rules of a 2N-Tree data center network architecture.	[84]	1
8	***Generalized flattened butterfly***	Intra	Generalized flattened butterfly is a hierarchical extendible network architecture, which tries to benefit from the "minimum hop" feature of Flattened butterfly network architecture and the "high bandwidth" feature of DCell network architecture.	[240]	1
9	***Hybrid WDM PON***	Intra	The main focus of this network architecture is to replace all inter-racks links with optical interconnects in a Fat-Tree architecture. In this scenario, aggregation switches are set as Optical Link Terminators and ToR switches are used as Optical Network Units.	[128]	1

10	***MIMO OFDM optical***	Intra	In a MIMO OFDM data center network architecture, traditional ToR switches will be replaced with WDM-OFDM ToR switches and all aggregation switches will be replaced with one CAWG, which does the routing procedure in a cyclic manner.	[125]	1
11	***Torus***	Intra	Torus provides a 3-dimensional network architecture, which in each dimension all switches are connected together. It is linearly scalable and with low latency.	[140]	1
12	***AWG-based architecture***	Intra	It is a kind of optical network architecture, which makes use of AWG to route incoming wavelengths to output ports. There are different versions of this architecture introduced recently.	[129]	1
13	***WSS-based architecture***	Intra	Each ToR switch sends out its traffic using optical transceivers to upper layers namely MEMS switch.	[129]	1
14	***B&S architecture***	Intra	"Broadcast and Select" architecture is another example of optical data center network architectures providing low delays.	[129]	1

Decision Framework Technologies

The "Decision framework" solution type is considered as a self-adaptive software system that can either be composed of other solution types or be implemented as a standalone solution. Decision frameworks aim to control energy consumption of the networking component by collecting relevant information from different components. Each decision framework follows a specific approach and accordingly other solutions and technologies are selected. Different permutations of the solution types are selected for each decision framework, which we categorize as the following:

- **Traffic Engineering:** In order to minimize the energy consumption of the networking components, decision frameworks following this approach try to shape the network load. Decision frameworks collect and correlate context-dependent information as a basis for making static/dynamic decisions. Based on how the traffic engineering is done, different types of strategies would be realized. For example [125] proposes a decision framework, which collects statistical information from network switches. Then based on utilization rates, a network subset with minimum number of networking devices and minimum needed capacity is selected to transfer the traffic. In this case the traffic is consolidated and the idle networking devices are deactivated, which means "Sleeping mode/Switching off" and "Traffic consolidation" strategies are realized. Differently we see in NESS [85] that network devices are put in the sleeping mode according to pre-learnt traffic patterns, which shows "Sleeping mode/Switching off" and "Traffic patterns" strategies are realized.

- **VM Assignment/Migration:** Using this approach decision frameworks try to place/reallocate the virtual machines in order to aggregate them into fewer number of physical machines. In the VM migration process the energy efficiency of the networking component is addressed by decision frameworks. "VM consolidation" strategy is realized by this type of decision frameworks. To provide an energy efficient networking solution this approach is used along with other two introduced approaches.

- **Routing:** According to this approach, decision frameworks try to choose the best routing path, which is the route with the minimum energy consumption compared to the other possible routes. The routing decisions can also be made using an overlay virtual network, which abstracts physical links in virtual links to provision network services with maximum energy efficiency. This approach is aligned with the "Energy aware routes" strategy. For instance DCe-CAB [192] is a decision framework for the inter-data center scale and places virtual machines in different data centers. A data

center is selected if not only the energy consumption of the data center satisfies quality requirements but also the routing path to that specific data center has the shortest path and the maximum efficiency in quality requirements (minimum delay, minimum energy consumption).

Decision frameworks can follow more than one approach. One more remark is that decision frameworks can be scaled up for large scale cloud-based environments. Intra- , Inter- and mixed-data center scales can be supervised by decision frameworks from an energy efficiency perspective. Overall in the case of large scale cloud-based environments, a decision framework is an excellent choice because of their management possibilities for a network of heterogeneous data centers.

3.5.5 Results regarding "Energy Efficiency Evaluation"

Each proposed energy efficient networking solution has to be evaluated and analyzed. Primary studies present the energy saving results for each solution while calculating the impact on other efficiency factors like network delay and network throughput. It is not a trivial task to measure, model and optimize energy consumption of large scale data centers. Therefore 72% of experiments have been done in simulations. Table 3.8 represents all the methodologies used for the energy efficiency evaluation process and also specifies the corresponding scale of the target cloud-based environments.

Table 3.8: Energy efficiency evaluation methods identified in primary studies

#	Energy efficiency evaluation method	Scale	Description	Primary studies	OCR
1	***Simulations***	intra, inter, mixed	There are plenty of simulators used to simulate cloud-based environments. Simulations become even more convenient when the implementation of the solution does not exist yet. For example, solutions to improve current network switches can initially be evaluated using simulation experimentation.	[12, 51, 65, 76, 84, 85, 110, 114, 122, 127, 128, 130, 132, 140, 152, 166, 168, 169, 172, 173, 192, 193, 211, 224, 225, 226, 228, 230, 237, 238, 240, 248, 250, 257]	34
2	***Empirical experiments in small scale***	intra, inter	In this methodology, the experimental set-up is prepared in hardware testbeds in small scales. The number of nodes and links in this kind of set-up is much lower than what is utilized in realistic scenarios.	[47, 53, 115, 127, 168, 184, 241, 250]	8
3	***Numerical analysis***	intra, inter	In this way the output results from a literature study are used as a reference for reasoning purposes.	[103, 129, 148]	3
4	***Empirical experiments in production data centers***	mixed, intra	The best way of evaluating a solution is to implement it in realistic scenarios. By using data centers to analyze the solution, the results would be more representative and accurate. But it is not always easy to get a production data center for this purpose.	[125, 131]	2

Every evaluation method has its limitations and advantages. Using more than one evaluation method could help the derived conclusions be more concrete. For instance [127] evaluates their decision framework in both simulations and empirical experiments.

Primary studies also have shown how much they have gained in terms of improvement in energy efficiency. eAware [230] as a modified version of an Ethernet switch and as an example of the device solution type, can save the total energy consumption of network switches from 30% to 50% while it introduces only 3%-20% increase in the end-to-end packet delay. The proposed MIMO OFDM DCN architecture [125] has been evaluated by means of large scale empirical experiments and it can provide up to 25% energy efficiency improvement in comparison to the equivalent network architecture with commodity electrical switches. GreenDCN decision framework [248], which is an intra-data center scale solution, increases the energy saving of the networking component up to 50%.

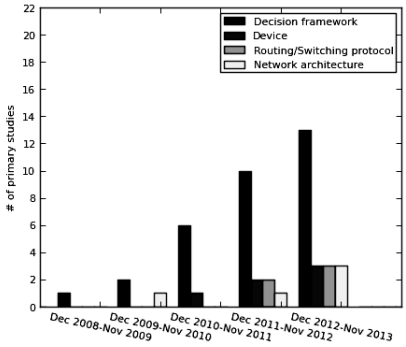


Figure 3.4: Yearly distribution of each solution type, from Dec 2008 to Nov 2013

ElasticTree decision framework [115] as a pioneer in making use of OpenFlow technology in 2010 for energy efficiency improvements in the networking component (more detail in section 3.6), is able to save the energy consumed by the networking component up to 25%-40%.

3.6 Discussion

To discover the trends in energy efficient networking solutions we have split the articles in 5 yearly periods (the last period being Dec 2012-Nov 2013).

As seen in figure 3.4, the topic has been brought up from late 2008, with an increasing number of primary studies each year to testify to a growing interest. We can also see that the greatest attention has been paid to decision frameworks. Although the other three solution types (Device, Routing/Switching and Network architecture) are proposed less often, still they show an increasing trend.

The very first article on this topic is a decision framework from the first time period for intra-data center scale [169]. Initially, decision frameworks were the most natural solution: they simply required extensions to existing optimization frameworks, where the energy efficiency was an additional quality requirement.

The continuous attention to decision frameworks in the course of the years is due to their flexibility: they provide a global insight and collect relevant information from all over the data center. They allow to make accurate energy-related decisions according to correlated information.

Figure 3.5 shows the yearly distribution of solutions at different scales. We see that the research done within one data center is still the main focus of the studies, partly due to the easier manageability in these environments.

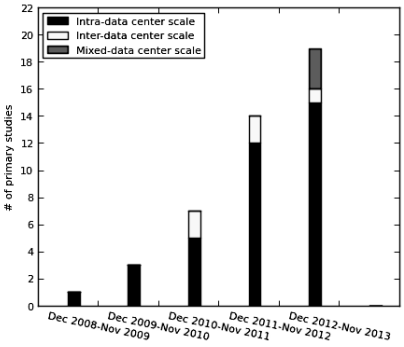


Figure 3.5: Yearly distribution of solutions in different scales, from Dec 2008 to Nov 2013

The research on inter-data center scale started in December 2010, while solutions for the mixed-data center scale have been introduced more recently in December 2012. Achieving energy efficiency at these scales is basically more challenging. Due to the emergence of new business models and new requirements for energy efficiency, we expect to see a growing interest in large scale solutions, especially mixed-data center scale in the coming years. This will likely result in a decreasing interest in pure inter-domain solutions, and this trend is already visible in figure 3.5.

Decision frameworks can use other solution types in their design. We identify different permutations of solution types that decision frameworks use. Figure 3.6 shows that over the years decision frameworks have become more diverse. We also see that *Routing/Switching* is the most widely deployed solution type, individually and in combination with others. The next most frequent solution type is the *Network architecture solution*, usually with FatTree implementation. The *Device* solution type is instead adopted in only three decision frameworks. The combination of *Device* and *Routing/Switching* solutions appears twice in our studies, and it is used in large scale cloud-based environments, concentrating on optical "IP over WDM" backbone networks.

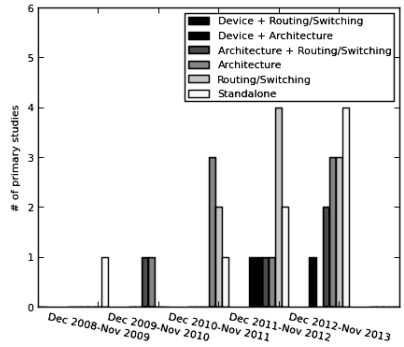


Figure 3.6: Yearly distribution of solution types used in decision frameworks, from Dec 2008 to Nov 2013

There are some decision frameworks that are not a combination of other

solution types. We call them "standalone" decision frameworks. Standalone decision frameworks target non-energy aware infrastructures. They deploy other techniques to collect the required information and to take action.

For example they might use traffic pattern information to make more accurate decisions on putting network devices in sleep mode. Figure 3.6 shows an ascending trend for standalone decision frameworks in the last years.

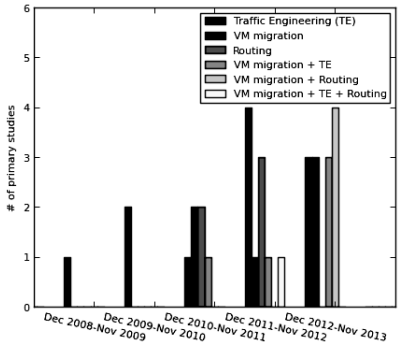


Figure 3.7: Yearly distribution of approaches used by decision frameworks, from Dec 2008 to Nov 2013

Figure 3.7 displays how decision frameworks make use of different approaches to improve the energy efficiency of data centers. We recall from section 3.5.4 that Decision Frameworks can use different approaches: Traffic Engineering (TE), VM Assignment/Migration and Routing. From the figure we see that the Traffic Engineering approach is the most frequently used one, individually and combined with others. Also VM Migration is frequently used to minimize the network traffic by grouping virtual machines into fewer number of physical machines, implicitly reducing the energy consumption of the networking component. The virtual machine migration process can include application dependencies as well.

Using the approaches only focusing on the networking component or the computation domain has a risk of missing some pieces of the big picture to ensure energy efficiency. The combination of the VM Migration approach and one or two of the networking related approaches (either Traffic Engineering or Routing) in decision frameworks, provides broader view of changes that need to be made in cloud-based environments. That is why the combination of "TE + VM Migration" and "Routing + VM Migration" have been the most widely used approaches recently. Interestingly, the number of decision frameworks using the "Routing" approach has dropped to zero from December 2012, while the combination of "VM migration + Routing" approaches has been introduced.

Decision frameworks additionally make use of other supporting techniques to implement approaches. Supporting techniques help them extend their collected information domain and improve the efficiency of taking actions. Programmability of the networks and traffic pattern discovery are some of the example techniques used by decision frameworks. The network traffic originated from applications will produce a traffic pattern. By taking this collectable information into account, it will be clear at any given time that what decision would fit the best according to the distribution of traffic load.

Decision frameworks benefit from the programmability feature of the software-defined networks. This type of networks can be programmed because the data plane and the control plane of the network switches are separated. An external controller will do the "updating flow table" process for each switch. Software defined networking protocols, namely OpenFlow, make the communication between the controller and the OpenFlow-enabled switches possible. Using OpenFlow, the controller is able to schedule flows dynamically/statically between the network switches. Decision frameworks deploy the OpenFlow technique in order to apply required dynamic changes in data center networks with the means of an OpenFlow controller.

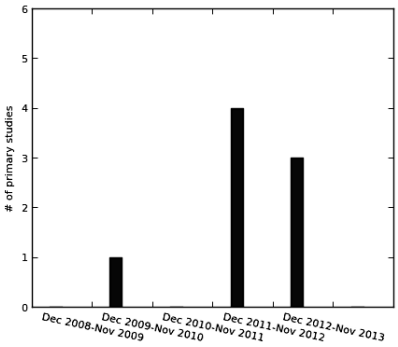


Figure 3.8: Yearly distribution of OpenFlow technique adoption by decision frameworks, from Dec 2008 to Nov 2013

We have 8 primary studies providing solutions for intra-data center scale that investigate the OpenFlow technique to flexibly control the traffic and operate the flow scheduling. 6 out of 8 decision frameworks follow the Traffic Engineering approach and the rest investigate the Routing approach. As figure 3.8 presents, decision frameworks made use of the OpenFlow technique at early 2010 for the first time. Recently the usage of this technique has increased remarkably and we expect it to grow even more in the coming years.

3.7 Threats to Validity

There are some uncontrolled elements that could change the conclusions we draw and threaten the accuracy and validity of our findings. The potential validity threats to our study are:

a) *Subjective analysis:* Since the study selection phase is mainly conducted by one researcher; there might be possibilities of biased subjectiveness in finding primary studies. Defining extensive and clear selection criteria through a systematic protocol helps to mitigate this threat.

b) *General applicability:* Another threat to our study is the generality of identified solutions and the extent they can apply to cloud-based environments. We included studies that have been peer-reviewed. This, of course, does not mean that works from industry are not relevant; on the contrary, we plan a

follow-up study with the same research question but intended to survey the state of practice to discover if both domains are aligned together. In spite of our academic focus, we have observed that 17% of the primary studies have authors from industrial affiliations. This suggests some non-negligible mingling of academic research and industrial solutions. On the other hand, the evaluation phase in our primary studies mostly makes use of industrial-driven datasets.

3.8 Emerging research directions

We have analyzed the primary studies to discover in which direction current research is evolving. In the following we list some of the future challenges and opportunities we identified:

- Decision frameworks: Our findings show that there has been an increasing focus on this solution type and we expect to see even more evolvement of decision frameworks in the coming years. Not only they are able to apply other solution types but also they can benefit from information collected with higher level components in the data center. For example decision frameworks can make energy efficient decisions based on behavior of applications in the environment. Also, we have seen more attention to standalone decision frameworks recently, which will be even more because of the accurate data that can be collected from the application layer.

  It is interesting to note that decision frameworks have evolved from one-aspect approaches to multiple-aspect approaches to get data as accurate as possible from the components. Therefore, the combinations of "TE + VM Migration" and "Routing + VM Migration" approaches have higher chances to be investigated. Decision frameworks can benefit greatly from the advances in artificial intelligence, which aim to maximize optimization objectives.

- Large scale cloud-based environments: Recently large scale environments have attracted more attention from the researchers. However, energy efficient solutions in large scale environments are challenging to implement. Therefore, academia and industry can collaborate more on the infrastructure provisioning aspect.

- Software-defined networking: According to the Gartner report "Hype Cycle for the Telecommunications Industry, 2014" [39], the SDN field is in phase "Sliding into the Trough". As they define this phase, there is still some time for SDNs to become more mature. Our results show the increasing

importance of SDNs in data centers. SDN is the key to shorten the distance between hardware and software in cloud-based environments. This feature can help decision frameworks significantly in order to get accurate information from the networking component and reconfigure them on the fly.

3.9 Conclusion

This literature review outlines the state-of-the-art in energy efficient networking solutions in cloud-based environments, which is the answer to our RQ2.1, namely "What are energy efficient solutions for networking in the cloud?". An original contribution of our effort is that, for the first time in this research field, we have followed a systematic literature review (SLR) method to be as objective as possible to select our primary studies. By including the SLR protocol, the chapter provides the instruments necessary to replicate the study in the future or eventually extend it to cover new aspects, e.g. "smart green networks". In the latter case, it would require to extend the query. For the example research identified above, this would require the inclusion of "smart" as a keyword.

All the primary studies we have analyzed clearly demonstrate a growing attention to the problem and a lively and dynamic research space. Our findings show that the so called "Decision Framework" is the most frequently investigated solution type to accomplish the energy efficiency goal. Decision frameworks are in fact self-adaptive software solutions that show a continuous increase in number and diversity over the years. The most promising approaches used by decision frameworks are combinations of the computation related approach "VM migration" and the networking related approaches "Traffic engineering" or "Routing". The main advantage of decision frameworks compared to other solutions is their ability to use other supporting techniques such as programmability of networks and traffic pattern discovery.

Another important observation is that although the majority of the primary studies target solutions for intra-data center scale, recently we have seen an increase in the number of large scale solutions, namely inter- and mixed- data center solutions.

We expect that the focus on using network programmability in large scale environments will continue to grow in the coming years, as a response to the increasing size of cloud-based environments. According to Gartner research in 2014 [39], the SDN field will be at the phase of "Plateau of Productivity" in 5 to 10 years, which is a confirmation to our conclusion about the growing attention to this topic. However, it needs to be investigated if SDNs can provide higher energy efficiency in the data centers.

In our next chapter, we aim to examine to what extent SDNs can be beneficial in data center networks to make accurate energy-related decisions.

4

Case Study 1: Energy Efficient Scheduling in Software-Defined Networks

Software-defined networks have been proposed as a viable solution to increase the energy efficiency of the networking component in data centers because of their programmability features. Scheduling algorithms are implemented in the control plane of software-defined network devices to plan runtime reconfigurations. In this chapter, we propose a number of linear programming scheduling algorithms to answer our RQ2.2. We evaluate our algorithms in simulations in terms of power saving and time to complete. All our algorithm variations outperform the shortest path scheduling algorithm, our baseline on power savings, depending on the instrumented power models. We show that in FatTree networks, where switches can save up to 60% of power in their sleeping mode, we can achieve 15% minimum improvement assuming a one-to-one traffic scenario. Two of our algorithm variations favor performance over power saving and still provide around 45% of the maximum achievable savings. Therefore, different software systems with different quality requirements can still benefit from our algorithms.

4.1 Introduction

Networking devices are one of the main contributors to power consumption in data centers. However, improving on their total power efficiency is not a trivial task. From the data center provider perspective, power efficiency in the network should be weighted against other quality requirements such as its performance, reliability and availability. Given the statistics on the average utilization rate of data centers, which is around 30% of the peak hours [164], we see room for improvement in the power efficiency of the networking component.

Nowadays, software-defined networks (SDN) are replacing traditional networks with the means of the programmability of the underlying network. It is

estimated that the adoption of SDN will grow exponentially by 2021 [41]. Given SDNs wide range of applications and their rapid adoption rate in data centers, they are often chosen as part of power efficient solutions. In Chapter 3, we studied the existing energy efficient networking solutions and we observed a growing trend in using SDNs to ensure energy awareness. The existing energy efficient networking solutions are implemented as flow schedulers that fulfill user-defined quality requirements while considering energy consumption improvements. However, the effectiveness of SDNs with respect to energy efficiency is not fully studied and it is not determined how the programmability of networks can be an added value in this matter. Besides, scalability and performance cost of energy efficiency optimization are still open questions.

Linear programming algorithms have been recently deployed to perform as a scheduler for the incoming flows on the available paths [47, 115, 127, 130, 131, 152, 250]. Finding the optimal solution, e.g. mapping the flows to the paths, is an NP-hard problem. In this chapter, we find an alternative to reduce the complexity of the problem. We propose 4 linear programming algorithms that differ in their objective functions. We study each algorithm to see to what extent they impact other quality requirements such as performance. Our main contributions are:

- We derive four different flow scheduling algorithms that take into account a number of constraints to provide either highest throughput or highest power efficiency.
- We implement a modular decision framework to evaluate our algorithm variations. It collects statistical information from the network and schedules the existing/new flows accordingly.
- We support our hypothesis with a number of simulation experiments and we compare our results with a baseline, namely the shortest-path scheduling algorithm.
- We assess to what extent the scalability quality requirement is fulfilled by performing experiments in network of different sizes, different numbers of flows, and different characteristics of switches.

The rest of the chapter is structured as follows. Existing related work on power efficiency solutions for software-defined networks is discussed in section 4.2. In section 4.3, we present our scheduling algorithm and its implementation. We describe our evaluation decision framework and its components in section 4.4. Sections 4.5 and 4.6 present the simulation scenarios and experimental results, respectively. Our findings and observations are discussed in section 4.7. Finally, the chapter is concluded and directions for future work are introduced in section 4.8.

4.2 Related Work

Power and energy efficiency of data center networks has often been a by-product of optimization strategies targeting other quality requirements. For instance, energy-aware VM placement has attracted more attention in data centers compared to the flow scheduling in the underlying network. However, there is still some effort spent on power savings in the network with traffic consolidation or traffic locality [84, 130]. For example, the frameworks introduced in [247, 249], minimize the number of active racks in the data center networks by consolidating VMs in a fewer number of racks. VM placement has been identified as a routing problem in [127] and it has been combined with network optimization. Solely focused on the networking component, [166] introduced a distributed flow scheduling scheme suitable within a data center network. Differently, in this chapter, we focus on the runtime tradeoffs between network-related qualities, running software qualities, and power saving objectives. Our findings can help software engineers to develop the best fitting optimization algorithms for running software systems with a clear overview of the pros and cons of their algorithm choices.

Fang, *et al.* in [86] study different network architectures other than FatTree, namely, 2N-Tree, VL2 and BCube from the power saving point of view. L. Gyarmati, *et al.* design a comparison study on energy consumption of BCube, DCell, FatTree and balancedTree network architectures [110]. As we will discuss in Section 4.4.1 the decision framework we developed allows to define arbitrary topologies as input, and as such will allow comparisons between different data center architectures. We have selected the FatTree topology, a very well known three-tiered data center network architecture, to evaluate the effectiveness of our algorithms.

There are other studies that, like ours, deploy software-defined networks in order to apply the energy-aware changes decided by their decision frameworks or applications. The pioneer work in this era, ElasticTree [115] designs a centralized decision framework to turn off the idle network devices. A disadvantage of this model is that the multi-minute booting time of switches makes handling bursty traffic appropriately more difficult. Differently, we use in our simulation the sleeping mode of switches; this takes much shorter time (around 1s) to turn the switches back on, while still providing significant amount of power savings. There are similar approaches that put idle network devices into the sleeping mode or turn them off by implementing heuristic scheduling algorithms [224, 250, 257]. They mostly provide the traffic matrix as an input to the scheduling algorithm, which is not always a realistic scenario. Contrarily, we focus on real time flow requests while keeping the scalability quality requirement in mind, as we will discuss in Section 4.3.4.

4.3 The Flow Scheduling Algorithm

To enable communication between nodes in a SDN, a scheduling algorithm needs to select the route first. Such an algorithm can be energy-agnostic and update the flow tables of the switches, regardless of energy consumption of the route and its throughput. Shortest-path finding approach (SP) is such an example as it does not retrieve any bandwidth/energy related information from the infrastructure and consequently can not make smart decisions based on this. We use SP and an extension of it as a baseline to compare with the quality metrics of our algorithms. In order to design our power-savvy scheduling algorithm, we deploy the Integer Linear Programming model, which defines the problem as a linear function of different variables and the values selected for variables need to meet the limitations of pre-defined constraints. Since our focus is on scheduling the incoming network traffic in the most power efficient ways, our assumption is that the VMs are allocated beforehand. Also, no initial traffic matrix is assigned to the data center and the incoming load is routed in realtime. This makes scalability a very important quality requirement in our scheduling algorithm design.

4.3.1 Objectives

There are three objectives that can be considered when scheduling traffic flows:

- **Minimize the power consumption of the data center network:**

  To minimize the total power consumption, there are multiple options. One is to turn off the idle network devices, but this takes a significant amount of time for the devices to boot and be running again. Another option is to turn off the idle switch ports separately, when needed. This is the essence of approaches proposed by for instance, Energy Efficient Ethernet, which are shown to provide some energy savings. Still the device chassis is the main energy consuming component; [115] states that switches consume up to 90% of their maximum power as soon as they are turned on, before any incoming load. Therefore, we place our focus on putting the idle network devices into the sleeping mode. Switches in the sleeping mode still consume energy but much less than when they do in the active mode. The energy consumption in the sleeping mode highly depends on the adopted technologies. Besides, their transition time is significantly less than the booting duration. Ultimately, we aim to minimize the number of active switches for a given number of flows in the network.

  Equation 4.1 defines the algorithm objective, with total power consumption PC being the sum of the power consumption of each active switch, m the number of active switches selected to serve n number of flows.

$$Minimize(PC = \sum_{j=1}^{m} PC_j) \tag{4.1}$$

- **Minimize the number of transitions from the sleeping mode to the active mode:**

  Another energy-aware approach that we apply is prioritizing the already existing active switches over the sleeping ones. This help us save the transition time. Consequently, the duration of the scheduling phase will be shorter although the achieved bandwidths for the flows might be reduced due to link sharing. Therefore, it is up to our optimization algorithm to keep the balance between the bandwidths and the power consumption by turning on the switches. We associate this concept with a new variable called *transition_degree*, which indicates the number of switches that need to be turned on for the incoming load. The *transition_degree* can have a value between 0 and the total number of switches. The objective expressed in Equation 4.2 is to minimize this variable.

$$Minimize(transition\_degree) \tag{4.2}$$

- **Maximize the bandwidth for the flows**

  From the user perspective, the main quality requirement might be performance. In real life scenarios, different software systems might have different performance requirements, so they should get different priorities. In this work, we assume that all the software systems, hence the flows we schedule, have the same priority, and we try to optimize the bandwidth for the flows with fairness.

$$Maximize(\sum_{i=1}^{n} BW_i) \tag{4.3}$$

  Equation 4.3 defines the objective, with n being the number of flows in the data center network.

4.3.2 Algorithm Variations

Software systems can have different quality requirements; some are power-sensitive, while some are performance-sensitive. This leads us to define different variations of our scheduling algorithm accordingly. For each variation, different objectives are bound together to formulate an objective variable. However, it is important

to note that all the variations are power efficient as the idle switches are put into the sleeping mode.

We identified four distinct objective variables:

LP-v1: ***Full version***

As Equation 4.4 shows, all the objectives are taken into account, where n is the number of flows and m is the number of active switches. In order to avoid zero denominator in the fraction, 1 is added to *transition_degree*.

$$\frac{\sum_{i=1}^{n} BW_i}{(transition\_degree + 1) * \sum_{j=1}^{m} PC_j} \tag{4.4}$$

LP-v2: ***Without priority***

In this version, all the switches either in the active mode or in the sleeping mode have the same chance of being selected for the next coming flow. We do not give priority to the already active switches as Equation 4.5 displays:

$$\frac{\sum_{i=1}^{n} BW_i}{\sum_{j=1}^{m} PC_j} \tag{4.5}$$

LP-v3: ***Only throughput guaranteed***

For delay-sensitive applications, degrading throughput for the purpose of energy efficiency is not acceptable. Therefore, only performance improvements are taken into account in the objective variables to make sure that it does not increase the number of idle devices with the price of decreasing performance. This version still achieves some power efficiency by putting the idle switches into sleeping mode if there are any. Equation 4.6 shows that in this version we only consider bandwidth values.

$$\sum_{i=1}^{n} BW_i \tag{4.6}$$

LP-v4: ***Only energy efficient***

Our last variation only takes the power efficiency metric into account and minimizes the total power consumption of the data center network, when serving a specific number of flows. To be more efficient, we also include *transition_degree*, which is shown in Equation 4.7.

$$(transition\_degree + 1) * \sum_{j=1}^{m} PC_j \tag{4.7}$$

4.3.3 Algorithm Implementation

In order to model our linear programming problem we define *combination*. A *combination* consists of a set of selected paths for the requested flows. We keep a list of *combinations*, which differ in the selected paths and consequently in the achieved bandwidths and the total power consumption. As new flows requests arrive, the combinations will grow both in size and in number. Unlike the growth in size, which is adding only one flow and one selected path, the growth in the number of combinations is more accelerated. Equation 4.8 calculates the total number of combinations, where n is the number of requested flows. For scalability reasons that will be discussed later in section 4.3.4, we reduce the number of combinations to only three.

$$\text{Number of combinations} = \prod_{i=1}^{n} \text{number of possible paths}_i \tag{4.8}$$

Each *combination* could be a candidate for providing the paths for requested flows. The pseudo-code in Algorithm 1 describes the steps performed in our scheduling process for each of the algorithm variations. *Combination* selection is done by comparing them based on their objective variable. We calculate the necessary metrics (the maximum power consumption, the maximum bandwidths or *transition_degree*) for each combination. For l number of combinations in Equation 4.9, we model our linear programming objective function with a list of boolean variables (x), which shows if a *combination* is selected, multiplied by the objective variables. In case of variations LP-v1, LP-v2 and LP-v3, the objective function is to maximize the objective variables, whereas it minimizes the objective variable in case of variation LP-v4.

$$\sum_{k=1}^{l} x_k * (\text{Combination objective variable}_k) \tag{4.9}$$

We define the constraints ($\#(\sum_{k=1}^{l} x_k = 3)$ and $\#(\sum_{k=1}^{l} x_k = 1)$) to ensure that the top 3 combinations are selected for *CombinationList* and only 1 combination is selected as *SelectedCombination*.

4.3.4 Scalability Analysis

The load on our algorithm can be scaled up by increasing the size of the network and the number of flow requests. We will store only the top three combinations in *CombinationList* regardless of the network size and the load growth. In this way the complexity of the algorithm in terms of number of combinations will

Algorithm 1 Scalable Linear Programming Flow Scheduler

```
while TRUE do
    if newRequestedFlow not Null then
        possiblePaths ← List of possible paths in the data center network
        if CombinationList not ∅ then
            CombinationList ← Update each combination with new possible paths
        else
            CombinationList ← Create a combination with each new possible path
        end if
        for all combination ∈ CombinationList do
            Update the objective variable:
            ←if necessary estimate PC
            ←if necessary estimate BW
            ←if necessary calculate transition_degree
        end for
        CombinationList ← Keep only top 3 combinations (Output of linear programming
formulation)
        SelectedCombination ← Top 1 combination (Output of linear programming formula-
tion)
        Remove/Modify/Add flows on the switches
    end if
end while
```

be independent from the number of flow requests. However, the size of each combination will grow as it includes the path set for all the flows.

The algorithm has two phases. First, combinations will be added based on the number of possible paths. There will be (3 * number of possible paths for the new flow) new combinations, where later the top three will be selected and stored. Second, each new combination needs to be updated in terms of its performance metrics. The execution time for the first phase of the algorithm is $O(n)$, where n denotes the number of possible paths for each step. Since n is usually a small number (especially in the FatTree networks) in data center networks and does not grow rapidly along with the network size, it is reasonable to run the algorithm upon each flow request. The second phase of the algorithm updates the performance metrics of a subset of the active switches that will carry the incoming load and it executes in $O(1)$. For each new possible path, active switches can be updated/modified in constant time, which makes our algorithm very scalable with the number of flow requests and the size of the network.

4.4 Design of Evaluation Decision Framework

We design an evaluation decision framework to deploy the different variations of the scheduling algorithm. The decision framework collects the statistical information from the network and schedules new flows or reschedules the existing flows based on online decisions by the LP algorithm. Figure 4.1 presents the architecture of the decision framework consisting of three main modules: 1) Scheduler,

2) Controller, and 3) Monitor.

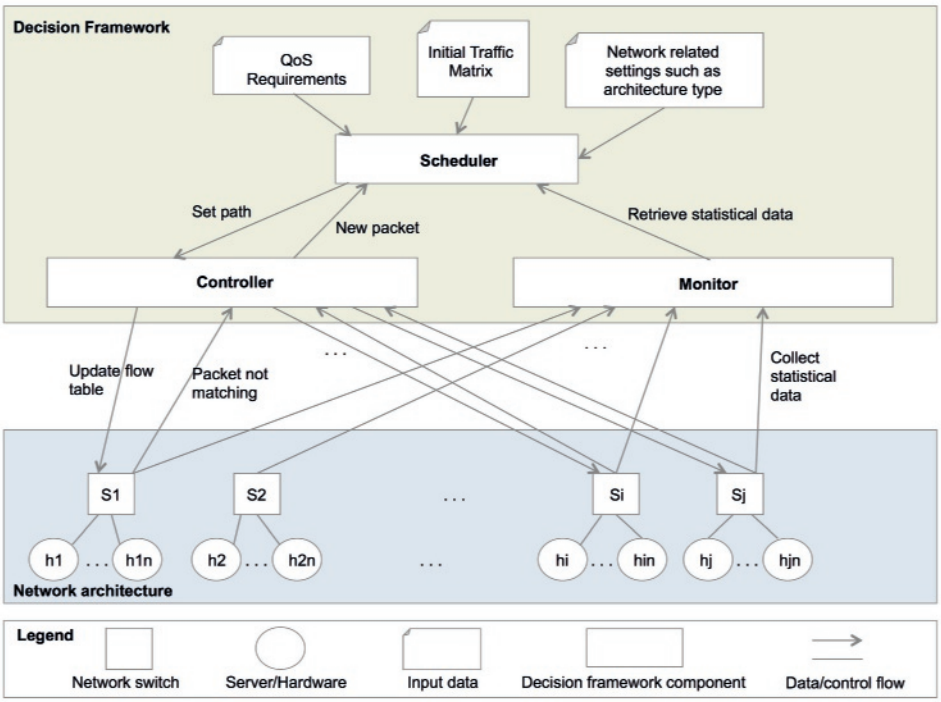


Figure 4.1: Our evaluation decision framework consisting of the scheduler, the controller and the monitor components

4.4.1 Scheduler

The scheduler is at the center of our framework. All the information from other modules and external data sources are inputs to this module. The scheduler deploys the flow scheduling algorithm and it relies on two types of inputs: offline and online. Monitoring data on power consumption and bandwidth of switches are online inputs. Offline input is given to the module at the initial time and will not change during the runtime. The offline inputs by the scheduler are:

- *QoS requirements*: An application will provide the minimum quality requirements to perform as expected. For example an application might define boundaries for bandwidth and latencies, which will be added as constraints to our linear programming approaches.

- *Initial traffic matrix*: It is possible to provide the scheduler with the traffic setup at the initial time. In this case all flows are fully scheduled at the beginning. We did not use this feature in our experiments.
- *Network architecture*: The scheduler is informed at the beginning which network topology is deployed in the data center. Our scheduler can be used to support a number of known network architectures. In our experiments, we use the FatTree topology.

4.4.2 Controller

The controller component receives the requests for setting new flows from the underlying devices. Any time a new packet arrives to a switch, for which an action is not known (the "Packet not matching" function), the packet is sent to the controller component. The controller passes on the requests to the scheduler, which makes decisions regarding the flow paths. The paths will be communicated through the "Set path" function to the controller component. The controller, which performs as a middleware between the network switches and the scheduler, applies modifications to the flow tables of the switches (the "Update flow table" function).

4.4.3 Monitor

This module collects periodically statistical information from the network (the "Collect statistical data" function) and provides the scheduler with the current status of the flows and utilization of the network devices (the "Retrieve statistical data" function). Requesting stats data from the switches on a timely basis adds overhead in terms of performance degradation and delay increase. It is important to select the right sample rate for switches to report on their flow stats in such a way that no considerable information will be missing and overhead is still tolerable. Reports provided by the monitor are further used by the scheduler component. Other than performance-related data, power-related information can also be collected by the monitor. The information provided by this module are the input for power consumption models. If realtime or actual monitors are not present, it is possible to estimate the power consumption of the switches based on the existing power models from the literature. We used power estimations in our simulation experiments.

Realtime inputs are produced by the network devices and are passed to the scheduler through the controller and the monitor components during runtime.

4.5 Simulation Scenarios

We implemented a number of experiments in a simulation environment using our decision framework. To simulate the "Network Architecture" in Figure 4.1 we use Mininet[1], which provides a network testbed to develop software-defined networks. For the controller component we use the open-source POX control software[2], which can update the flow tables of the simulated Open vSwitches[3]. We implement a combination of the Gurobi Python optimizer[4] and POX as our optimization solver in the scheduler component.

As discussed in chapter 3, FatTree is the most widely deployed network architecture in data centers and it is designed to address the "single point of failure" problem. FatTree places the switches in a three-level hierarchy, namely core, aggregation and edge switches. Edge switches are also known as top of rack switches (ToR). Aggregation and edge switches form the so-called PODs. We adopted the FatTree topology in our simulations and we use PODs to simulate distant traffic scenarios. We define network architectures of variable sizes namely, 20, 45 and 80 switches all connected with links of 1Gbps, summarized in Table 4.1.

Table 4.1: The three simulation configurations of the FatTree network architecture used in our simulation

Switch ports	***Switches***	***Hosts***	***PODs***	***Maximum Flows***
4	20	16	4	8
6	45	54	6	27
8	80	128	8	64

4.5.1 Efficiency Metrics

Efficiency metrics help quantify the qualities achieved with each variation of our algorithms. We focus on two efficiency metrics:

- *Power consumption*: Since we deploy our experiments in a simulation environment, we estimate the power consumption of the data center network. We adopted the following model in our simulations:

[1]http://mininet.org
[2]http://www.noxrepo.org/pox/about-pox/
[3]https://www.openvswitch.org/
[4]http://www.gurobi.com/

$$Power_{switch} = Power_{chassis} + num_{linecards} * Power_{linecard} + \sum_{i=0}^{configs} (Power_{configsi} * \sum_{j=0}^{numports} utilizationFactor_j) \quad (4.10)$$

This is based on the model proposed by Mahadevan *et al.*[170]. In Equation 4.10, $Power_{linecards}$ represents the power consumption of the linecard and $num_{linecards}$ is the number of plugged-in cards. $Power_{configsi}$ is the power consumption of a port with the specified link rate i and $utilizationFactor_j$ is the utilization of port j of the switch.

In our experiment, we assume that each flow sends data with the highest possible bandwidth. Therefore, it is always possible to calculate the utilization rates of the switch ports.

- *Time to complete*: We define *time to complete (TTC)* as the time it takes for a software system running in a VM to send a predefined number of bytes to another VM in the network. TTC is a representative QoS requirement because it is aligned with response time and performance.

4.5.2 Traffic Patterns

Depending on the use of the data center, the incoming traffic can follow certain patterns, as identified by [37], which distinguish enterprise and university data centers from cloud data centers based on their running applications. In this work, we generate the traffic in the data center network based on *One-to-One*, modeled by having all the hosts in pairs [137, 257].

As described in [241], the network traffic is categorized based on the length of the associated paths (the number of switches): 1) Far, 2) Middle, and 3) Near. The latter two involve only one POD respectively with 3 switches and 1 switch. We implement the *Far* traffic study case, where nodes from different PODs transfer data to each other. Since the Far traffic involves more number of switches and less switches can be turned to the sleeping mode, this provides a more interesting scenario to validate our algorithms. However, we can utilize the network links fully and investigate more options for the traffic consolidation. For example, in Figure 4.2, $h1$ connects to a randomly selected host from POD2, POD3 or POD4 and the procedure is continued until all the hosts have been paired.

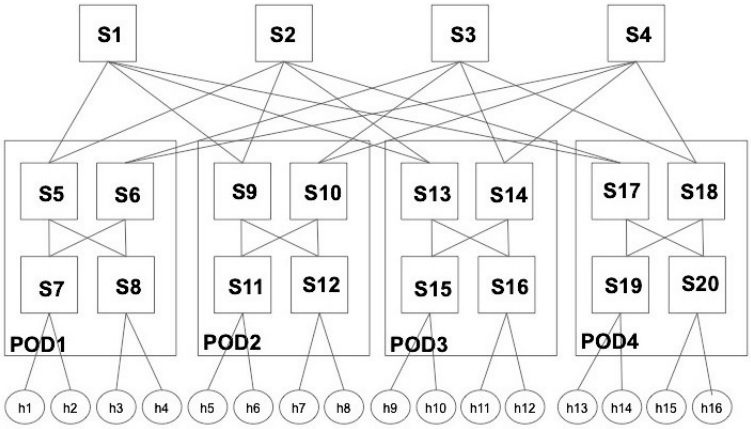


Figure 4.2: A FatTree network architecture with 20 switches (S1-S20) and 16 hosts (h1-h16)

4.6 Results

We ran our simulations in the three *FatTree* network architectures identified in Table 4.1, namely with 20, 45 and 80 switches.

In order to evaluate our algorithm variations in terms of power savings and TTC degradation, we first defined the minimum and maximum values of the efficiency metrics. We select the shortest-path (SP) algorithm and an extension of it as our baseline algorithms. One drawback of SP is that it always selects the first possible shortest-path for a new flow request and it might schedule many flows on the same link degrading the overall performance. Therefore, we define an extension of SP (Smart SP) that guarantees the highest throughput (a non-energy efficient version of our LP-v3). Smart SP does not put idle devices into the sleeping mode, hence it is not energy efficient, but it maximizes the bandwidth achievements of all the flows and does not select only the first possible path.

We compared SP and Smart SP algorithms in the three network sizes. Table 4.2 summarizes the power consumption of data center network of variable sizes when running each of the two algorithms with 100% utilization rate in the one-to-one traffic scenario. As expected, the power consumption of SP and Smart SP is in the same range because they do not put the idle switches into the sleeping mode and effectively the power consumption is given by the number of active switches. The little variations we observe between the two algorithms are due to different utilization factors that appear in Equation 4.10, given the fact that the chosen paths will not be always the same.

Figure 4.3 shows the TTC metric for the two candidate baseline scheduling algorithms in the three simulated network topologies with 100% utilization rate

Table 4.2: Total power consumption of SP and Smart SP in the three simulated network topologies (20,45 and 80 switches)

Baseline candidates	***Total power consumption (20 switches)***	***Total power consumption (45 switches)***	***Total power consumption (80 switches)***
SP	3032W	6856W	12114W
Smart SP	3039W	6847W	12198W

of the one-to-one traffic scenario. In all the 20, the 45 and the 80 switches topologies, the TTC of the Smart SP algorithm is lower than the one of the SP algorithm. In fact Smart SP, which makes intelligent decisions regarding bandwidth achievements, effectively ends up providing higher bandwidths to the requesting flows.

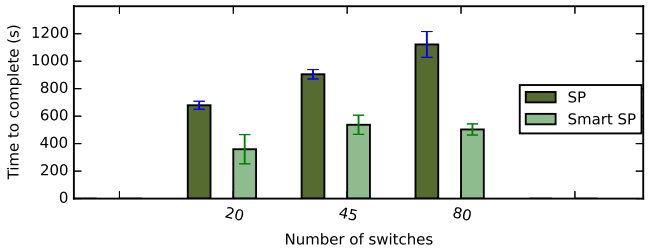


Figure 4.3: Time to complete for SP and Smart SP scheduling algorithms in the three simulated network topologies (38GB of data)

Given their equivalent power consumption and the better TTC of Smart SP, we adopt this algorithm rather than SP as baseline algorithm. We expect that the power consumption of our four variations of LP will all improve on Smart SP; at the same time we are interested in quantifying what is the degradation in the TTC in each of the four variations.

It must be noted that power consumption will increase if the load goes up. This increase could be small when due to higher link utilization rates or large when turning on the switches from the sleeping mode. To assess the effectiveness of our algorithms we measure power savings rather than only power consumption values. Power saving is the amount that each of scheduling algorithms will save if there is some room for improvement.

The achievable power saving in the sleeping mode will heavily influence the total power saving in the data center. To quantify this, we first examined the power consumption of the four scheduling algorithms under different power savings percentages in sleeping mode. Our switches could save between 20%, 40%, 60% and

80% when in this state, where the 40% and the 60% are the most realistic values. In the rest of this chapter we will use 60% as the power saving of the devices in the sleeping mode [3]. Table 4.3 shows the maximum power saving of each algorithm compared to the Smart SP in the network of size 45 with 27 flows. In all cases, the maximum power saving is achieved by LP-v4, followed by LP-v1, LP-v2 and LP-v3. There is a nearly linear relation between the maximum achievable power saving and the maximum sleeping mode power saving. 20% improvement of power saving in sleeping mode results in 10% improvement for power sensitive algorithms (LP-v1 and LP-v4) and 5% improvement for performance sensitive algorithms (LP-v2 and LP-v3).

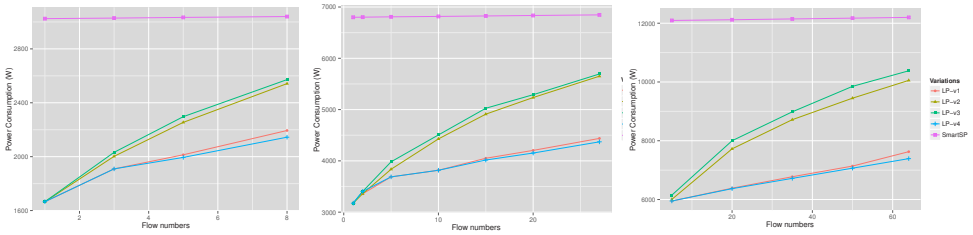
Table 4.3: Maximum power savings as function of sleeping mode power savings (45 switches with 27 flows)

Algorithm variations	***Maximum sleeping mode power savings:***			
	20%	**40%**	**60%**	**80%**
LP-v1	12%	23%	35%	46%
LP-v2	6%	12%	17%	23%
LP-v3	5%	11%	16%	22%
LP-v4	13%	24%	36%	48%

To assess the scalability quality requirement of the algorithms we investigated the power consumption of the four algorithm variations when increasing the flow numbers. Fig. 4.4 shows the results for a network of size 20, 45 and 80 respectively. In the network of size 20 we measure the power consumed in presence of 1, 3, 5 and 8 flows; we used 2, 5, 10, 15, 20 and 27 flows in the network of size 45 and 5, 20, 35, 50 and 64 flows in the network of size 80. In all cases LP-v4 and Smart SP present the lowest and highest values, and as such they identify the lower and highest bound for the power consumption.

The three subfigures show that LP-v1 selects paths such that the total power consumption remains close to the optimum energy efficient variation (LP-v4). Gradually, when the number of flows increases, LP-v1 and LP-v4 diverge as LP-v1 needs to take into account the throughput requirements too. LP-v2 and LP-v3 show almost identical patterns when increasing the number of flows. Similar to square-root functions, they change rapidly in terms of power consumption for the small number of flows particularly in case of 45 and 80 switches. Their power consumption shows less acceleration for larger number of flows because there are no more switches to turn on from the sleeping mode.

When the number of flows is small all the algorithm variations show the same power consumption; in this case it will not be possible to provide power savings from shaping the network traffic and increasing the number of idle devices.



(a) FatTree network with 20 switches **(b)** FatTree network with 45 switches **(c)** FatTree network with 80 switches

Figure 4.4: Power consumption of the data center network for different algorithm variations with variable number of flows based on the one-to-one traffic scenario for a fixed network size

However, small number of flows will involve small number of switches and power savings can be achieved by putting the existing idle devices into the sleeping mode.

Figure 4.5 shows the TTC measured from the applications running in the hosts as a function of the four possible scheduling algorithms when running with maximum number of flows. We configure running software to send 38GB of data from the sender VM to the receiver VM. Algorithm variations that perform better in power consumption exhibit higher TTC measurements, e.g. LP-v4 that focuses purely on the energy efficiency will produce a much larger *time_to_complete*, 71% for the network of 20 switches and 502% for the network of 80 switches. LP-v1 also shows a considerable increase in TTC when the network size grows. Differently, LP-v2 and LP-v3 appear to be more performance-focused and stable for different sizes of the network.

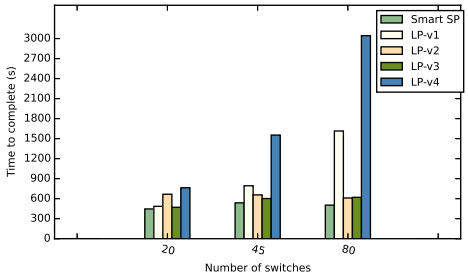


Figure 4.5: Time to complete for the four different scheduling algorithm in the three simulated network topologies (38GB of data)

If performance is the decisive factor LPv2 and LPv3 will be the likely choice. In this case it is interesting to quantify the degradation in power savings. Figure

4.6 shows the degradation in power saving compared to the baseline values (LP-v4 and Smart SP) when using different variations of the scheduling algorithm, compared to the optimal power saving achieved with LPv4. Degradation percentage is calculated from the following, given that x is the power consumption of the algorithm variation:

$$\text{Degradation Percentage}_x = \left(\frac{PC_x - PC_{\text{LP-v4}}}{PC_{\text{Smart SP}} - PC_{\text{LP-v4}}}\right) * 100 \tag{4.11}$$

In all three topologies we observed that LP-v1 is the most power efficient variation that shows around 95% improvement, which means only 5% degradation from the optimum power saving. LP-v2 achieving 50% of the maximum power saving outperforms LP-v3 achieving 45% of the maximum power saving, which provides more bandwidth for the running applications. It is interesting to see that all the variations remain with the same range of power savings for different network sizes.

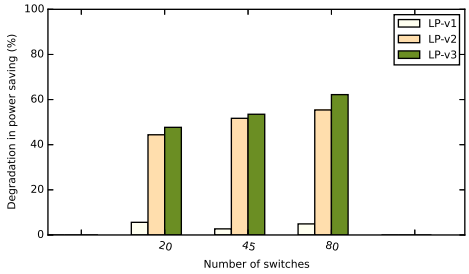


Figure 4.6: Degradation in power savings of three scheduling algorithms namely, LP-v1, LP-v2 and LP-v3 in the three simulated network topologies. LP-v4 and Smart SP are considered as the baselines for calculations of power savings.

4.7 Discussion

Software systems running in data centers have different quality requirements. Our results can be used as inputs for them to decide which algorithm to implement for scheduling their corresponding network traffic in software-defined networks. As shown in section 4.6, our four algorithm variations result in different qualities as a function of the size of the network, the number of flows to be scheduled, and the expected power saving of the switches when in their sleeping mode. LP-v2 and LP-v3 provide higher bandwidth and smaller time to complete, while still saving power compared to the shortest-path algorithm. This makes them a great choice for delay-sensitive software systems. LP-v1 and LP-v4 focus on the energy

efficiency quality requirement rather than the time to complete and as such they can be adopted for delay-insensitive software systems. In a data center we will often see a combination of delay-sensitive and delay-insensitive software systems. In this case, the scheduler component in our framework can be easily extended to deploy multiple variations of linear programming algorithms concurrently.

Our findings show a nearly linear relation between the total power savings in data center network and the power saving of switches in sleeping mode. It is important to assess the power saving of the switches in their sleeping mode beforehand, which helps the scheduling algorithms adjust their decisions. Switches with lower sleeping mode power savings are prioritized to carry the incoming load either by performance sensitive algorithms (LP-v2 and LP-v3) or power sensitive algorithms (LP-v1 and LP-v4).

4.8 Conclusion

In this chapter, we address RQ2.2, namely "How can optimization algorithms utilize infrastructure in an energy efficient way?". We presented different variations of linear programming scheduling algorithms that can optimize the power savings in software-defined networks. Each variation has a different objective in terms of power optimization and performance-related quality requirements. Each objective function implements a combination of the following approaches: "put the idle devices into sleeping mode", "increase the number of idle devices", "prioritize the existing active switches over the sleeping ones". The programmability of software-defined networks empowers software systems to self-adapt when runtime changes occur; they can choose the best network flow path and reconfigure the flow tables of the network switches. Our algorithm variations are designed for data center networks, in which realtime and scalable traffic scheduling is crucial. We ensure scalability by keeping the top 3 flow set candidates, which saves scheduling time as new flows come in. We evaluated our algorithm variations in a simulated FatTree network architecture using a one-to-one traffic scenario. We also quantified the relation between power saving of each algorithm and sleeping mode power saving. Our results show that two of the variations (LP-v2 and LP-v3) remain stable in terms of power saving and the time to complete metrics, as the network size grows. Two other (LP-v1 and LP-v4) provide the highest power savings in the network and they are suitable for delay-insensitive software systems.

5

Case Study 2: Empirical Evaluation of Cyber-Foraging Architectural Tactics

This chapter is based on empirical evaluation of self-adaptation architectural tactics in the context of cyber-foraging to answer our RQ3.1. We particularly focus on surrogate provisioning tactics, as they engage the underlying networking infrastructure. We used the Green Lab of Vrije Universiteit Amsterdam to setup and carry out our experimentation. We develop a synthetic mobile application as our subject, which is equipped with two types of surrogate provisioning tactics: Static and Dynamic. Our results show a significantly higher resilience for Static surrogate provisioning than Dynamic surrogate provisioning. Also both architectural tactics improve energy efficiency compared to non-cyber-foraging architectures (our baseline measurements). However, none of the two tactics outperforms the other with respect to energy efficiency, which means that the overhead of runtime optimization remains similar.

5.1 Introduction

In 2014, the number of mobile users exceeded the number of desktop users globally, which was about 1.7 billion users [43]. Consequently, many computation tasks are migrated to handheld devices as mobile apps. Statistics provided by "The Statistics Portal" forecast approximately 269 billion mobile app downloads for 2017, which is around 20% more than the previous year [2]. Although handheld devices are often selected as the main target for consumers and app developers, they are still limited in resources in terms of computational power and battery life.

The importance of extended device battery life has motivated software architects to introduce *Mobile Cloud Computing* solutions, in which the cloud takes charge of compute- and data-intensive tasks. Although these solutions signifi-

cantly help to address resource limitations, a number of prerequisites need to be met. For example, a reliable Internet connection must exist between the handheld device and the cloud, which is not necessarily guaranteed in resource-scarce environments. Resource-scarce environments usually lack stable environmental conditions. *Cyber-foraging* has been introduced to enable resource-limited devices to benefit from available external resources in such environments with dynamic conditions.

A number of cyber-foraging tactics have been identified and categorized in [156, 160] to help software architects select the best tactics to meet system requirements. In this study, we particularly focus on the "Surrogate Provisioning" tactics from an experimentation point of view. We study to what extent the cyber-foraging architectural tactics for surrogate provisioning impact system resilience and energy efficiency. Our findings guide software architects and software engineers to trace the impact of their design decisions with scientific insights concluded from quantifiable metrics. Our main contributions are:

- we provide a detailed description of cyber-foraging tactics for surrogate provisioning;
- we present a runtime optimization algorithm to support surrogate provisioning tactics and describe a proof-of-concept implementation;
- we show the systematic design and execution of our experimentation approach applied to surrogate provisioning, which can be reused for validating other cyber-foraging architectural tactics;
- we report on the execution and the results of our empirical experimentation aimed at quantifying the impact of the cyber-foraging tactics for surrogate provisioning on resilience and energy efficiency in a controlled environment;
- we provide an evaluation of the cyber-foraging tactics for surrogate provisioning, emphasizing trade-offs with respect to different system qualities.

This chapter is organized as follows: Section 5.2 presents an overview of the cyber-foraging architectural tactics. Section 5.3 focuses on the surrogate provisioning tactics and how online optimization algorithms play a role in the system. In Section 5.4 we describe the scope of the experimentation using the goal, research questions, and metrics. Section 5.5 provides details of the planning steps from different perspectives such as context selection, variable selection, hypothesis formulation, subject selection, experiment design, and instrumentation. The steps taken to execute the experiments are explained in Section 5.6. In Sections 5.7 and 5.8 we present and discuss our results. Section 5.9 discusses the implications of our findings for software architecture. In Section 5.10 we describe

the possible threats to validity and their mitigation. Section 5.11 discusses related work. Finally, Section 5.12 concludes the chapter and outlines the research direction for our future work.

5.2 Background

Cyber-foraging is a mechanism that leverages cloud servers, or local servers called surrogates, to augment the computation and storage capabilities of resource-limited mobile devices while extending their battery life [215]. There are two main forms of cyber-foraging [92, 154, 227]. One is computation offload, which is the offload of expensive computation in order to extend battery life and increase computational power. The second is data staging to improve data transfers between mobile devices and the cloud by temporarily staging data in transit on intermediate, proximate nodes. While cyber-foraging can take place between mobile devices and cloud resources, our focus is on systems that use intermediate, proximate surrogates.

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both [33]. Software architectures are created because a system's qualities, expressed as functional and non-functional requirements, can be analyzed and predicted by studying its architecture.

One of the main challenges of building cyber-foraging systems is the dynamic nature of the environments that they operate in. For example, the connection to an external resource may not be available when needed or may become unavailable during a computation offload or data staging operation. As another example, multiple external resources may be available for a cyber-foraging system but not all have the required capabilities. Adding capabilities to deal with the dynamicity of the environment has to be balanced against resource consumption on the mobile device so as to not defeat the benefits of cyber-foraging. Being able to reason about the behavior of a cyber-foraging system in light of this uncertainty is key to meeting all its desired qualities, which is why software architectures are especially important for cyber-foraging systems.

Given the potential complexity of cyber-foraging systems, it would be of great value for software architects to have a set of reusable software architectures and design decisions that can guide the development of these types of systems, the rationale behind these decisions, and the external context/environment in which they were made; this is called *architectural knowledge* [145, 150]. One way to capture architectural knowledge is in the form of *software architecture strategies*.

We define a *software architecture strategy* as the set of architectural design decisions that are made in a particular external context/environment to achieve particular system qualities. Software architecture strategies are codified as archi-

tectural tactics that can be reused in the development of software systems. We define *architectural tactics* as design decisions that influence the achievement of a system quality (i.e., quality attribute) [33].

Software architecture strategies for cyber-foraging systems are therefore the set of architectural design decisions, codified as reusable tactics, that can be used in the development of cyber-foraging systems to achieve particular system qualities such as resource optimization, fault tolerance, scalability and security, while conserving resources on the mobile device [157].

In a previous work we conducted a systematic literature review (SLR) on architectures for cyber-foraging systems [154, 156]. The common design decisions present in the cyber-foraging systems identified in the SLR were codified into functional and non-functional architectural tactics [154, 158]. Functional tactics are broad and basic in nature and correspond to the architectural elements that are necessary to meet cyber-foraging functional requirements. Non-functional tactics are more specific and correspond to architecture decisions made to promote certain quality attributes. Non-functional tactics have to be used in conjunction with functional tactics.

A cyber-foraging system must have at a minimum the following combination of functional tactics:

- Computation Offload and/or Data Staging tactics to provide cyber-foraging functionality
- A Surrogate Provisioning tactic to provision a surrogate with the offloaded computation or data staging capabilities
- A Surrogate Discovery tactic so that the mobile device can locate a surrogate at runtime

Then, based on additional functional and non-functional requirements, such as fault tolerance, resource optimization, scalability/elasticity, and security, complementary tactics are selected.

The work in this chapter focuses on surrogate provisioning tactics. We compare the different surrogate provisioning tactics from an architectural point of view with respect to their resilience and energy efficiency.

5.3 Surrogate provisioning tactics

5.3.1 Tactics description

To be able to use a surrogate for cyber-foraging, it has to be provisioned with the offloaded computation and/or the computational elements that implement

the offloaded computation or enable data staging. There are two main types of tactics for surrogate provisioning [154]:

- *Static Surrogate Provisioning*: Surrogates are pre-provisioned with the capabilities that are requested by mobile clients. Figure 5.1 shows the sequence diagram of how a pre-provisioned surrogate interacts with the mobile device. The mobile app decides whether to request remote execution or execute the computation locally. To make that decision it first collects data on the network connection and the surrogate status through a *monitoring service*. A *runtime optimization algorithm* outlines the optimum offloading plan based on the input data. If the plan is not to offload, the computation will be executed locally in the mobile device. If the plan is to offload, a *provisioning request service* is called that starts a JVM in the surrogate and notifies the mobile app with the resulting status. The mobile app waits for a specific period and then requests the results of the computation through a *provisioning result service*.

- *Dynamic Surrogate Provisioning*: Surrogates are provisioned at runtime with the computation capabilities. Surrogates can receive the offloaded computation from either the mobile device or the cloud. Figure 5.2 shows the steps that take place in dynamic surrogate provisioning. Similar to static surrogate provisioning, the mobile app must first collect data through a *monitoring service* but the monitoring is more complex as the status data of the cloud repository is also required. A runtime optimization algorithm can suggest either different provisioning sources (the cloud or the mobile device) or local execution. If the plan is to offload, the mobile app calls a *provisioning request service*. In the case of provisioning from the mobile device, the mobile device *sends the computation capability* to the surrogate itself while in provisioning from the cloud, the mobile device only informs the surrogate with the location of the offloaded computation in the form of a URL. Therefore, the surrogate will be able to download the computation from a cloud repository and install the computation inside an execution container (JVM as shown in the figure). Again, the mobile app can retrieve the results by calling a *provisioning result service*.

In our previous study, we propose a decision model to select the best fitted architectural tactics according to functional and non-functional requirements in cyber-foraging [160]. Figure 5.3 shows the decision model specified for surrogate provisioning tactics. As the figure shows, there are pros and cons for each surrogate provisioning tactic. For instance, Static Surrogate Provisioning *simplifies the deployment process*. Therefore, it is a good match for applications with a small set of computations or data processing operations that can be pre-loaded on the surrogate. Static Surrogate Provisioning performs the best in cyber-foraging

applications, in which multiple surrogates offer the same capabilities. This reduces *flexibility* because surrogates are limited by the pre-installed capabilities. Another disadvantage with Static Surrogate Provisioning is a reduction on *maintainability* because changes to capabilities must be propagated to all surrogates. Differently, Dynamic Surrogate Provisioning offers *greater flexibility* because capabilities are not limited by what is already installed, making it a good match for when there is a large set of capabilities that can execute on a surrogate. Various capabilities that reside on the mobile device can be offloaded to a surrogate at runtime. However, Dynamic Surrogate Provisioning has a negative impact on *provisioning time* compared to Static Surrogate Provisioning because capabilities have to be downloaded first from either a cloud repository or the mobile device. In the case of provisioning from the cloud, the capabilities must exist in a repository in the cloud, and connectivity between the surrogate and the repository is required to download the capabilities, which affects *availability* negatively. This improves *maintainability* because changes to capabilities only need to be propagated to the cloud repository. In contrast, in the case of provisioning from the mobile device, *maintainability* is reduced because changes to offloadable capabilities must be propagated to all mobile devices. Depending on the size of the capability to be transferred, *bandwidth efficiency* could be negatively affected. Consequently, *energy efficiency* is decreased on the mobile device because of the battery power required on the mobile device to send the capability to the designated surrogate.

5.3.2 Runtime optimization algorithm for surrogate provisioning

In this chapter, we propose an algorithm for runtime selection of a task execution environment aided by surrogate provisioning tactics. The objective of our optimization algorithm is to minimize response time, which is the period of time it takes for the mobile device to receive the results, either by offloading or by local execution. To do so, response time (RT) is estimated for different scenarios using equations 5.1 and 5.2, which are adopted from [55]. RT_{offload} is used for both types of surrogate provisioning (Static and Dynamic). However, data size differs for different tactics.

$$RT_{\text{local}} = T_{\text{local}} \tag{5.1}$$

$$RT_{\text{offload}} = T_{\text{surrogate}} + \frac{\text{data size}}{BW_{network}} + T_{delay} * (1 + \frac{\text{data size}}{\text{TCP window size}}) \tag{5.2}$$

T_{local} and $T_{\text{surrogate}}$ show the time it takes for the mobile device and the surrogate to execute the computation task. The local execution time is known in advance. The surrogate execution time is calculated in the algorithm at runtime.

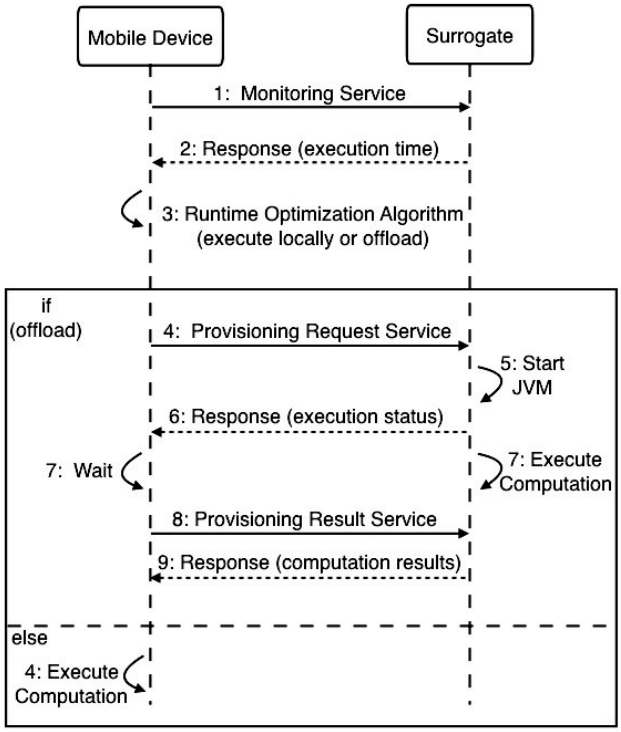


Figure 5.1: The sequence diagram for Static Surrogate Provisioning in case of computation offloading

We calculate the network connection overhead using $BW_{network}$, which is the wireless network bandwidth between the mobile device and the surrogate, and T_{delay}, which shows the network delay. $BW_{network}$ is obtained offline using the *iperf*[1] application and then hard-coded on the mobile device. T_{delay} is measured at runtime using *ping* messages. We use the default value of TCP window size[2] on Android, which is 65,536 bytes (64KB).

The pseudo-code in Algorithms 2 and 3 specifies the steps taken by our algorithm in cases of static and dynamic surrogate provisioning. The algorithm relies on equations 5.1 and 5.2 to calculate the response time values. As shown in the pseudo-code, the optimization algorithm for dynamic surrogate provisioning performs a number of extra steps, in which it calculates the time-overhead to

[1]It is a known tool to measure network performance: `https://iperf.fr/`

[2]The Transmission Control Protocol (TCP) is one of the main standard network protocols that complements the Internet Protocol (IP).

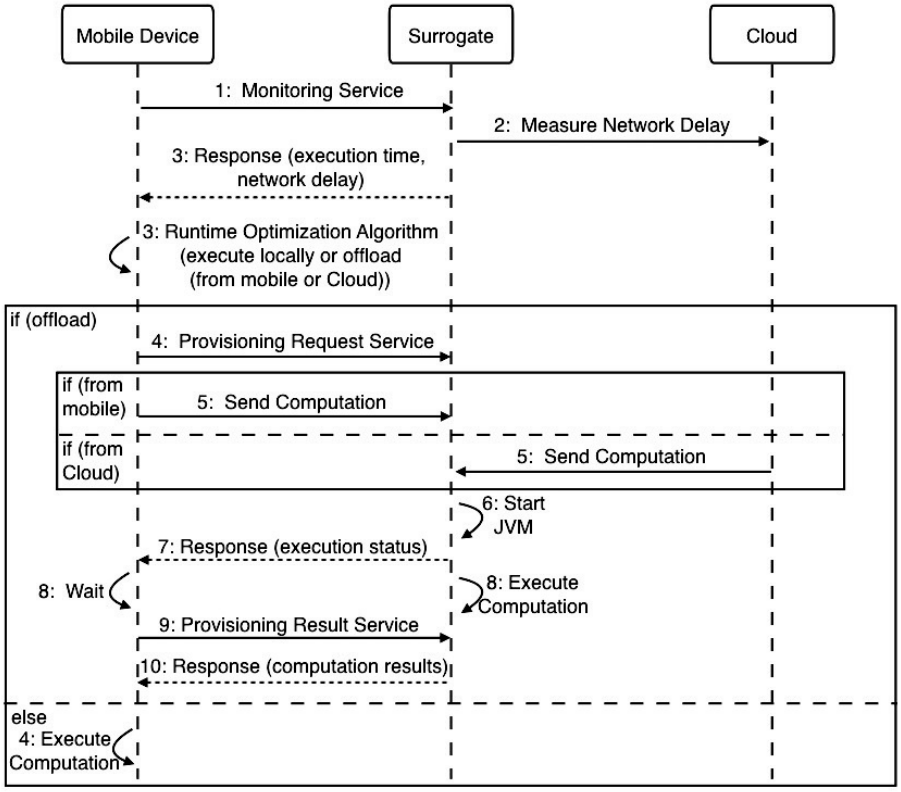


Figure 5.2: The sequence diagram for Dynamic Surrogate Provisioning in case of computation offloading

install the computation on the surrogate.

5.4 Experiment definition

With our experimentation we aim to empirically evaluate the surrogate provisioning tactics, and provide software architects and software engineers with reproducible scientific insights. The systematic design of our experimentation can be adopted by other researchers as a viable, reusable approach. Our experimentation process follows the well-known framework introduced by Basili V.R. *et al.* [31]. It consists of four phases: 1) Definition, 2) Planning, 3) Execution, and 4) Analysis. For the first phase (Definition), we selected the Goal-Question-

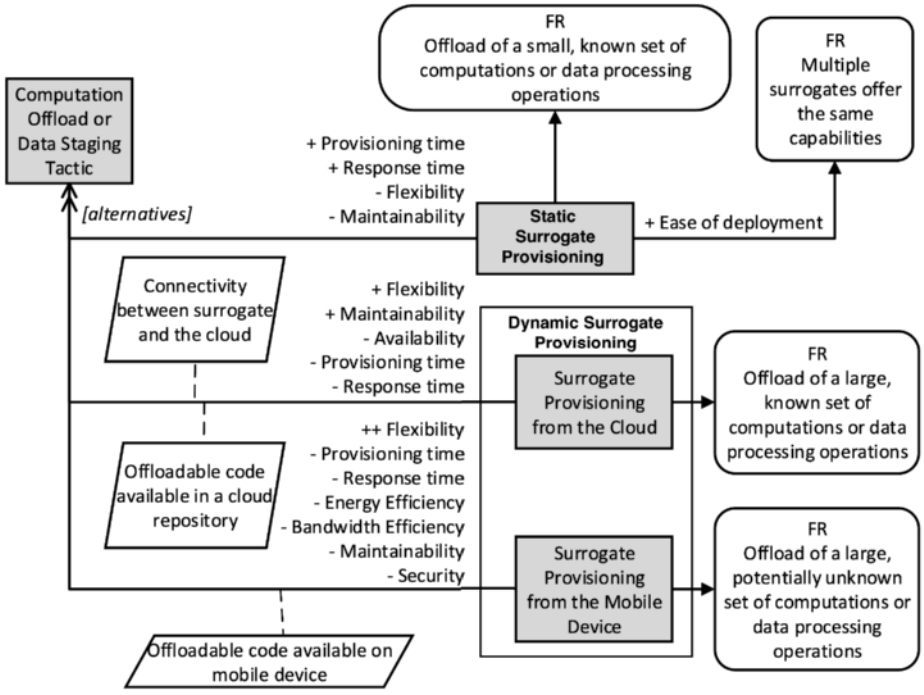


Figure 5.3: The decision model for surrogate provisioning tactics adapted from our previous study [160]

Metric (GQM) paradigm.

The GQM is a top-down conceptual decomposition of the goal into questions and metrics, that provides the traceability of measurement data in the goal achievement [30]. Our goal is formulated as follows:

"Analyze **architectural tactics for surrogate provisioning** for the purpose of **evaluation** with respect to **resilience and energy efficiency** from the viewpoint of **software architects** and **software engineers** in the context of **cyber-foraging applications**"

Our objects are surrogate provisioning tactics. We focus on two quality attributes: resilience and energy efficiency. We evaluate these tactics from the point of view of a software architect or software engineer: this means that our results will be helpful when making design decisions related to these quality attributes and their possible trade-offs. Our results apply to the general field of cyber-foraging applications, although they might provide useful insights for a broader range of software systems.

Algorithm 2 Runtime Optimization Algorithm for Static Surrogate Provisioning

while $TRUE$ **do**
  if *newComputationRequest* not Null **then**
    RT_{local} ←estimate the local execution time (T_{local})
    Calculate $RT_{offload}$ (the output of equation 5.2):
    $T_{surrogate}$ ←estimate the remote execution time
    $BW_{network}$ ←estimate the bandwidth of the connection between the mobile device and the surrogate
    T_{delay} ←measure the network delay of the connection between the mobile device and the surrogate
    if $RT_{local} < RT_{offload}$ **then**
      Execute the computation locally and update RT_{local} based on the execution results
    else
      Start the offloading process and update the $RT_{offload}$ variables based on the execution results
    end if
  end if
end while

We compare the tactics based on a number of predefined metrics. To proceed, we define the following questions, which elaborate our goal in a quantifiable way.

RQ1: What is the difference in terms of resilience between the surrogate provisioning tactics?

Cyber-foraging tactics are known to provide systems with dynamic behavior to account for unavailable resources. However, the extent to which systems can benefit from this dynamicity has not been studied. We answer this question by quantifying the resilience of a system [17]. Before and during the execution of the mobile applications we introduce a change to the system, which requires an online decision. The changes vary from low battery level to bad network connection. Figure 5.4 shows the phases that a resilient system goes through when a change occurs. At first the system is in its "Initial steady state" until the time of the change. With change, a two-phase transition state starts, which includes the "Reaction" and "Adaptation" phases. At the end of the adaptation phase, the system returns to a steady state. The faster that the system passes through the transition phase to the steady state, the higher resilience the system provides [17]. We compare the resilience of the system when using different surrogate provisioning tactics by analyzing the reaction and the adaptation times.

RQ2: What is the difference in terms of energy efficiency between the surrogate provisioning tactics?

Algorithm 3 Runtime Optimization Algorithm for Dynamic Surrogate Provisioning

while $TRUE$ **do**
  if *newComputationRequest* not Null **then**
    RT_{local} ←estimate the local execution time (T_{local})
    Calculate two values of $RT_{offload}$ for two data sources (the output of equation 5.2):
    $T_{surrogate}$ ←estimate the remote execution time
    $BW_{network}$ ←estimate the bandwidth of the connection between the mobile device and the surrogate
    $BW_{network-cloud}$ ←estimate the bandwidth of the connection between the surrogate and the designated cloud server
    T_{delay} ←measure the network delay of the connection between the mobile device and the surrogate
    $T_{delay-cloud}$ ←measure the network delay of the connection between the surrogate and the designated cloud server
    if $Min(RT_{local}, RT_{offload}$ from the mobile, $RT_{offload}$ from the cloud) == RT_{local} **then**
      Execute the computation locally and update RT_{local} based on the execution results
    else
      Start the offloading process and update the $RT_{offload}$ variables based on the execution results
    end if
  end if
end while

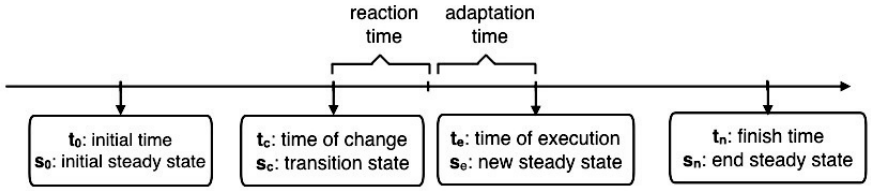


Figure 5.4: The states of a resilient system when introducing a system or environmental change

We analyze the energy efficiency of the surrogate provisioning tactics at runtime. We calculate the energy consumption of the mobile device using the measured average power consumption during the execution of a synthetic application. Basically, the synthetic application consists of a computation task, which can either be performed locally or offloaded to a surrogate.

We use the following metrics to quantitatively answer our questions:

- **Mobile energy consumption**: the number of joules consumed to execute the computation task.
- **Reaction time**: the time the system takes to detect a change and decide on a suitable setup.
- **Adaptation time**: the time the system takes to adapt to the new setup.

The directed GQM graph in Figure 5.5 indicates how our goal is covered by providing quantified answers to the questions using the metrics. *RQ1* is dependent on the metrics "Reaction time" and "Adaptation time" which enable us to quantify resilience of a system. *RQ2* requires the metric "Mobile energy consumption" to investigate how the surrogate provisioning tactics impact the energy consumption of the mobile application.

5.5 Experiment planning

In this section we describe our experimentation in terms of 1) Variable selection, 2) Hypothesis formulation, 3) Subject selection, 4) Experiment design, and 5) Instrumentation.

5.5.1 Variable selection

The metrics identified in the GQM tree (See Figure 5.5) are our *dependent variables*: Mobile energy consumption, Reaction time, and Adaptation time. We use the dependent variables to answer the research questions and draw conclusions. We also consider two additional dependent variables, which help to calculate the mobile energy consumption metric:

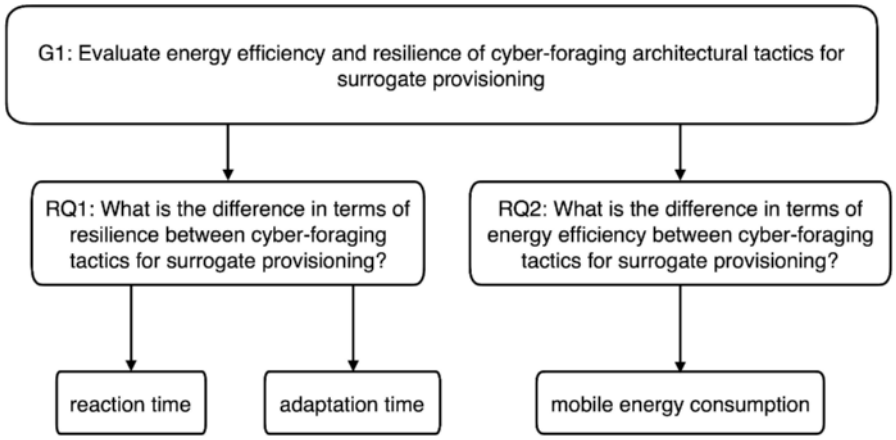


Figure 5.5: The GQM graph summarizing the relation between the goal of our experiment, the research questions, and the metrics

- Average power consumption: The power consumption of the mobile device is measured at runtime in 1 second intervals using a system profiler. We calculate the average value of the power consumption values for each trial.
- Response time: It specifies the duration from the moment that the computation is requested until the results are collected on the mobile device. It includes the time to execute the computation task and the time to offload the task to the surrogate if offloading is decided.

In contrast, the *independent variables* are those that are controllable in the experiment. In our case, the deployment of a surrogate provisioning tactic is the main independent variable that we select as a factor. We define two distinct treatments:

- **Treatment 1: Static surrogate provisioning**: In this treatment, the surrogate is pre-provisioned with the offloaded computation.
- **Treatment 2: Dynamic surrogate provisioning**: In this treatment, the surrogate is provisioned at runtime. Depending on environmental conditions such as the quality of the network connection, the capabilities can be downloaded from the mobile device itself or from a remote host in the cloud.

The runtime optimization algorithm implemented for the treatments decides at runtime whether the computation should be executed locally or remotely. In addition, we performed a number of baseline measurements to provide a reference set of values for our metrics. These measurements were performed on the same mobile device used for our experiment, running the computation task *locally*.

Other independent variables, such as hardware and software configurations are related to the execution environment. These variables have been kept constant in our experimentation (see Section 5.5.5) to avoid confounding factors. Another independent variable we considered is the network connection because in cyber-foraging scenarios these connections might not always be available and reliable. For this reason, in our experimentation we introduced a 10% probability of having a faulty connection, in which case our adaptation algorithm has to select local computation.

5.5.2 Hypotheses formulation

In this section, we formulate the aforementioned research questions into *Null* and *Alternative* hypotheses.

- *RQ1: What is the difference in terms of resilience between surrogate provisioning tactics?*

  As explained earlier, the resilience of the system is dependent on the duration of the transition phase when introducing a change. We observe the difference in resilience of the two cyber-foraging systems based on Eq. 5.3.

  $$\Delta T = T_d - T_s \tag{5.3}$$

  T_s, in seconds, is the sum of the reaction and adaptation time of the system using a static provisioning tactic, and T_d shows the same for a dynamic provisioning tactic.

  The null hypothesis in Eq. 5.4 suggests that both surrogate provisioning tactics provide the same level of resilience. In contrast, the alternate hypothesis in Eq. 5.5 states that the resilience of the cyber-foraging system with dynamic surrogate provisioning is lower (i.e. the transition time is higher) compared to the static surrogate provisioning.

  $$H1_0 : \Delta T \approx 0 \tag{5.4}$$

  $$H1_a : \Delta T > 0 \tag{5.5}$$

- *RQ2: What is the difference in terms of energy efficiency between surrogate provisioning tactics?*

  Eq. 5.6 shows the difference in energy efficiency of the surrogate provisioning tactics. Energy efficiency is measured as the energy consumed (in Joules) when performing a task. EE_d is the energy efficiency of the system with dynamic surrogate provisioning and EE_s is the energy efficiency of the

system with static surrogate provisioning. The null hypothesis in Eq. 5.7 indicates that both surrogate provisioning tactics have the same level of energy efficiency. In contrast, the alternate hypothesis in Eq. 5.8 implies that energy efficiency differs between the surrogate provisioning tactics.

$$\Delta EE = EE_d - EE_s \tag{5.6}$$

$$H2_0 : \Delta EE \approx 0 \tag{5.7}$$

$$H2_a : \Delta EE \neq 0 \tag{5.8}$$

5.5.3 Subject selection

The main objects of our experiments are the architectural tactics for surrogate provisioning. We select a synthetic application as our subject, using convenience sampling. This defines our study as a **quasi-experiment** as our sample is not randomly selected. Therefore, we can not guarantee that our sample is representative of all mobile applications. However, we mitigate this concern by implementing the typical behavior for a real mobile application in resource-scarce environments (i.e., low coverage and hostile environments [159]). We further discuss this concern in Section 5.10.

Our mobile application executes a specific computation-intensive task. It converts input colored images to grayscale. It imitates the real life resource-hungry computations that utilize the available resources in the mobile device notably. Hence, our experimentation still provides valuable information with regards to the cyber-foraging applications.

We implemented our Android mobile application to support three different scenarios, as shown in Figure 5.6. Despite the architectural differences in each scenario, they all perform a specific computation task, known as our subject:

- In the *baseline* scenario, the computation task can only be executed on the mobile device. The mobile app will not have the flexibility to offload the computation task to external devices. No cyber-foraging architectural tactics are implemented in this scenario.

- The *static* treatment implements the cyber-foraging architectural tactic for static surrogate provisioning. This tactic gives more flexibility to the mobile app compared to the baseline scenario, as follows: depending on the battery level of the mobile device and other availability factors, the mobile app might decide to offload the computation task to a nearby surrogate. In this treatment, the surrogate capability is static, i.e., the surrogate is provisioned at design time with the specific computation task. So, if at runtime the mobile app decides to offload the computation, it only needs to invoke the execution on the surrogate.

Figure 5.6: Our mobile application, providing three execution scenarios: No Surrogate Provisioning, Static Surrogate Provisioning, and Dynamic Surrogate Provisioning

- The *dynamic* treatment implements the cyber-foraging architectural tactic for dynamic surrogate provisioning. This tactic provides the highest flexibility to the mobile app in terms of the variation of the offloaded computation. However, the surrogate must be provisioned with the capability at runtime. The dynamic surrogate provisioning can take place from two different resources, the mobile device itself or a remote cloud server. The mobile app must decide if the computation should be offloaded and if so, from which resource the surrogate should be provisioned.

In order to implement the two treatments, we set up a number of web services on the surrogate. Figure 5.7 shows how the services interact with different components of the mobile app for each treatment. In particular, the **Optimizer** of the mobile app, which receives the task execution request from the users, is the component that starts the optimization process. It retrieves monitoring data from the **Monitor** component and accordingly selects the task execution environment, which is either *The mobile device* or *The surrogate*, where the **Computation Task** is executed. For the surrogate the following describes each web service:

- *Monitoring Service*: It collects status data such as computation execution time and remote host network delay, which can be invoked by the mobile app.

- *Provisioning Request Service*: It receives the offload request from the mobile app. In the case of the static treatment, the computation task is already installed on the surrogate and the mobile app will only send the necessary execution parameters to start the computation. Differently, in the dynamic treatment, the mobile app sends the surrogate provisioning parameters along with the execution parameters. In the case of surrogate

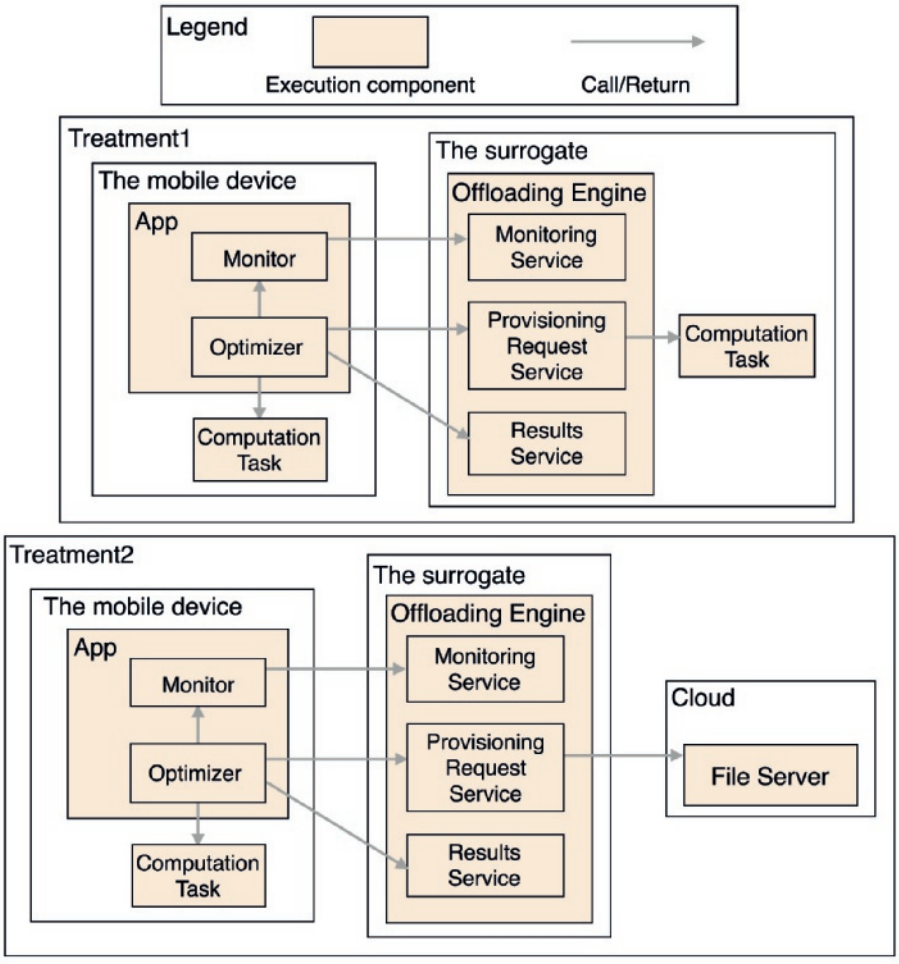


Figure 5.7: The architecture of the cyber-foraging mobile application implementing the static (Treatment 1) and the dynamic (Treatment 2) surrogate provisioning tactics

provisioning from the cloud, the provisioning parameter is a URL to a cloud-based **File Server** that provides the computation capability. However, in case of the surrogate provisioning from the mobile device, the provisioning parameter is the computation task itself that will be installed on the surrogate.

- *Results Service*: It provides the results of the computation to the mobile app. The mobile app will wait for a certain amount of time and then invoke this service to collect the results.

A cyber-foraging mobile application that implements the surrogate provisioning tactics will follow the steps in figures 5.1 and 5.2 to interact with the web services residing in the surrogate.

5.5.4 Experiment design

The experiment factor in our experimentation is the use of cyber-foraging architectural tactics for surrogate provisioning. Our factor has two treatments — Dynamic and Static — that identify the two main experimental groups. In addition, a baseline scenario with no cyber-foraging tactics deployed has been evaluated as a control group. Each group is composed of 30 trials (i.e., experimental runs).

	***Factor:* Surrogate Provisioning**	
Baseline	Static	Dynamic
Control (30 trials)	Group A (30 trials)	Group B (30 trials)

Table 5.1: The trial set for the experiment.

5.5.5 Instrumentation

All the experiments were executed in the Green Lab of Vrije Universiteit Amsterdam[3]. Figure 5.8 displays the high-level architecture of our experimentation for the different scenarios. The number of components for each scenario varies depending on the required *flexibility*. For our experimentation we used a number of hardware and software tools:

Hardware

- Test Server (HP DL360 G5): hosts the web services of our surrogate and has a LAMP server running on its Ubuntu Server 12.04 operating system.
- Mobile Device (HTC One X): runs our surrogate client program. It is based on Android OS 4.2.2.

[3]http://www.s2group.cs.vu.nl/green-lab/

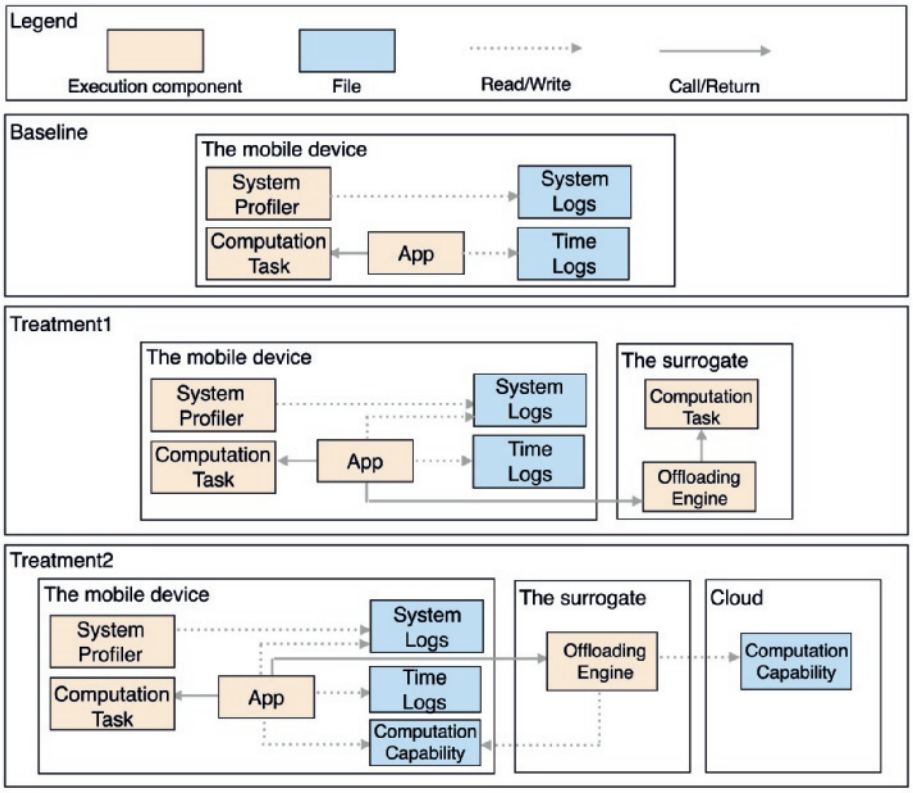


Figure 5.8: The experiment architecture of different scenarios: the baseline and both treatments

- Linksys X2000 Wireless-N Router: provides a wireless connection between the mobile device and the surrogate.

Software

- LAMP stack: hosts the web services provided by the surrogate.
- JVM: runs the byte-codes offloaded by the mobile device. The JVM is needed on both the surrogate for remote execution and the mobile device for local execution.
- PowerTutor: open-source application that logs the power consumption of different system components on a mobile device [267].

5.6 Experiment execution

In this section we describe the operational phase of our experimentation.

5.6.1 Data collection

For each scenario (*Baseline*, *Static surrogate provisioning*, *Dynamic surrogate provisioning*) we performed 30 trials. During each trial, we logged the power consumption of the mobile device using PowerTutor. Also, we recorded the timestamps of different actions in the mobile application.

Using the collected power values, we calculate the energy consumption of the mobile device based on Eq. 5.9. P_{avg} is the average power consumption of the mobile device during runtime. T_{response} shows the execution time of the task as measured by the mobile device, including (when applicable) the time required for offloading to the surrogate and receiving the results.

$$\text{Energy Consumption(J)} = P_{\text{avg}} * T_{\text{response}} \tag{5.9}$$

With regards to resilience measurements, we recorded the reaction time and the adaptation time of the cyber-foraging system when a change occurs. We do so by mapping the logged timestamps to the different execution phases.

- *Reaction time*: the period of time it takes for the optimization algorithm to decide on the execution platform, i.e., locally on the phone or remotely on the surrogate.
- *Adaptation time*: if the optimization algorithm decides for local execution, then the adaptation time will be 0. Otherwise, in case of offloaded execution, the adaptation time is measured from when the decision is made until the receipt of the notification from the *"Provisioning Request Service."*

Independent Variable	*Treatment*	*Min*	*Median*	*Mean*	*Max*
Energy consumption (J)	Baseline	61.13	94.60	96.05	152.80
	Static	7.73	8.09	12.70	43.69
	Dynamic	7.80	8.16	9.88	38.79
Power consumption (W)	Baseline	0.424	0.518	0.518	0.604
	Static	0.347	0.363	0.364	0.378
	Dynamic	0.348	0.362	0.362	0.381
Response time (s)	Baseline	121.9	181.1	185.7	286.7
	Static	22.04	22.17	34.89	120.70
	Dynamic	22.21	22.37	27.29	107.90

Table 5.2: General overview of the dataset.

5.6.2 Data analysis

During our data analysis process, we used the Shapiro-Wilk test to determine whether the normality assumption holds for our data. For hypothesis testing, we used the Wilcoxon signed rank test to determine mean differences between our samples. In addition, we report the effect sizes for our treatments using Hedges' g and Cliff's δ. We use the significance level of 0.05 in all our tests ($\alpha = 0.05$).

5.7 Results

Before we present the results of hypothesis testing, we provide an overview of our observations of the collected data. The summary of the data set is shown in Table 5.2, which reports descriptive statistics on our response variables: 1) Energy consumption, 2) Power consumption, 3) and Response time. In the table, the *Treatment* column indicates the different treatments used (*static* vs. *dynamic* surrogate provisioning) and the baseline (local execution) presented as a reference.

5.7.1 General observations

For all three variables, the baseline values follow a normal distribution (as tested by means of the Shapiro-Wilk test). With regards to our treatments, the data is normally distributed for the power consumption variable. However, the response time values of both treatments do not have a normal distribution. There is an evident significant difference in the execution time between local and the remote computation. Therefore, the non-normality of the values might be caused by the optimization algorithm, which decides on alternative platforms to execute the computation task. However, we checked the distribution of the data separately for each platform and we still detected a non-normal distribution of the response

time values. Only the response time values for each treatment were normally distributed with respect to the execution platform.

The distribution of the energy consumption values is only normal for Static Surrogate Provisioning. Also, it is interesting to note that the maximum energy consumed in our treatments (43.69 J and 38.79 J) is still lower than the minimum energy consumption of the baseline (61.13 J). This difference in energy consumption can be explained by the difference in power consumption and response time. Furthermore, Table 5.2 shows that the local execution when deploying cyber-foraging techniques (our treatments) has better performance than the baseline that does not have the overhead of the optimization algorithm.

5.7.2 Hypothesis testing

1. ***H1 (Resilience):*** we compare the resilience of both surrogate provisioning tactics based on their transition time (See Eq. 5.10). The shorter the transition time, the higher the resilience of the system.

$$\text{Transition time} = \text{Reaction Time} + \text{Adaptation Time} \tag{5.10}$$

   We first check the normality of the transition time values for our treatments. The distribution of the transition data is not normal for both treatments (Shapiro-Wilk's p-value for Treatment 1=0.0006378 and for Treatment 2=$1.995e-10$)

   Given the non-normal distribution of the values, we make use of the non-parametric *Wilcoxon signed rank* test. It shows that there is a significant difference between the median values of the two treatment samples (p-value = $5.47e-10$). As shown in the box-plot in Figure 5.9, static surrogate provisioning benefits from higher resilience compared to dynamic surrogate provisioning. In Table 5.3 we report the group medians and the effect size of the treatment. In this case, group median is reported instead of mean, as it is a more robust indicator of central tendency in presence of non-normally distributed values.

Median Transition Time (Static)	0.097
Median Transition Time (Dynamic)	0.266
Hedges'g	0.634 (medium)
Cliff's δ	0.933 (large)

Table 5.3: Group means and effect sizes of the transition time values of the treatments

2. ***H2 (Energy efficiency)***: as explained earlier, we calculate the energy efficiency of each trial based on the amount of energy consumed to perform

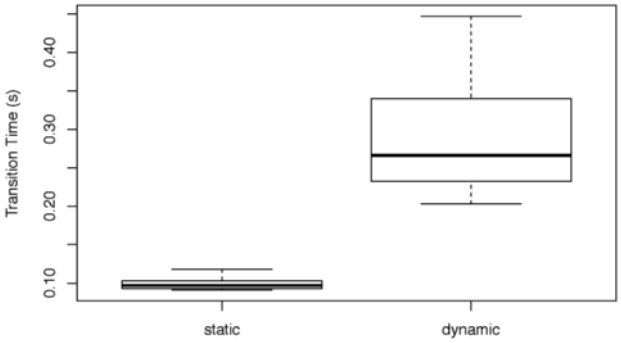


Figure 5.9: The box-plot of the transition time values of static and dynamic surrogate provisioning. Outliers are excluded from the plot.

one computation task. The execution of the computation task is platform independent, and energy consumption measures are done on the mobile device regardless where the task is executed (i.e., locally or remotely). Because the energy consumption values for different treatments do not have a normal distribution, we again use *Wilcoxon signed rank*. We cannot reject our null hypothesis ($p - value = 0.1646$) which indicates there is no significant difference between our treatments.

Median Energy Consumption (Static)	8.087
Median Energy Consumption (Dynamic)	8.161
Hedges'g	-0.296 (small)
Cliff's δ	0.21 (small)

Table 5.4: Group medians and effect sizes results on the energy consumption values of the treatments

In the box-plot of Figure 5.10 we report the energy consumption values for static and dynamic surrogate provisioning. In Table 5.4 we report the group medians and the effect size of the treatment. Also in this case, group median is reported instead of mean, as it is a more robust indicator of central tendency in presence of non-normally distributed values.

5.8 Discussion

From Table 5.2 we can already notice that there is a significant difference in the energy consumption, power consumption, and response time between the baseline and the surrogate provisioning tactics. The box-plots in Figure 5.11 clearly show

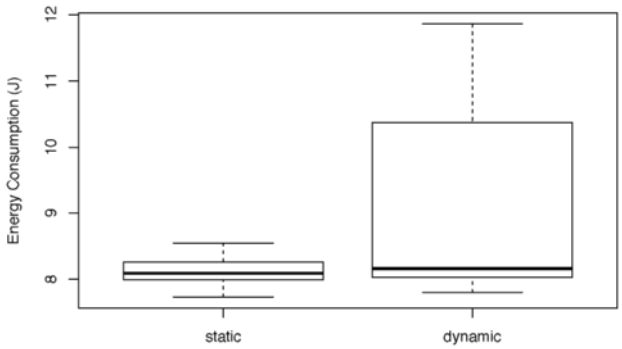
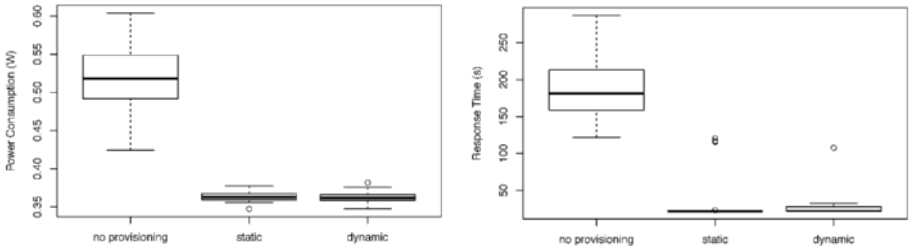


Figure 5.10: The box-plot of the energy consumption values of static and dynamic surrogate provisioning. Outliers are excluded from the plot.

this difference.



(a) Power Consumption **(b)** Response Time

Figure 5.11: The box-plots of the power consumption values and the response time of the baseline compared with our treatments: "Static Surrogate Provisioning" and "Dynamic Surrogate Provisioning"

The outliers of Static and Dynamic surrogate provisioning box-plots in Figure 5.11b represent cases when the optimization algorithm has decided not to offload the computation task to the surrogate. Surprisingly, in those cases the local execution of the task does not increase power consumption in the same way as in the baseline. The power consumption of the mobile device in presence of cyber-foraging tactics is significantly lower than a non cyber-foraging system. A possible explanation for this effect is related to the temperature of the mobile device, known to affect the variance of the battery discharge curve [267]. During our baseline measurements, the computational task was only performed locally. This resulted in a high CPU load for a longer amount of time, which in turn may have raised the temperature of the device and consequently its battery usage [267]. We tried to collect temperature data from our logs to confirm our analysis, but unfortunately the amount of data we collected was not sufficient to draw a

final conclusion. For this reason, we are preparing a replication of our experiment where we will also collect temperature data.

As discussed in Section 5.3, both Static and Dynamic Surrogate Provisioning tactics benefit from the increased *flexibility* that the runtime optimization algorithm introduces. Our expectation was that using Dynamic Surrogate Provisioning, the optimization algorithm itself would add an overhead in terms of resource usage. This overhead would result in a difference in the variable "Reaction Time" which includes the execution time of the algorithm. However, we found no significant difference (Wilcoxon signed rank test $p - value = 0.6349$) in Reaction Time among different cyber-foraging architectural tactics. This indicates that in our experimentation scenario, performing the online choice does not introduce a significant overhead. Although this result cannot be generalized to all cyber-foraging scenarios, we can conclude that Dynamic Surrogate Provisioning increases *flexibility* while not negatively influencing *performance*.

5.9 Reflection

Cyber-foraging architectural tactics offer reusable design decisions to accommodate certain types of functionality (in our case computation and data offload) while ensuring required system qualities (such as energy efficiency and resilience). We specifically focus on surrogate provisioning tactics to perform an empirical evaluation. Our work is one of its kind because it measures the effectiveness of such tactics, and it can lead to a reduction of the learning curve. Having the reusability requirement of the tactics in mind, our work, as an initial step, enriches the knowledge on the tactics with the help of quantitative insights. Therefore, software architects and software engineers can make more conscious and better-informed decisions on selecting the tactics.

In this chapter, we adopt a systematic experimentation framework to objectively collect and analyze data on the impact of surrogate provisioning tactics. As for data collection, we describe the design procedure of our experimentation step by step, which can benefit other researchers to repeat our experimentation approach, and carry out similar experimentations to study other types of tactics. As for data analysis, we perform a series of statistical tests to extract insights out of data and to validate our hypotheses. Our analysis procedure is meant to be reused by other researchers to observe findings from different measurements.

The quantitative analysis shows that Dynamic and Static provisioning deliver the promised flexibility with surprisingly negligible costs in terms of resilience and energy efficiency. We investigate resilience, a key quality attribute of sustainable systems, from the transition time perspective. We introduce a specific change that both treatments are resilient to at runtime, namely low battery level. We show that a system adopting Static Surrogate Provisioning is able to take quick action

in the presence of a change in its runtime environment. From another angle, a system with Dynamic Surrogate Provisioning can be responsive to more diverse changes. For example if the change is an error in the computation capability of the surrogate, a system with Dynamic Surrogate Provisioning can recover seamlessly by downloading the computation capability from different resources. In general, Dynamic Surrogate Provisioning has a greater self-organizing degree than Static Surrogate Provisioning. Dynamic Static Provisioning has more *flexibility* in terms of lower number of pre-assigned configurations. Consequently, it corresponds to longer transition phases for the system during runtime.

In our study, we report that both surrogate provisioning tactics perform well with no significant difference to fulfill the main objective, which is extending the limited resources lifespan. We measure the energy efficiency from the point of view of the mobile device as an example of resource-scarce environments. However, for software architects the energy efficiency of cyber-foraging systems in its entirety plays an important role as well. The cyber-foraging architectural tactics involve different components, which are utilized differently. Increasing the *adaptability* degree by adding more operational components, might influence the energy efficiency of the entire system negatively. Yet, such trade-offs need to be systematically evaluated, which is our plan for future work.

While cyber-foraging software is extremely novel (see also discussion in Section 5.10), its applicability and added value can be pervasive. As discussed in [159], cyber-foraging brings benefits to many contexts, from healthcare and emergency management to Internet of Things and wearable computing. According to the GeSI: Global e-Sustainability Initiative [101], usage of mobile devices already accounts for nearly half of the ICT sectors emissions, and is expected to steadily grow. Accommodating needs while balancing system qualities and energy efficiency will require smart software solutions such as smart architectural tactics.

Cyber-foraging was developed with resource-scarce environments in mind, where battery life and connectivity are critical. However, our results offer a glimpse of its potential for environments where flexibility is necessary because the context continuously changes (such as smart city sensing) or some resources hosting data/computation can be charged more economically or effectively than others (thanks to, for example, advances in smart grid integration of renewables, or in micro-grid applications).

5.10 Threats to validity

Our aim is to illustrate the premises and the assumptions behind our experimentation. The classification of the threats follows that by Cook and Campbell [63]. As a general consideration, in this study we are mainly interested in *theory*

testing, hence we focus more on internal and construct validity, i.e., prove that our effects are representative of the theory and caused by the outcome, rather than conclusion and external validity.

5.10.1 Conclusion validity

Threats to conclusion validity affect the statistical significance of the findings. In our experimentation, we identify the following conclusion validity threats:

- *Reliability of measures.* Our measures of energy consumption were carried out by means of the PowerTutor software tool. We chose this approach instead of hardware power meters for two main reasons: first, it was deemed more practical; second, this allowed us to measure energy and temporal data on the same device. This removes the problem of multiple data sources with consequent data synchronization and data handling operations which are arguably error-prone. Regarding the accuracy of PowerTutor, according to its developers [267] for 10s intervals, it is accurate to within 0.8% on average with at most 2.5% error.

- *Low statistical power.* As discussed in Section 5.5, our sample is a single, synthetic software application. This small sample size obviously reduces the statistical power of our test. However, our target population is also small: cyber-foraging applications are extremely novel, hence scarce and not easily portable across multiple platforms. For this reason, our results are valuable in providing solid evidence on an emerging technique.

5.10.2 Internal validity

Threats to internal validity affect the interpretation of our findings with regards to the causality link between treatment and outcome.

- *Treatment implementation.* The effects we measured on the outcome might be affected by the specific implementation of the tactics. In order to mitigate this threat, the cyber-foraging tactics were implemented following the guidance of an expert in cyber-foraging whom worked on several practical applications of the tactics. In particular, we made sure that the difference between the Static and Dynamic tactic implementation was modular enough to isolate its effects with respect to the rest of the application.

- *Maturation.* This threat is related to the effects of time on our instrumentation during the measurements. Specifically, the battery of the phone depletes and the temperature of the components varies due to physical phenomena. To mitigate this threat, we performed a randomized application of

the treatment, i.e., the application was executed in the Static Provisioning or Dynamic Provisioning version in a random order. This averaged out the effects of battery depletion and different temperatures across our repeated measurements.

5.10.3 Construct validity

Threats to construct validity affect the relationship between theory and observation. The only threat to construct validity we identify is related to the *definition of the constructs.* As argued previously, the theory behind cyber-foraging is extremely novel and the tactics we evaluated have been proposed in a limited amount of cases. For this reason, we cannot claim our implementation of the tactics can be taken as a reference for cyber-foraging theory. We mitigated this threat by involving the expert on cyber-foraging that defined and proposed the tactics we evaluated in this study. This ensures us that our implementation is a correct reification of the architectural tactics proposed in the theory.

5.10.4 External validity

Threats to external validity affect the generalization of our findings. We identified the following external validity threats:

- *Subject selection.* As discussed in Section 5.5, our study is a quasi-experiment with no randomized subject selection. This poses a clear problem of generalization. In fact, we cannot claim our results would generalize as such to a larger population of cyber-foraging applications.
- *Experimental setting.* Our instrumentation and experimental setting is based on a single mobile device and specific hardware technologies. Hence, our results might be affected by the specific experimental setting in which we operated. More evidence is needed to ensure our findings would also apply to other technologies and device families.

5.11 Related work

There are many studies on introducing architectural tactics for cyber-foraging applications e.g. [24, 59, 68, 201, 259] and presenting reference architectures and frameworks that can be adopted by software engineers to realize cyber-foraging functionalities in different systems e.g. [25, 93, 144, 231, 268]. However, in this paper, we focus on studies that provide insights on existing architectural tactics and evaluate their effectiveness on system qualities. In this respect, some work has been done on the evaluation of tactics in different domains and from

a different perspective. For instance, Wu and Kelly present a *qualitative* comparison of architectural tactics for system safety, which as a result can extend software design methodologies [254]. In another example, Harrison and Avgeriou model how different tactics can fit in different software architectural patterns from a compatibility perspective [112]. Differently, our work assesses the impact of architectural tactics with respect to energy efficiency and resilience.

In particular, we are interested in architectural tactics in the domain of cyber-foraging with an emphasis on the impact on system qualities. Related work from this perspective mostly focuses on a *qualitative* evaluation of the tactics, which usually results in design guidelines for cyber-foraging applications. Agrawal and Prabhakar present Appification, which is a methodological framework to provide guidelines for architectural design, implementation and deployment of self-adaptive mobile apps [9]. According to the Appification framework, one should analyze the quality requirements of the application and choose the best fitting tactics. The framework, however, does not provide a *quantitative* evaluation of the tactics in such applications. Orsini *et al.* provide design guidelines for mobile-cloud computing applications and include a *qualitative* analysis. They classify computation offloading solutions from the literature based on their impact on a number of system quality requirements [191]. Liu *et al.* review application partitioning algorithms for mobile-cloud computing [165]. They *qualitatively* discuss the implications of each algorithm in different usage scenarios. La and Kim present a taxonomy of computation offloading schemes, in which the schemes are evaluated in a *qualitative* manner based on five identified criteria for mobile-cloud applications [149]. Their insights help in selecting the optimum offloading scheme for target apps. Shiraz *et al.* focus on application offloading frameworks in mobile-cloud computing [229]. They introduce a thematic taxonomy to compare the existing frameworks. Abolfazli *et al.* survey cloud-based mobile augmentation approaches [6]. They introduce a comprehensive taxonomy and a number of decision-making flowcharts that can be used to build new approaches. Sharifi *et al.* review existing cyber-foraging solutions and present a categorization based on a number of factors such as the type of surrogate, the overhead of offloading, the granularity of offloading, and adopted metrics [227].

Differently, in our study, we conduct empirical experimentation to *quantitatively* evaluate cyber-foraging tactics in terms of their impact on system qualities. Our experimentation is an extension of our previous work on a decision model that helps software architects and software engineers select tactics to meet functional and non-functional requirements of cyber-foraging systems [160]. In our decision model, we review cyber-foraging tactics from several points of view such as quality impact, selection trade-offs, and dependencies between tactics. The work presented in our study complements the decision model by performing *quantitative* evaluation of tactics for energy efficiency and resilience. We place

our focus on surrogate provisioning tactics, which are one of the required tactics to build a cyber-foraging system (Section 5.2).

5.12 Conclusion

In this chapter, we answer our research question (RQ3.1), namely "How can we evaluate the effectiveness of the self-adaptation architectural tactics in different usage contexts?". This study presents an evaluation of the cyber-foraging tactics for static and dynamic surrogate provisioning. Such tactics aim to provide *adaptability* at runtime. However, the actual impact of adopting the tactics on energy efficiency and resilience of the system is not evident in the literature. We performed an empirical experiment, following the experimentation framework devised by Basili *et al.* [31], in order to analyze the cyber-foraging architectures systematically. We used the Green Lab of Vrije Universiteit Amsterdam to set up and carry out our experimentation.

Our results show a significantly higher resilience for Static Surrogate Provisioning than Dynamic Surrogate Provisioning. Also both architectural tactics improve energy efficiency compared to non-cyber-foraging architectures (our baseline measurements). However, none of the two tactics outperforms the other with respect to energy efficiency, which means that the overhead of the runtime optimization remains similar.

This chapter is a first step toward providing guidance for software architects and software engineers to minimize their learning curve on the selection of the best fitting cyber-foraging architectural tactics. Our empirical experimentation helps making better-informed trade-offs between the desired quality attributes, i.e., flexibility, resilience, and energy efficiency. In our future work, we will further quantitatively evaluate such trade-offs. Also, we will consider other cyber-foraging architectural tactics, emphasizing on runtime programmable infrastructures.

6

Case Study 3: Evaluation of Self-Adaptive Mobile Apps

This chapter performs an empirical evaluation of the self-adaptation tactic for mobile applications to answer our RQ3.1. Many software applications are executed on mobile devices. This poses new challenges on optimizing the limited capacity set by battery life without compromising energy efficiency and performance. This limitation is exacerbated by mobile applications due to their needs to transfer data. We propose a simulation framework for mobile applications to enable self-adaptability in the form of MAPE model functionalities. To provide realistic data sets to the framework, we empirically measure the resource consumption of the top 6 widely-used mobile apps. Our results show a significantly higher energy efficiency for the self-adaptation tactic compared to the baseline, which does not show any adaptability behavior. Also, in most cases the performance of the mobile apps increases.

6.1 Introduction

According to Statisica, the number of available apps in the Google Play store has increased from 300k in August 2011 to 3 million in June 2017 [234]. This remarkable growth poses new challenges for optimizing the available resources in mobile devices, such as computing power and battery life, which have limited capacity. The mobile cloud computing field addresses this limitation by designing mobile apps that rely on services hosted in the cloud, which carry on the computation or the data migration. Therefore, many mobile apps require network connections to transfer data.

Mobile network interfaces (wifi and cellular), if utilized, consume energy. Mittal *et al.* show that only the cellular network interfaces can contribute up to 50% of the total energy consumption in smartphones [177]. Given the fact that the

technologies (e.g. Wifi and 4G) employed on the network interfaces provide different network bandwidths, and have different patterns of energy consumption, network scheduling strategies are required to optimize energy efficiency and performance. The demand for scalable and energy efficient scheduling approaches has led to a number of existing solutions focusing on Wifi and cellular network interfaces. For example, packet batching and request bundling have been introduced to minimize the time that the network interfaces are at their high-power state [26, 161, 265]. Although, these solutions provide improvements in energy efficiency they do not necessarily customize for different application requirements, and perform as "one size fits all" approaches. Hence, they possibly affect the user experience negatively as they can delay the data transfer of the running mobile apps.

Mobile apps can have various network-related quality requirements; some are delay-sensitive (their objective is to minimize the network delay to transfer data), some are long-lived (their objective is to transfer data for a longer period of time compared to other apps), and some are data-intensive (their objective is to transfer large amounts of data over the network). We need network simulation tools as a testbed to extensively evaluate the achievement of application quality requirements when different network scheduling strategies are deployed on a mobile device. Although there are a number of general purpose network simulators that can be adopted for mobile devices [77, 134] and a few mobile-specific network simulators [57, 116], the mobile computing field has not attracted a lot of attention on mobile-specific network simulators with an emphasis on energy efficiency [36]. In this chapter, we evaluate the impact of self-adaptability of mobile applications on energy efficiency of a mobile device. To simulate self-adaptability in mobile applications, we empirically measure the resource consumption of the top 6 widely-used mobile apps with different configurations (high-consuming and low-consuming states). We introduce a mobile network scheduling framework to extensively evaluate network utilization by mobile apps, and to simulate the network interfaces with several built-in energy states. We develop the framework in the form of MAPE model functionalities (Monitor-Analyze-Plan-Execute) [46] in a modular fashion that can be extended with new components (e.g. adding new scheduling strategies, or adding network technologies other than Wifi and 4G). Our main contributions are:

1. We follow a systematic approach to measure the resource consumption of mobile apps in a controlled environment.

2. We develop a network scheduling framework in order to facilitate the analysis of network energy efficiency of mobile devices. It simulates the network interfaces (e.g. Wifi) in Android including all the states to start a network connection, network technologies, and energy models extracted from

the peer-reviewed existing literature. The framework provides a flexible testbed, in which different scheduling strategies and application scenarios can easily be added.

3. We compare the energy efficiency of mobile devices under different application scenarios. Application scenarios flexibly describe traffic patterns generated by the mobile apps. Also, quality requirements such as being delay-sensitive or delay-tolerant are specified as input for the scheduling strategies that challenge optimizing the trade-off between energy efficiency and performance.

This chapter is organized as follows: Section 6.2 includes an introduction on mobile network interfaces, their energy states, and the adopted power models. In Section 6.3, we describe our experimental framework, its main components and its embedded scheduling strategies. We define the scope and the goal of our experimentation in Section 6.4. Section 6.5 presents the experimentation planning including context selection, variable selection, hypothesis formulation, subject selection, experiment design, and instrumentation. We describe thoroughly the steps taken to execute the experiments in Section 6.6. Our findings and reflections are discussed in Sections 6.7 and 6.8. We describe the threats to validity and their mitigation in Section 6.9. Existing related work on simulation tools and network scheduling solutions for mobile apps are discussed in Section 6.10. Section 6.11 provides reflection points that emerged from our findings. Finally, in Section 6.12 we conclude the chapter, and we introduce directions for future work.

6.2 Background

Over the past years, the usage of mobile devices has increased. For instance, statistics compare the number of website visits from mobile devices in 2016 and 2017, which grows from 57% to 63% [82]. The mobile network interface itself is energy consuming with or without data transfer. It has multiple energy states that cause different measures of energy consumption. Figure 6.1 shows the four energy states (idle, promotion, transfer and tail) and the order in which they can occur during the data transfer.

In the *idle* state, the network interface does not transfer data. Consequently, the interface consumes a low amount of energy. The interface stays in this state until a new request for data transfer is created by mobile apps. When an application request arrives at the network interface, the *promotion* state starts, in which the interface powers up to get ready for data transfer. In the promotion state no data will be transferred and the duration to remain in this state is fixed.

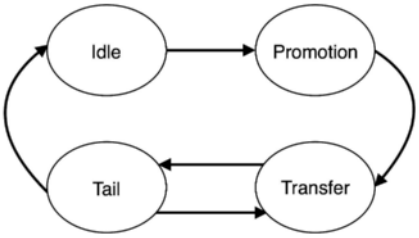


Figure 6.1: The four energy states for a mobile network interface: idle, promotion, transfer and tail

After promotion, the *transfer* state starts, in which the requested data can be transferred. The network interface will be in the high-power state, and its energy consumption will almost be linear to its utilization rate. The duration of this state highly depends on the amount of data to be transferred. Finally, the *tail* state starts. In this state, the network interface waits for application requests in a low-power state, in which no data is transferred. The idea with the tail state is to skip the overhead of the promotion state. So, the interface will wait for new transfer requests for some fixed time before entering the idle state. After this state, the network interface can enter either the transfer state or the idle state again.

Although wifi and cellular network interfaces follow the same order of energy states, the employed technologies (e.g. Wifi and 4G) are very different in states' duration and maximum data rates. Studies show that Wifi G mode (802.11g), which we refer to as the wifi technology in this chapter, remains shorter in the promotion and the tail states compared to the 4G technology [121]. However, Wifi has a lower data rate than 4G. The maximum throughput provided by Wifi based on the 802.11g for downlink and uplink is 54Mbps, while 4G provides up to 300Mbps for downlink and 75Mbps for uplink [4, 187]. It is worth mentioning that Wifi standards vary in the resulting throughput. For instance, Wifi N mode (802.11n) outperforms both wifi G mode and 4G [194]. These differences between wireless standards allow us to make runtime tradeoffs that select the best configuration plan of the network interfaces according to the quality requirements of the mobile apps. In this study, we perform our experimentation only on Wifi G mode configurations since we could empirically measure the resource consumption of the Wifi network interface when launching mobile applications.

6.2.1 The energy states

In our experiments, we estimate the energy consumption of the network devices of a mobile device. To do so, we adopt the energy models presented in [121] based on power traces measured in an Android smartphone. Table 6.1 summarizes the

Table 6.1: The power consumption and the duration of the promotion and the tail energy states of Wifi and 4G, adopted from [121]

Technology	***Energy State***	***Power (mW)***	***Duration (ms)***
Wifi	Promotion	124.4	79.1
	Tail	119.3	238.1
4G	Promotion	1210.7	260.1
	Tail	1060.0	11576.0

Table 6.2: The power consumption and the duration of the sleep mode and the wake-up mode in the idle state of Wifi and 4G, adopted from [121]

Technology	P_{base} ***(mW)***	$T_{cycle} - T_{on}$ ***(ms)***	P_{on} ***(mW)***	$T_{on}(ms)$
Wifi	11.4	300.6	77.2	7.6
4G	11.4	1237.0	594.3	43.2

power consumption and the duration of promotion and tail energy states. As it can be seen, the power consumption and the duration of these states in 4G is significantly higher compared to Wifi. However, the higher data rates provided by 4G make it an interesting technology to employ. We calculate the energy consumption by multiplying the values of power consumption and duration.

The energy consumption of the idle state with duration T_{idle} can be calculated with Equation 6.1. In the idle state, the network interface periodically (every T_{cycle}) gets into the wake-up mode with power consumption P_{on} and duration T_{on}, and the sleep mode with power consumption P_{base} and duration $T_{cycle} - T_{on}$. To calculate the energy consumption in this state, we need to find the number of such cycles. For simplicity, we assume T_{idle} is divisible by T_{on}. Table 6.2 lists the corresponding values for Wifi and 4G.

$$E_{idle} = \frac{T_{idle}}{T_{cycle}}(T_{on} * P_{on} + (T_{cycle} - T_{on}) * P_{base}) \tag{6.1}$$

The energy consumption of the transfer state highly depends on the type of the data transfer, namely upload and download. Equation 6.2 calculates the energy consumption of the transfer state with duration $T_{transfer}$. β is the base power consumption of the interface. Depending on the type of transfer, the slope of energy consumption is defined with α_u and α_d. D_u and D_d show the data rates for uplink and downlink, respectively. Table 6.3 lists the corresponding values for Wifi and 4G.

$$E_{transfer} = T_{transfer}(\alpha_u * D_u + \alpha_d * D_d + \beta) \tag{6.2}$$

Table 6.3: The values constructing power consumption in the transfer state of Wifi and 4G, adopted from [121]

Technology	$\alpha_u(mW/Mbps)$	$\alpha_d(mW/Mbps)$	β
Wifi	283.17	137.01	132.86
4G	438.39	51.97	1288.04

6.3 Our Experimental Framework

We present a simulation framework to assess the energy efficiency of self-adaptive mobile apps. Self-adaptability helps mobile apps recover from runtime changes, which can improve/degrade the achievement of qualities at runtime. For instance, a sudden drop in the battery level of a mobile device is considered as a negatively-impacting runtime change for the running mobile apps. When a runtime change is detected, the framework realizes self-adaptability in two forms, namely software reconfiguration and infrastructure reconfiguration.

- **Software Reconfiguration** At the software architectural level, reconfigurations are achieved by modifying the availability of the app features. At design time, app features are designed in a way that they can be disabled on demand at runtime. Therefore, if there is shortage on resources (e.g. battery level or network bandwidth) some high resource-consuming features can be replaced with low resource-consuming ones. Figure 6.2 shows how a self-adaptive mobile app can transition between software reconfigurations, namely app states. Both the states illustrated in the figure aim at performing one specific operation. The difference between the states relies on the number of available features in the app, which can eventually result in different resource consumption. The transition between the states is triggered by the detection of runtime changes, such as lack of resources.

- **Infrastructure Reconfiguration** At the infrastructural level, reconfigurations are realized as rescheduling the available resources at runtime. For instance, the network resources are reallocated to the running mobile apps based on their priorities and quality requirements. In this work, we turn our focus on scheduling specifically the network resources, which are considered as essential types of resources for modern mobile apps that extensively rely on cloud-based services. The framework is capable of choosing different types of network interfaces (Wifi, 3G, and 4G) to transfer data depending on the quality requirements of running apps. The framework optimizes the duration of the network interfaces in the promotion and the tail states. The transfer requests of the mobile apps are queued up to be transferred. As soon as the network interface is active and in its transfer state, the requests

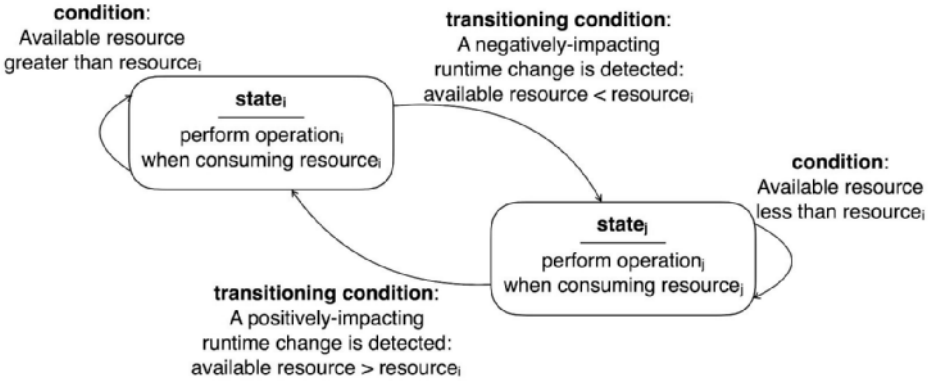


Figure 6.2: A state machine consisting of two states for performing specific operation with different levels of resource consumption. We assume resource i > resource j

are scheduled. If there is more than one request at a given time, they will fairly share the available bandwidth. If the bandwidth capacity is not sufficient to satisfy the requirements of all the requests at a given time, the in-progress requests get higher priority than the new requests.

6.4 Experiment Definition

We aim to analyze the effectiveness of the self-adaptation tactics on improving energy efficiency and performance of mobile applications. We follow the well-known Basili framework [31] to plan our experimentation. The following GQM paradigm describes our goal, research questions and the metrics selected for this study.

- **Goal**

  We formulate the goal of this chapter as:"Analyze self-adaptation tactics for the purpose of evaluation with respect to energy efficiency and performance from the viewpoint of software architects and software engineers in the context of mobile applications". Self-adaptation tactics are our objects, which we evaluate based on two quality attributes: performance and energy efficiency. Our results can be beneficial for software architects and software engineers to make design-time trade-offs between performance and energy efficiency. We focus on the context of mobile applications but our findings can be generalized for software design approaches in general. We compare the effectiveness of the self-adaptation tactic with a baseline control group, in which the mobile apps do not adapt to runtime changes.

- **Research Questions**

  We define the following research questions to approach our goal in a systematic way:

  RQ1: What is the impact of the self-adaptation tactic on energy efficiency?

  We analyze the energy efficiency of the self-adaptation tactic at runtime. We define energy efficiency as the number of joules consumed in the network interface per usage scenario (the sequence of transfer requests from the running mobile apps). We estimate the energy consumption of the network interface using the deployed energy models in our framework. The energy models rely on the power consumption and transfer duration for each transfer request.

  RQ2: What is the impact of the self-adaptation tactic on performance?

  We evaluate the performance of the self-adaptation tactic compared to our baseline. We define performance as the time it takes for a usage scenario to be completed, namely time-to-complete. Usage scenarios consist of a series of transfer requests generated by mobile apps. We measure the time difference between the end time of the last request and the start time of the first request in the usage scenario.

- **Metrics** We measure the following predefined metrics that help us answer our research questions in a quantifiable manner.

  - **Network energy consumption**: the number of joules consumed in the mobile network interface to transfer all the requests of the mobile apps.
  - **Time-to-complete**: the total duration that it takes for the mobile apps to transfer all their requests.

The GQM graph in Figure 6.3 indicates how the metrics relate to our research questions. *RQ1* depends on the metric "Network energy consumption" and *RQ2* relies on the metric "Time-to-complete".

6.5 Experiment Planning

We describe our experimentation in the following sections: Variable Selection, Hypothesis Formulation, Subject Selection, Experiment Design, and Instrumentation.

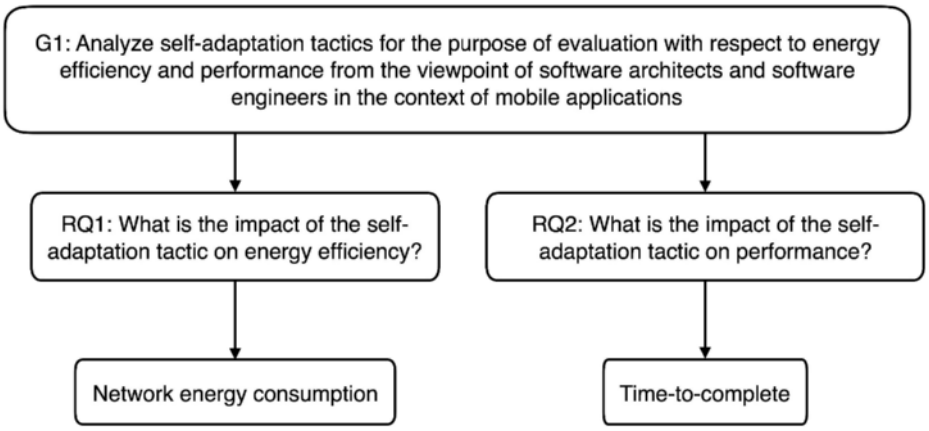


Figure 6.3: The GQM graph summarizing the relation between the goal of our experiment, the research questions, and the metrics

6.5.1 Variable Selection

The metrics identified in the previous section, namely network energy consumption and time-to-complete are our dependent variables, which help us find answers for our research questions.

Using the independent variables we control the experimentation environment. Our main independent variable is the deployment of the self-adaptation tactic in our framework that we define as a factor. Our two treatments in this experimentation are:

- **Experimental Treatment**: In this treatment, the mobile apps have two different energy-consuming states, which they can transition between. We realize self-adaptability of mobile apps with this treatment.
- **Control Treatment**: In this treatment, the mobile apps are indifferent to contextual runtime changes (e.g. low battery level, low bandwidth capacity). They can not adjust their resource utilization as they only have one energy-consuming state. The measurements of this treatment are used as a reference set to compare the effectiveness of the experimental treatment.

We have implemented an optimization algorithm that schedules the transfer requests for available network interfaces in the mobile device. The objective with the algorithm is to minimize the energy consumption for transferring all the requests. Given the fact that in real-life scenarios network bandwidth can fluctuate, in our experimentation we simulate the same behavior by introducing a possibility of low/mid/high capacity bandwidth. This defines the *available bandwidth*

independent variable. Also, running applications in the mobile device can send their transfer requests in any order. Therefore, we define *applications order* as one of our independent variables, which can be controlled in our framework. Our optimization algorithm is sensitive to both the runtime network changes and the applications order. The algorithm adjusts the scheduling plan for the applications accordingly.

6.5.2 Hypothesis Formulation

We define *Null* and *Alternative* hypotheses for each of our research questions.

- *RQ1: What is the impact of the self-adaptation tactic on energy efficiency?*

  Eq. 6.3 shows the difference in energy efficiency of the self-adaptive tactic and the baseline. $EE_{adaptive}$ is the energy efficiency of the mobile device when employing the self-adaptive tactic and EE_{base} is the energy efficiency of the mobile device with non-adaptive mobile apps. The null hypothesis in Eq. 6.4 indicates that the energy consumption of the mobile device when employing the self-adaptive tactic is equal to the energy consumption of the mobile device when using the baseline. Differently, the alternate hypothesis in Eq. 6.5 shows that the energy consumption of the mobile device is lower when employing the self-adaptive tactic.

$$\Delta EE = EE_{adaptive} - EE_{base} \tag{6.3}$$

$$H1_0 : \Delta EE = 0 \tag{6.4}$$

$$H1_a : \Delta EE < 0 \tag{6.5}$$

- *RQ2: What is the impact of the self-adaptation tactic on performance?*

  As mentioned before, we measure performance using the time-to-complete metric (TTC). So, mobile apps will have higher performance if they have shorter time-to-complete to transfer all their requests. Eq. 6.6 shows the impact of the self-adaptation tactic on TTC, which is measured in seconds. $TTC_{adaptive}$ shows the time-to-complete of mobile apps when employing the self-adaptation tactic and TTC_{base} shows the time-to-complete without any adaptive behavior by mobile apps. The null hypothesis in Eq. 6.7 indicates that the time-to-complete of the mobile apps when employing the self-adaptive tactic is equal to the time-to-complete of the mobile apps when employing the baseline, while the alternate hypothesis in Eq. 6.8 shows

that the time-to-complete of the mobile apps is lower when employing the self-adaptive tactic.

$$\Delta TTC = TTC_{adaptive} - TTC_{base} \tag{6.6}$$

$$H2_0 : \Delta TTC = 0 \tag{6.7}$$

$$H2_a : \Delta TTC < 0 \tag{6.8}$$

6.5.3 Subject Selection

The object of this experimentation is our proposed self-adaptation tactic for mobile apps. As our subjects, we select the top 6 most-used mobile apps, namely Facebook, Youtube, Google Maps, Google Search, Instagram and Facebook Messenger [153]. This defines our experiment as a **quasi-experiment**, in which the subjects are not selected randomly. We further discuss in Section 6.9 that our subjects are still representative of mobile apps and the results of our experiments provide helpful insights in the research field.

Each app offers a set of configurations that can reduce the consumption of resources. For example, in Youtube, one could select for lower data rates when streaming videos and in Facebook Messenger, one could make a voice call instead of a video call. We have analyzed the apps with respect to their configuration possibilities. This helped us draw finite state machines that show the transition between high-consuming and low-consuming states for the apps. For some of the apps, transitioning to a target state (either high-consuming or low-consuming) requires an intermediate state that helps changing the right configurations for the mobile app.

6.5.4 Experiment Design

To evaluate the effectiveness of our self-adaptation tactic, we define a list of factors (independent variables) to form our trial groups:

- *Available bandwidth*: The applications require a minimum bandwidth to transfer their requests. We define this minimum value as the high-capacity bandwidth, in which the applications compete the least for the resources. This factor varies among three different capacity ranges: [0%-30%] as low-capacity, [30%-70%] as mid-capacity and [70%-100%] as high-capacity. In our framework, we change the available bandwidth for the applications from 0% to 100% of their required capacity.

	Factors	
Trial Groups	***Bandwidth capacity***	***Apps order***
Self-adaptation tactic	high-capacity (30)	high-to-low (30)
	mid-capacity (30)	low-to-high (30)
	low-capacity (30)	zig-zag (30)
Baseline	high-capacity (30)	high-to-low (30)
	mid-capacity (30)	low-to-high (30)
	low-capacity (30)	zig-zag (30)

Table 6.4: The number of trials for our experimental and control groups.

- *Applications order*: In the usage scenarios, which will be used as input by the framework, the order of applications might impact the results of our dependent variables (i.e. energy efficiency and time-to-complete). We define three types of applications orders: *high-to-low*, in which high-demand applications start earlier than low-demand application, *low-to-high*, in which low-demand applications start sending their requests first and *zig-zag*, in which high-demand and low-demand applications are combined together and placed one after another.

We have one experimental group that implements the self-adaptation tactic, and one baseline control group. Table 6.4 summarizes the number of trials we have for each factor.

6.5.5 Instrumentation

We conducted our experimentation in the Green Lab of Vrije Universiteit Amsterdam. Figure 6.4 shows the high-level design of our experimental framework. Our treatment consists of two main parts namely, the *network scheduler* and the *mobile app wrapper*, which are shown in different colors.

The *mobile app wrapper* helps simulating the self-adaptive behavior for typical non-adaptive mobile apps. Therefore, our framework embeds *mobile apps*, which imitate the behavior of typical apps with the help of the empirical measurements, and then wraps them with the specific components of the app wrapper such as *Adaptive status* and *Software reconfiguration*. The *network scheduler* has built-in scheduling strategies to schedule the available resources in the mobile device for the running mobile apps. The framework realizes the MAPE functionalities with the help of these components, as described in the following:

- *Monitor*: *Application profiler* and *efficiency profiler* collect data based on measured/estimated pre-defined metrics. Application profiler monitors the mobile apps to check whether the requirements of the apps are met. Along

with the application-specific data, Efficiency profiler monitors the availability of resources in the mobile device such as computing power or network bandwidth.

- *Analyze*: The collected data on the mobile apps and the resources, are the input for the *runtime change* component. Runtime change analyzes the data in comparison to expected patterns of resource consumption for the mobile apps. This leads to detection of dynamic changes at runtime.
- *Plan*: The next step is to find the best-fitting plan to adapt to the runtime change. *Adaptive status* offers a list of adaptation possibilities that can be applied to mobile apps to recover from the detected change. Likewise, *Resource scheduler* provides a list of configuration options regarding the available resources in the mobile device. From both lists, one solution will be accepted that we refer to as the adaptation plan.
- *Execute*: As the final step in the MAPE paradigm, the adaptaion plan will be executed. The *software reconfiguration* component applies the plan to the running mobile apps and the *infrastructure reconfiguration* applies the plan to the resources.

6.6 Experiment Execution

To execute our experimentation, we mainly follow two steps: data collection and data analysis.

6.6.1 Data Collection

In order to make our simulation more realistic, we collect the real resource consumption values from our subjects. Therefore, the data collection has two phases in our experimentation, namely empirical data collection and simulation data collection. The former points to the fact that we launch a selected list of applications in a mobile device and empirically collect data on their resource consumption. We use the measurements as input to our simulation framework. The latter, differently, refers to the techniques we employ to collect simulation data out of our framework. In the following we describe each of the data collection types independently.

For *empirical data collection*, we measure the resource consumption of the selected apps on each of their defined states. For each of the selected apps we have defined a typical usage scenario based on our best of knowledge. For instance, finding directions and streaming videos are identified for Google Maps

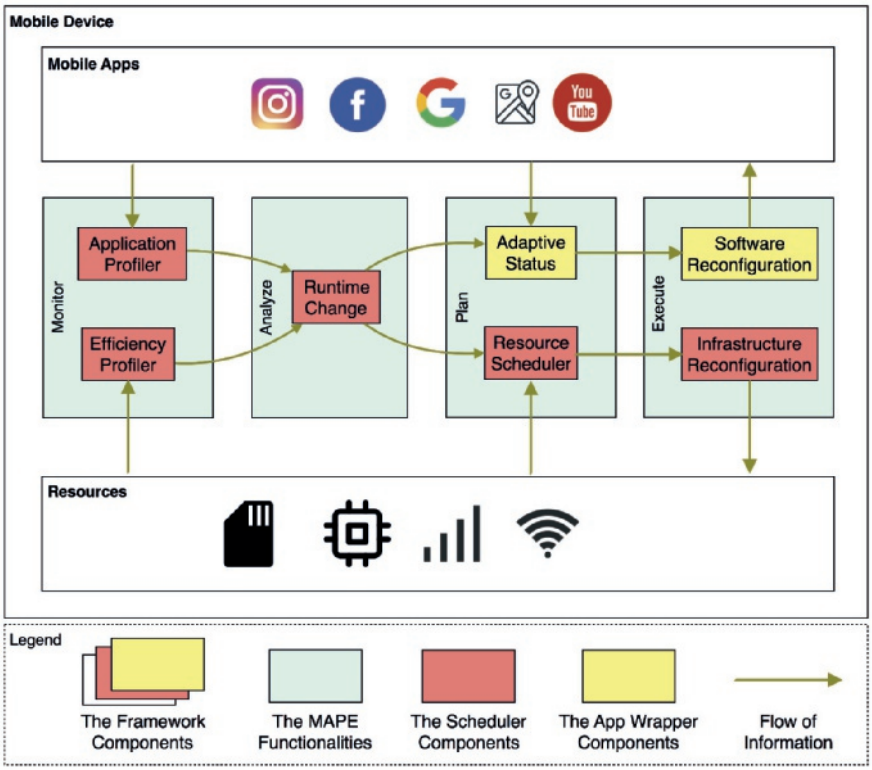


Figure 6.4: The main components of our framework categorized by their relevance to the MAPE model functionalities.

and Youtube, respectively. Table 6.5 summarizes the number of trials for the apps and their states. As it is clear from the table, not all the apps require a transitioning state, for which we use *N/A* as an indicator.

We use Trepn Profiler[1], which is a power and performance profiling application, to identify the utilization pattern of each running application in the mobile device. Additionally, we use Wireshark[2] to analyze the network traffic generated by each application. We use the collected data to define high-consuming and low-consuming energy states for the apps as an input to our simulation framework.

For *simulation data collection*, we use energy models to estimate the energy consumption of the network interfaces in the mobile device. The models we use are based on power consumption rates. Therefore, for each trial we log the power consumption of the network interfaces every second of the execution period. This

[1]https://developer.qualcomm.com/software/trepn-power-profiler
[2]https://www.wireshark.org/

App	***High-consuming State***	***Low-consuming State***	***High-low Transitioning State***	***Low-high Transitioning State***
Facebook	10	10	10	10
Facebook Messenger	10	10	N/A	N/A
Google Maps	10	10	N/A	N/A
Google Search	10	10	N/A	N/A
Instagram	10	10	10	10
Youtube	10	10	N/A	N/A

Table 6.5: The number of trials for each resource-consuming state of the selected apps

helps us calculate the energy consumption of the network interfaces using Eq. 6.9, in which P shows the power consumption and $T_{\text{execution}}$ shows the execution time.

$$\text{Energy Consumption(J)} = P * T_{\text{execution}} \tag{6.9}$$

Regarding the time-to-complete metric, we profile the start time and the end time of transfer requests generated by the mobile apps. The time difference between the largest end time and the smallest start time gives us the total time-to-complete for all the apps.

6.6.2 Data Analysis

The output of our data collection step is a set of log files, which we analyze using the statistical Python libraries. In order to test the normality of our data sets we use the Shapiro-Wilk test. The null hypothesis for this test is that the data set is normally distributed. Using the output p-value we can reject the null hypothesis and draw conclusions on the normality of data sets.

We use the non-parametric MannWhitney U test to test our one-sided hypotheses. The null hypothesis for this test examines if randomly selected values from one data set are smaller or larger than randomly selected values from another data set. We use this test to check whether the results of our self-adaptation tactic are lower than the results of the baseline regarding energy consumption and time-to-complete. In addition to the MannWhitney U test, we calculate Hedges' g, which measures the effect sizes of our trial groups. This test targets the differences between the mean values of two data sets. If the calculated g is in the range of [-0.5, 0.5] the effect size is considered as small and otherwise the effect size is large.

We use the significance level of 0.05 in our tests ($\alpha = 0.05$). It means if the calculated p-value is lower than 0.05, we can reject the null hypothesis of our tests with 95% of confidence interval.

6.7 Results

We categorize our findings based on their relevance to our research questions, as the following:

6.7.1 Results regarding energy consumption (RQ1)

Table 6.6 summarizes the collected data for each of the factors and trial groups. It shows that for all the factor values, the mean and the median of the results are lower when employing the self-adaptation tactic. However, the resulting energy consumption of employing the self-adaptation tactic varies in some cases. The self-adaptation tactic performs the best when the bandwidth capacity has the middle range (*mid-capacity*) and it performs the worst when the apps order is from high-demanding to low-demanding applications (*high-low*). We observe some variations for the baseline as well. When the bandwidth capacity is in its highest range (*high-capacity*), the energy consumption of the mobile device is minimized and when the apps order follows the zig-zag pattern (*zig-zag*), the energy consumption is maximized.

The box-plots in Figure 6.5 compare the energy consumption of the mobile device for different bandwidth capacities. When the capacity is in its mid-range, the self-adaptation tactic certainly outperforms the baseline. With mid-capacity the applications at some moments need to compete over network resources. Given the fact that with the self-adaptation tactic the apps are able to switch between their states, the apps can tune their resource consumption according to the changes in the capacity availability. Differently, the baseline does not provide any self-adaptability quality to the mobile apps. Therefore, if there is limitation on the available capacity, the mobile apps can not adjust themselves. They will remain in their high-consuming states, which cause longer transfer periods and consequently, higher energy consumption.

With high-capacity bandwidth, we see that the energy consumption of both trial groups stays closer. The reason is that the mobile apps receive all the resources they need. Therefore, there is not many competitions over resources. When the bandwidth capacity is in its lowest range, the mobile applications are challenged the most to get the resources they need. For the baseline in which the applications are not adaptive to the contextual changes, the energy consumption increases the most compared to the self-adaptation tactic.

Independent Variable	***Factor***	***Factor Value***	***Trial Group***	***Min***	***Median***	***Mean***	***Max***
Energy Consumption (mJ)	Bandwidth Capacity	low-capacity	Self-adaptive	7959.6	15442.7	14929.2	31782.0
			Base	23778.1	27137.2	27350.2	30769.8
		mid-capacity	Self-adaptive	9129.2	9349.4	9436.7	9784.1
			Base	15641.0	16288.2	16342.8	17553.7
		high-capacity	Self-adaptive	8460.6	9705.5	9606.3	11644.6
			Base	10405.0	11048.5	10920.1	11478.5
	Apps Order	low-high	Self-adaptive	7924.4	10039.0	11092.9	16470.1
			Base	23674.4	27495.5	27117.9	30049.5
		high-low	Self-adaptive	7781.7	16143.8	17823.1	34256.8
			Base	22649.8	26591.0	26397.7	30063.5
		zig-zag	Self-adaptive	7815.8	15068.7	13801.4	28685.4
			Base	24441.4	27085.4	27478.0	33558.3

Table 6.6: General overview of the collected energy measurements.

The box-plots in Figure 6.6 compare the energy consumption of the mobile device for different applications orders. It is important to mark that the bandwidth capacity is fixed with the low range. Therefore, the applications need to compete over available resources. For the baseline it almost does not matter which order the applications follow. It is because the applications in the baseline trial group do not adjust themselves according to the capacity changes. The story is different for the self-adaptation tactic. When the applications are ordered as "low-high" and "zig-zag", the self-adaptation tactic outperforms the baseline. It can be explained using the fact that low-demanding applications, which start earlier than the high-demanding ones, get higher priority when allocating resources. This means that the high-demanding applications have higher changes to switch to their low-consuming states. For the same reason, we observe a wider range of energy consumption when the applications start with the order "high-low".

In Table 6.7 we report Mann Whitney's p-value and Hedge's g for both trial groups regarding their energy consumption. For all the factor values, Hedge's g shows a large effect size and we can reject our null hypothesis on energy consumption based on the p-values.

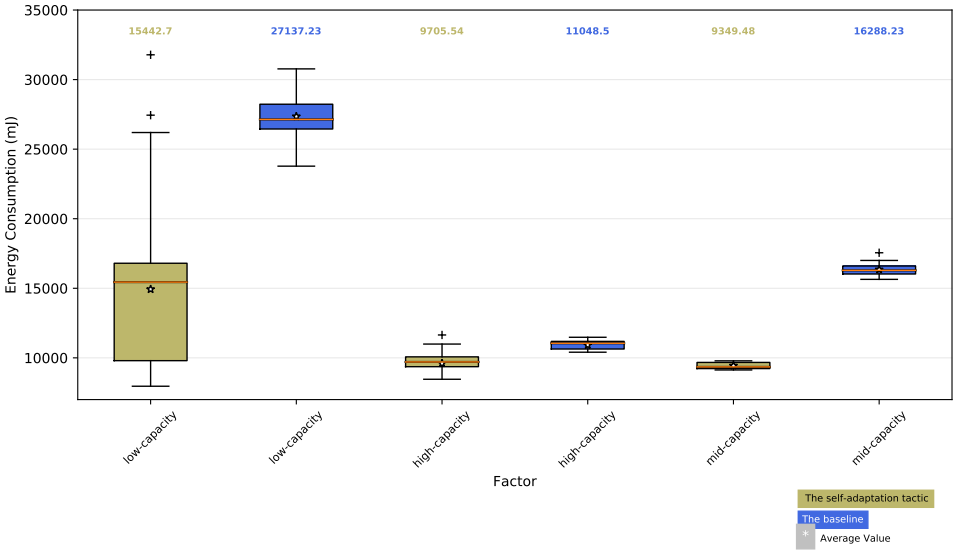


Figure 6.5: The box-plots of the energy consumption results of the self-adaptation tactic and the base for the bandwidth capacity factor.

Factor	***Factor Value***	***Mann Whitney (p-value)***	***Hedge's g***
Bandwidth Capacity	low-capacity	2.28e-09	-1.98 (large)
	mid-capacity	1.50e-11	-12.56 (large)
	high-capacity	1.01e-09	-1.48 (large)
Apps Order	low-high	1.50e-11	-4.21 (large)
	high-low	0.00013	-0.98 (large)
	zig-zag	1.57e-10	-3.18 (large)

Table 6.7: p-values and effect sizes of the energy consumption values of our trial groups

6.7.2 Results regarding time-to-complete (RQ2)

Table 6.8 summarizes the measured time-to-complete for each of the factors and trial groups. When the bandwidth capacity is in its highest range, the self-adaptation tactic and the baseline result in very similar time-to-complete values. For all the other cases, the mean and the median of the results are lower for the self-adaptation tactic. The low bandwidth capacity (*low-capacity*) causes a wider range of time-to-complete values for the self-adaptation tactic. For the apps order factor, in which the bandwidth capacity is set to its low range, the

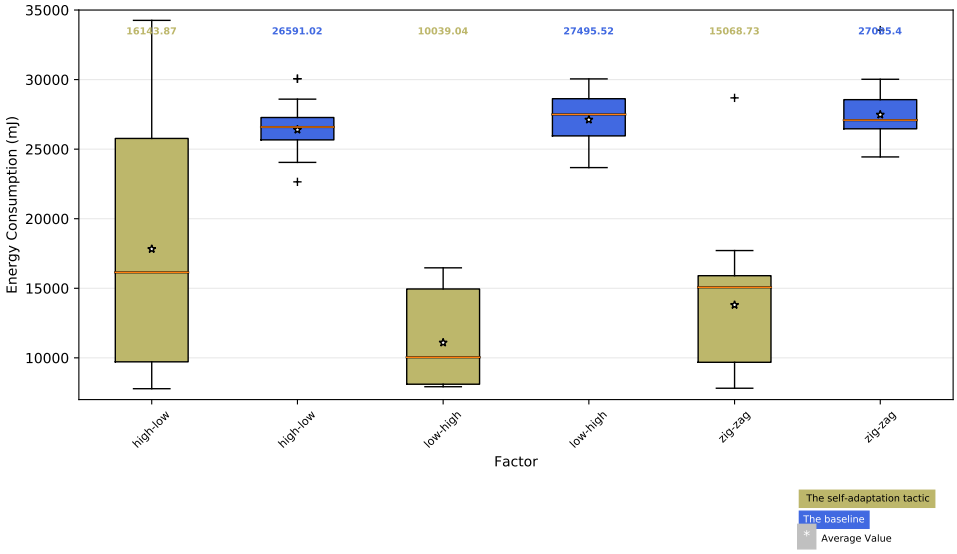


Figure 6.6: The box-plots of the energy consumption results of the self-adaptation tactic and the base for the applications order factor.

time-to-complete of the self-adaptation tactic has its highest values when the applications are ordered from high-demanding to low-demanding (*high-low*).

The baseline shows variations as well. With the high range of capacity bandwidth (*high-capacity*), the time-to-complete remains the shortest and with the low-demanding to high-demanding applications order (*low-high*), the time-to-complete is maximized.

The box-plots in Figure 6.7 compare the time-to-complete of the applications for different bandwidth capacities. We observe that the results of the baseline and most of the results of the self-adaptation tactic (except with the low-capacity setting) follow a normal distribution.

When the capacity is in its high-range, the self-adaptation tactic and the baseline have similar time-to-complete. For the low-capacity and the mid-capacity values, the self-adaptation tactic certainly outperforms the baseline. With these settings, the applications need to compete over network resources. The self-adaptation tactic gives the applications the opportunity to switch to their low-consuming states, which utilize less resources and usually have less number of transfer requests. Therefore, the time-to-complete resulting from the self-adaptation tactic is smaller.

The box-plots in Figure 6.8 compare the time-to-complete of the applications for different applications orders. It should be noted that the bandwidth capacity

Independent Variable	***Factor***	***Factor Value***	***Trial Group***	***Min***	***Median***	***Mean***	***Max***
Time-to-complete (s)	Bandwidth Capacity	low-capacity	Self-adaptive	141.0	175.0	172.9	240.0
			Base	228.0	259.0	257.8	265.0
		mid-capacity	Self-adaptive	146.0	151.0	150.8	151.0
			Base	238.0	239.0	239.2	241.0
		high-capacity	Self-adaptive	146.0	151.0	150.8	151.0
			Base	151.0	151.0	151.0	151.0
	Apps Order	low-high	Self-adaptive	140.0	148.0	156.4	179.0
			Base	218.0	258.0	252.6	268.0
		high-low	Self-adaptive	140.0	172.0	182.7	242.0
			Base	212.0	251.0	244.8	261.0
		zig-zag	Self-adaptive	142.0	171.5	164.8	237.0
			Base	237.0	255.0	255.1	267.0

Table 6.8: General overview of the collected time-to-complete measurements.

is fixed with the low range to evaluate the applications order. Therefore, the applications need to compete over available resources. The applications order does not impact the time-to-complete for the baseline. We observe more variations for the self-adaptation tactic. In all applications orders, the self-adaptation tactic outperforms the baseline. However, we see variations for the self-adaptation tactic. The high-low order results in the widest range of time-to-complete when employing the self-adaptation tactic. With the high-low order, high-demanding applications that start earlier will get a higher priority to transfer their requests. This means the chance to switch to low-consuming states is decreased for such apps.

In Table 6.9 we report the Mann Whitney's p-value and Hedge's g for both trial groups regarding their time-to-complete. For all the factor values except the high-capacity bandwidth, Hedge's g shows a large effect size and we can reject our null hypothesis on time-to-complete based on the p-values. We can not reject the null hypothesis when the bandwidth is in its highest range.

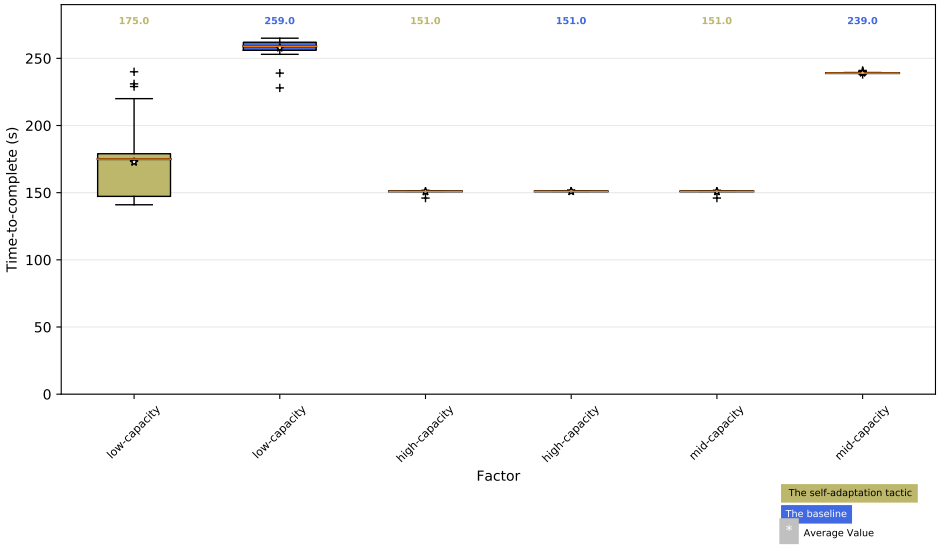


Figure 6.7: The box-plots of the time-to-complete results of the self-adaptation tactic and the base for the bandwidth capacity factor.

Factor	***Factor Value***	***Mann Whitney (p-value)***	***Hedge's g***
Bandwidth Capacity	low-capacity	2.20e-11	-3.07 (large)
	mid-capacity	7.21e-12	-84.40 (large)
	high-capacity	0.21	-0.18 (small)
Apps Order	low-high	1.48e-11	-4.52 (large)
	high-low	5.42e-09	-1.41 (large)
	zig-zag	1.61e-11	-4.54 (large)

Table 6.9: p-values and effect sizes of the time-to-complete values of our trial groups

6.8 Discussion

Our findings show that there is a significant difference between the self-adaptation tactic and the baseline regarding the energy consumption of the mobile device. Regarding the time-to-complete, the self-adaptation tactic outperforms the baseline in most cases. If the bandwidth capacity is in its highest range, the self-adaptation tactic and the baseline result the same range of time-to-complete for the apps. So, overall the time-to-complete with the baseline configuration is

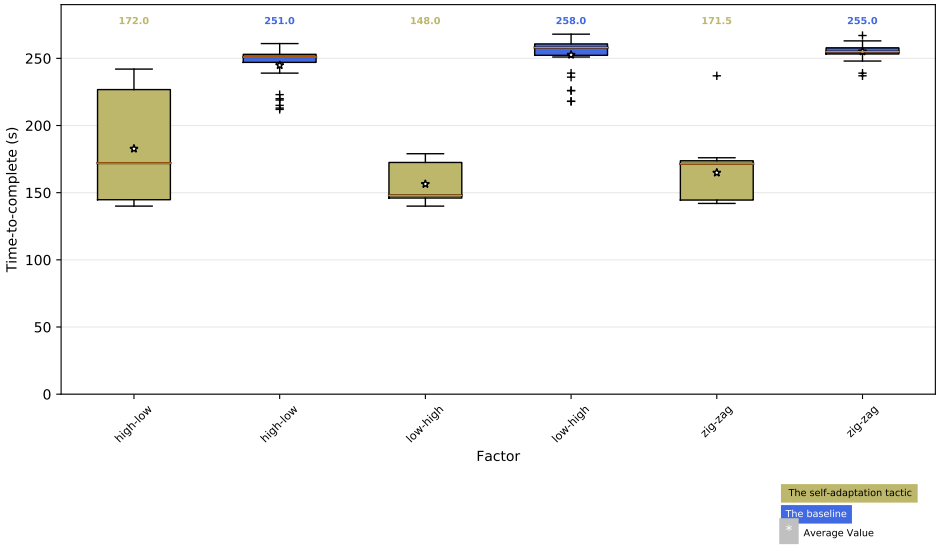


Figure 6.8: The box-plots of the time-to-complete results of the self-adaptation tactic and the base for the applications order factor.

greater than the time-to-complete with the self-adaptation tactic.

Comparing the box-plots of energy consumption and time-to-complete (Figures 6.5, 6.6, 6.7 and 6.8), we observe a correlation between these two variables. These variables show a direct relation, which means decrements in time-to-complete lower the energy consumption amounts as well. However, according to our findings, when the bandwidth capacity is in its highest range, the time-to-complete of the applications is the same for the self-adaptation tactic and the base but the energy consumption of the mobile devices is smaller when employing the self-adaptation tactic. It gives an insight on the different energy states of the network interface in the mobile device. As mentioned before, the energy consumption of the mobile device is composed of the energy consumption of the four energy states of the network interface. Some of the energy states depend on the duration of the transfer and being idle. For some other energy states the energy consumption is independent of the transfer duration. Figure 6.9 shows the box-plots comparing only the transfer energy consumption (*transfer energy state*) for different bandwidth capacities. As it can be seen from the figure, the self-adaptation tactic optimizes the energy consumption of the mobile devices when transferring data for all three types of bandwidth capacities. It is interesting to note that even with high-capacity bandwidth the applications might require to compete over resources. Therefore, the self-adaptation tactic can also

save energy when transferring data.

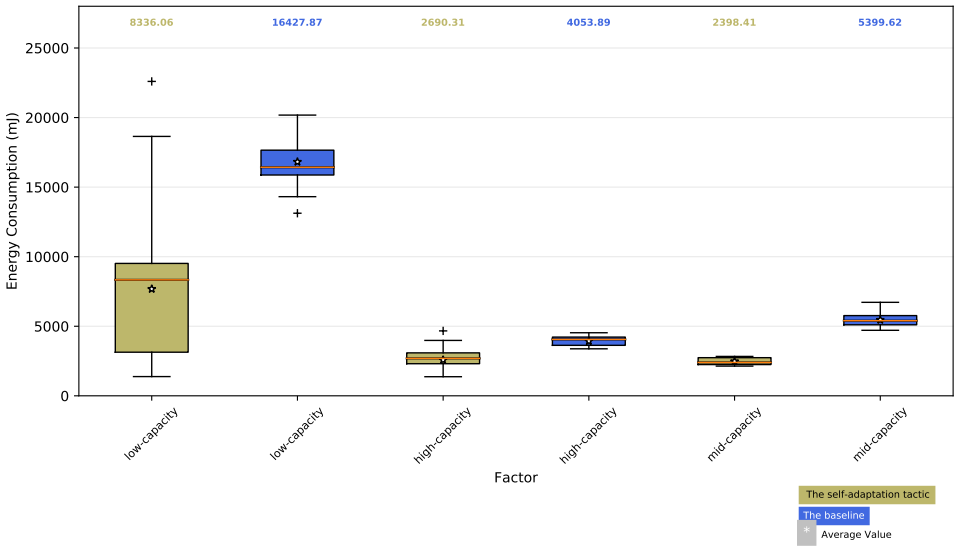


Figure 6.9: The box-plots of the transfer energy consumption results of the self-adaptation tactic and the base for the factor bandwidth capacity.

6.9 Threats to Validity

We identify a number of threats to validity of our work, which we mitigate as described in the following:

Internal validity targets the design and execution of the study. The threats to this validity are defined as the possible errors in the link between the implementation of an experiment and the findings of the experiment. We mitigate this type of threats by following a systematic approach to design our study. First, we use the commonly used mobile apps as the source of data on software reconfiguration in our study. Second, we use well-known scheduling methods to reconfigure the infrastructure utilization in our study.

External validity targets the generalizability of our findings. The threats to this validity are defined as non-representative conclusions that are drawn from too specific implementations of an experiment. We mitigate this type of threats by modelling the utilization behavior of top commonly used mobile apps. In fact, the 6 mobile apps that we select as our subjects, form a representative group of apps that show typical usage scenarios of the mobile devices. Also, to model the

energy consumption of the mobile device, we benefit from the models existing in the peer-reviewed literature.

Conclusion validity targets the statistical significance of findings. The threats to this validity are defined as non-reasonable relations among the collected data set and the variables. We mitigate this type of threats by minimizing the noise when collecting data. When measuring the resource utilization of the selected mobile apps, we stop all the background applications that consume resources.

Construct validity targets the alignment between the objective of the study and the collected data set. The threats to this validity are defined as unexpected noise and randomness when defining and measuring the constructs. We mitigate this type of threats with performing the experimentation in a controlled environment. Also, when simulating the mobile network interfaces we carefully employ the simulation models from the peer-reviewed literature to exclude any unexpected influence on the collected data.

6.10 Related Work

We categorize the related work in the mobile computing field in two groups of solutions: infrastructure reconfiguration and software reconfiguration.

Infrastructure reconfiguration:

There are many simulation frameworks that focus on optimizing the network infrastructure of mobile devices. Some simulate the network interfaces in a generic way, which can be employed for conducting experiments on mobile devices, while others are introduced specifically for mobile devices. For instance, Network Simulator 3 (NS-3)[3] provides additional extensions to evaluate network protocols regarding their energy efficiency. As another generic network simulator, OPNET[4] supports simulation of mobile networks with a focus on the network interfaces [264]. Kari and Mishra extend OPNET to incorporate energy calculations and security measures [134]. OMNET++[5] is another example of general purpose network simulators that is extensible to support mobile networks [78]. MiXiM is a mobile specific simulation framework, which uses the OMNET++ modeling engine as its core[6] [77]. However, the energy consumption values calculated using these tools often show a high-level mapping of the resource utilization and the energy consumption. To find a fine-grained link between the energy consumption of mobile devices and the impact of running mobile applications, we need mobile-specific network simulators, which support particular characteristics of mobile networks. Similitude is a simulation platform that combines Android

[3]https://www.nsnam.org/
[4]https://www.riverbed.com/nl/products/steelcentral/opnet.html
[5]https://omnetpp.org/
[6]MiXiM: http://mixim.sourceforge.net/

emulators and a network traffic simulator [116]. Its aim is to allow large-scale experimentation on efficiency metrics of the mobile apps. Moreover, Chenand *et al.* introduce a system-level simulator for mobile devices [57]. It provides a testbed to compare different mobile network interfaces. However, these frameworks do not offer a way to evaluate self-adaptability of mobile applications for improving energy consumption of mobile devices.

Software reconfiguration:

Most studies in this area focus on solutions to observe the behavior of mobile applications. PowerScope maps the power consumption of the mobile device to the code components of mobile applications and the underlying operating system [94]. PowerSpy splits the energy consumption of the mobile device for each CPU thread [28]. Seo *et al.* introduce energy consumption estimations of Java-based systems in two categories: computation and communication [220]. The aforementioned studies provide insights on energy awareness of mobile applications. However, we are more interested in solutions, in which mobile application can incorporate the energy awareness to adapt their resource utilization. Mizouni *et al.* introduce a framework to design adaptive mobile applications based on feature priorities [178]. Based on their results the mobile applications can save resources using adaptation to contextual changes.

In this study, we introduce a simulation framework to evaluate the impact of self-adaptability of mobile apps on energy efficiency of mobile devices. To do so, we combine the infrastructure reconfiguration and software reconfiguration strategies, which we did not find in the literature. We distinguish between software design approaches and system engineering approaches that can make our findings beneficial to both software architects and system engineers.

6.11 Lessons Learned

The first and the main outcome of our study points out the impact of self-adaptation tactics on energy efficiency. We show that mobile applications can benefit from the self-adaptation tactic to extend the battery life of mobile devices. We realize this goal with two means: software reconfiguration and infrastructure reconfiguration. As the second outcome of our study, we show that self-adaptation tactics combine both software-architectural and system-engineering concepts to realize self-adaptability. Software architects and software engineers can choose where in the process of software design/development is the self-adaptation tactic realized.

Regarding infrastructure reconfiguration, scheduling algorithms can be employed to optimize the utilization of available resources. According to the MAPE functionalities, the scheduling algorithms must monitor the infrastructure and plan the best fitting reconfiguration. Regarding the software reconfiguration,

software architects should design runtime-configurable mobile applications. This means the apps will be able to choose between their own features based on the available resources. For example, if the resources are limited, the apps can opt for disabling some of their features that do not influence the target functionality at that point in time.

Finally, our experimental design combines both empirical and simulation types of experimentation. The design of our framework can be re-utilized by other researchers in the field for energy efficiency studies. Our framework is designed in a modular way that can meet other applications requirements (e.g. delay-sensitiveness that will make them a priority for our scheduling algorithm) and infrastructure requirements (e.g. signal strength of the network interfaces that can vary from weak to strong for each interface). Additionally, the tools and the statistical tests we used represent a systematic methodology that can be employed for other similar experimental studies.

6.12 Conclusion

The increasing demand for mobile applications to transfer data imposes new challenges on the energy consumption of mobile devices, which have a limited battery life. Self-adaptability of mobile applications has become an important quality attribute to overcome resource changes at runtime. This chapter presents an evaluation of the self-adaptation tactic for mobile applications as an answer to our research question (RQ3.1), namely "How can we evaluate the effectiveness of the self-adaptation architectural tactics in different usage contexts?". We perform a systematic experimentation that we design based on the framework proposed by Basili *et al.* [31]. We implement a simulation framework that allows researchers to study the impact of self-adaptability of mobile apps on the energy consumption of mobile devices. To use realistic data sets as input for our framework, we empirically measure the resource consumption of the 6 most commonly used mobile apps in a mobile device. Our framework implements two types of runtime reconfigurations with a focus on the MAPE model functionalities. For software reconfiguration, mobile apps can dynamically adjust their available features and functionalities. For infrastructure reconfiguration, the framework re-schedules the available network resources to adapt to runtime changes. This work is a first step toward emphasizing self-adaptability approaches on the both types of reconfigurations in the mobile computing field.

Our results show that the self-adaptation tactic significantly improves the energy efficiency of mobile apps. Furthermore, in most cases the mobile apps take shorter time to transfer their requests. Our findings help software architects and software engineers to determine feature priorities for mobile applications at design-time. Therefore, at runtime, the mobile apps can make trade-offs between

availability of features when resources are limited. In our future work, we will quantitatively evaluate feature selection of the mobile apps at runtime that can maximize the self-adaptability quality.

7

A Domain Model for Self-Adaptive Software Systems

This chapter answers RQ3.2 with a focus on domain models for self-adaptive software systems. There are already a number of modeling frameworks that have been introduced to realize self-adaptive software systems. However, they usually focus either on runtime adaptation or on designing self-adaptability. As a consequence, they do not provide a clear link between architecture-level and system-level concepts. Without this link, we cannot ensure that the realized system will deliver the designed-for self-adaptability. In this chapter, we address this problem by introducing a domain model that encompasses both levels. Our model can be used to facilitate both architecture design (e.g. making better-informed design decisions) and system engineering (e.g. guiding self-adaptation at runtime). We show the application of our model in two case examples from the literature where self-adaptation aims at energy efficiency.

7.1 Introduction

During execution, modern software systems co-exist with all sorts of uncertainties in both the environment and the systems themselves. This demands for them to be self-adaptive in order to operate correctly when runtime changes happen. Intuitively, self-adaptive software systems can first detect a (possible) runtime change and then adjust their own behavior to accommodate it. Enabling self-adaptability, however, is not a trivial task. Software architects must resolve a number of challenges:

- The simultaneous achievement of quality requirements might be difficult because of conflicts among them. For example, improving energy efficiency is not always aligned with improvements on performance [14]. The challenge is to make the optimum trade-off between those quality requirements.

- The adaptation process itself is resource-consuming. Raibulet *el al.* investigate the adaptation overhead from different perspectives, such as cost and adaptation time [204]. Given the fact that most software systems self-adapt for resource optimization, it is required to minimize the adaptation overhead for a seamless adaptation (i.e. transparently autonomous adaptation to runtime changes without external interference [49]).

Existing conceptual models aim to describe the behavior and the characteristics of self-adaptive software systems [19, 98, 188, 189]. However, the association between (design-time) architecture-level concepts and (runtime) system-level concepts remains implicit. If so, software architects can not ensure the realization of self-adaptability. In this work, we address this problem by introducing a domain model for self-adaptive software systems. The term *domain model* refers to a model that specifies a domain (that of self-adapting software) with related concepts and relationships. Our domain model gives software architects better understanding of the relation between the software design and the corresponding system realization. In the model, they can navigate how the influence of their design decisions propagates to the runtime actions. To show the application of our model, we choose two case examples from the literature and explain how the model can describe their self-adaptability. The examples focus on self-adaptation for *energy efficiency*, which is establishing itself as a critical objective for software systems [44].

This chapter is organized as follows: In Section 7.2 we introduce our domain model for both runtime and design-time concepts. We indicate the mapping between the concepts and the MAPE functionalities. Section 7.3 describes two example case studies that instantiate our model, one from cyber-foraging and one from cloud-computing. In Section 7.4 we discuss our arguments regarding the effectiveness of our model. Existing related work is presented in Section 7.5. We close the chapter with Section 7.6, which includes our conclusions and future directions.

7.2 The Domain Model

A domain model is a collection of concepts and relationships for a specific domain. We introduce a domain model to describe self-adaptive software systems. It works as a visual dictionary for software architects to select the necessary mechanisms for self-adaptation. We have created our domain model as largely based on the KISS method [143] applied to the input from domain experts, which has been gathered incrementally for the past two years. For readability, we organize the concepts of the self-adaptation domain into two models[1]: runtime and design-

[1]The complete model can be found at Appendix A.

time.

The MAPE model identified by Brun *et al.* suggests that self-adaptive systems should include four main functionalities: Monitor, Analyze, Plan and Execute [46]. We identify runtime and design-time concepts that are essential for activating each of these functionalities. In the following we present each model by giving a short introduction followed by a more detailed description of the elements corresponding to each MAPE functionality. When necessary, we show the link to the elements by using the *italic* format.

7.2.1 The Runtime Model

In our description, the term "system" refers to the mix of the execution environment and its running software. Figure 7.1 shows the domain model for self-adaptive systems at runtime. It indicates the mapping of the runtime concepts to the MAPE functionalities:

- *Monitor*: The model allows *to Monitor* the system metrics, which are quantifiable indicators of the behavior of software systems at runtime. Monitoring results in the *System Metric Data* by using two techniques, corresponding to runtime activities: *to Measure* with *Meters*, and *to Estimate* with *Estimation Models*.

- *Analyze*: From the *System Metric Data*, the system can detect *Actual Runtime Changes*, which are (emerging) degradations or improvements in the achievement of quality requirements. For example, a high peak in energy consumption can be a potential threat to the system, while the high availability of more efficient energy sources can be considered as a new opportunity to improve system qualities. From the perspective of the software architecture, runtime changes can be either anticipated or unanticipated. Software systems are typically well equipped for "anticipated" runtime changes at design time. Therefore, the *Trend Recognition Algorithm* of the system can use the *Trend Specification* of the metrics to detect anticipated runtime changes.

- *Plan*: For a detected runtime change, the system must find available *Adaptation Architectural Tactics*. Architectural tactics are design decisions that impact the response of the system to quality attributes [32]. Adaptation tactics, in particular, are mechanisms that enable the system to react when runtime changes arise. As the FORMS reference architecture suggests to separate the self-adaptation concerns and the system functionality concerns, adaptation tactics are realized as system-specific mechanisms [252]. We identify two types of adaptation tactics, namely "software architecture reconfiguration" and "infrastructural reconfiguration".

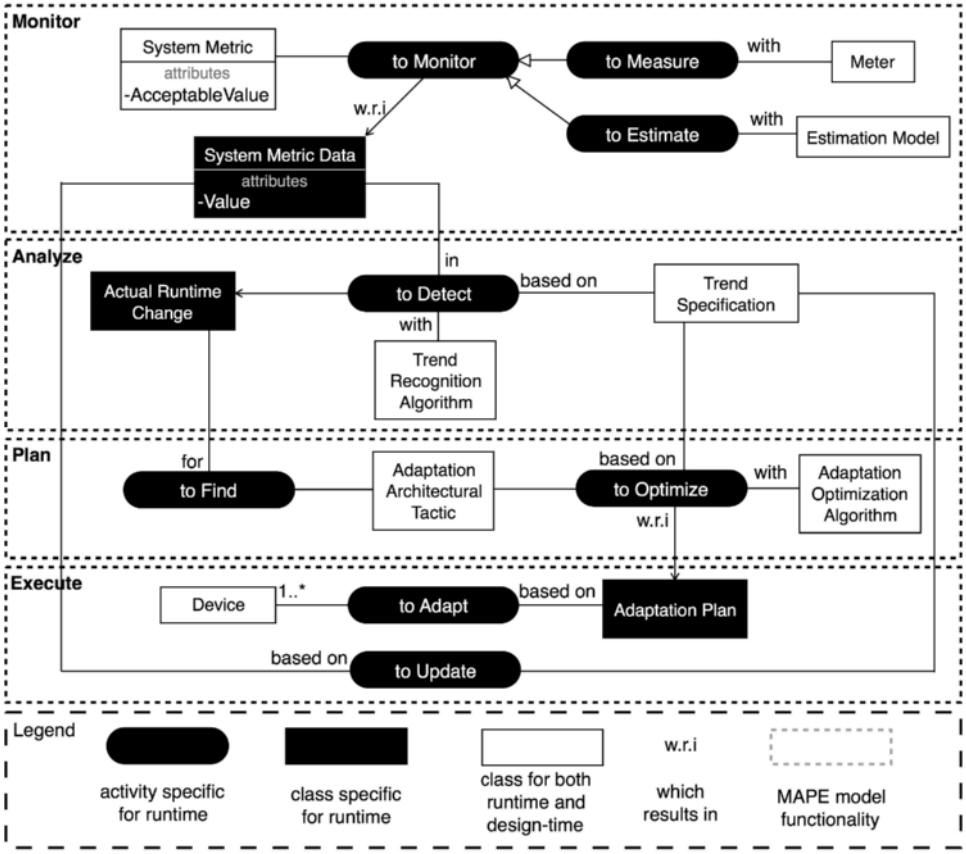


Figure 7.1: The runtime domain model indicating the concepts for self-adaptation at runtime.

An *Adaptation Optimization Algorithm* selects the optimum realization of the tactics, which results in an *Adaptation Plan*. The adaptation process is resource-consuming (e.g. time and energy) itself that must be considered in the adaptation plan. Therefore, the optimization algorithm selects a tactic with the minimum adaptation overhead and the maximum improvement on system qualities.

- *Execute*: The execution environment of the system, which consists of a number of devices, adapts based on the *Adaptation Plan*. If needed the *Trend Specification* is updated according to the *System Metric Data*. As mentioned before, trend specifications specify the expected behavior of the

system in the form of the system metric data. However, in the presence of runtime changes the expectations from the system can change. For instance, if during the execution, the system is configured to supply its energy from a more efficient energy source, its expected energy consumption can become lower, which should be reflected in the trend specifications.

7.2.2 The Design-Time Model

Figure **??** shows the domain model for enabling self-adaptation at design-time. To realize the functionalities of the MAPE model at runtime, a number of activities are performed at design time. In the following we show the relevant concepts for each relevant MAPE functionality:

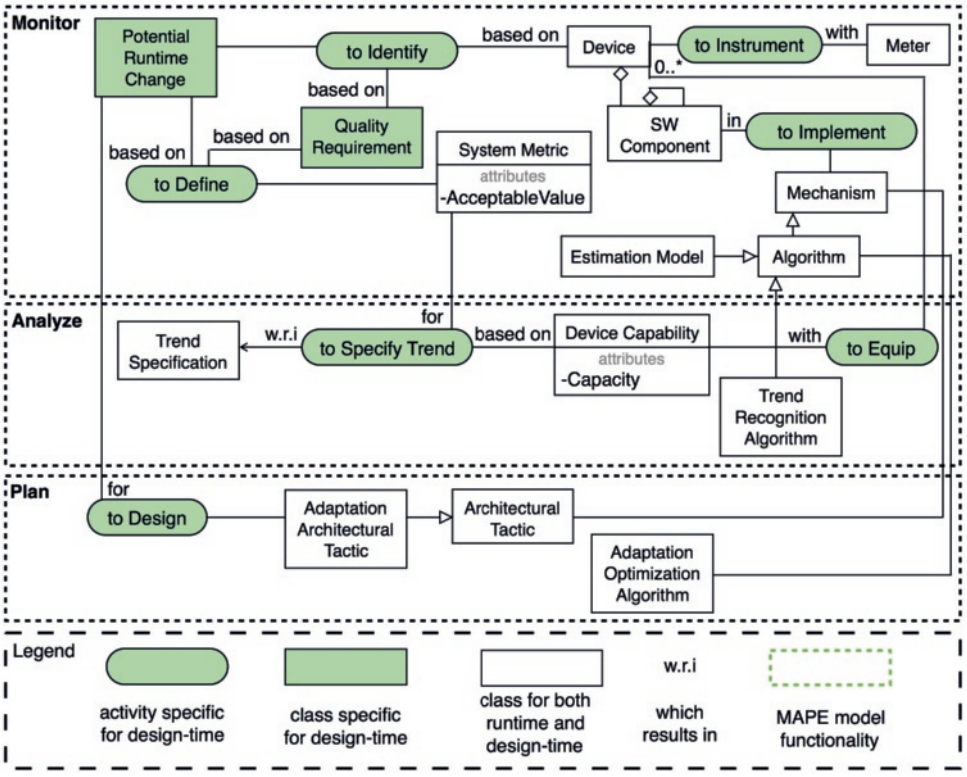


Figure 7.2: The design-time domain model indicating the concepts for enabling self-adaptation at design time.

- *Monitor*: To define what to monitor in the software system, it is necessary for software architects first *to Identify* the *Potential Runtime Changes* based on the *Quality Requirements*. Assessing the potential runtime changes and the quality requirements will lead to the definition of *System Metrics*. To collect the system metric data, the execution environment can either be instrumented with *Meters* or can implement mechanisms like *Estimation Models* that are algorithms to estimate varying/constant values. The metric data reported in device specifications are examples of estimation models with a constant value. Meters can be either physical devices or software components to measure the metrics. For instance, physical energy meters are used to measure energy consumption, while the execution time of a computation task can be measured in a software-based manner.

- *Analyze*: Output of this functionality is the detection of runtime changes. To detect any abnormality in the behavior of the software, one first needs *to Specify Trends* for system metrics based on the available capacity in the execution environment. This results in *Trend Specifications*, which (at runtime) will be used as input by an implemented *Trend Recognition Algorithm*. Trend recognition algorithms range from simple comparison calculations to complex prediction algorithms. In simple cases, upper/lower threshold values are defined for the target system metrics and if the system metric data exceed these values, a runtime change is detected. The cyber-foraging mobile application introduced in [68] can offload its computation tasks to a nearby surrogate for efficiency purposes. However, if the network delay to the surrogate and the mobile battery level exceed the assigned limits, the offloading plan is re-generated. In more complex cases, the response of the system to quality requirements is predicted using pattern prediction algorithms. The objective is to reduce the chance of occurrence for runtime changes. For instance, Hawarah *et al.* use Bayesian networks to predict user behavior in the energy consumption of smart buildings [113].

- *Plan*: According to the identified potential runtime changes, system-specific *Adaptation Architectural Tactics* must be designed. For instance, in cyber-foraging mobile applications, the design decisions regarding "computation offloading to a nearby surrogate" can be realized as an adaptation tactic in reaction to low battery level in mobile devices. They will be realized as reconfigurations of the software architecture and/or the infrastructure of the system. An *Adaptation Optimization Algorithm* will select the adaptation tactics at the time of the runtime changes.

7.3 Case Examples

We illustrate the application of our domain model with the help of two case examples from real world systems, in two different contexts. We explain how the examples are designed for enabling self-adaptation for the purpose of energy efficiency at runtime.

7.3.1 Example 1: Energy Efficient Offloading in Mobile Cyber-Foraging

Cyber-foraging is a technique to extend computing power of mobile devices [155]. We explain how the model can be applied to the case example and help realizing self-adaptability.

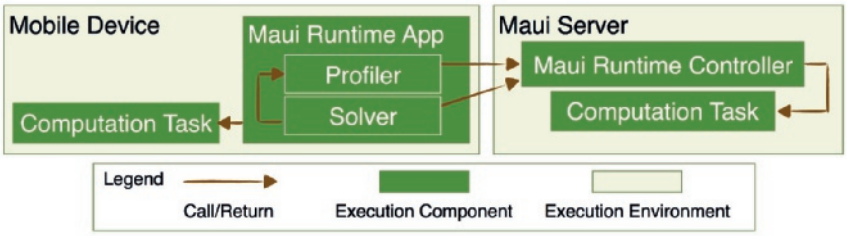


Figure 7.3: The high-level architecture of the Maui cyber-foraging system

The cyber-foraging mobile application can offload its computation tasks to external execution environments in order to save energy in the mobile device. Figure 7.3 shows the high-level architecture of the Maui cyber-foraging system proposed by Cuervo *et al.* [68]. **Maui Runtime App** is the cyber-foraging mobile application that receives the user computation requests. The app consists of two main components, the **Profiler** and the **Solver**. The Profiler monitors the status of the mobile resources and the network connection to the **Maui Server**. The Solver makes use of the collected data to select the best execution environment, which in this example is either the **Mobile Device** itself or the **Maui Server**. The objective is to extend the battery life of the mobile device by delegating the computation to a nearby surrogate, in this case the Maui server.

Figure 7.4 shows the instantiated runtime model for the Maui system. Each activity is marked with a number to show the order of realization steps:

1. Relevant system metrics, i.e. *execution time* of the computation tasks (the difference between the time the app receives the computation request and the time the computation is completed.), *the battery level* and *the network connection delay*, are continuously monitored. The mobile battery level and

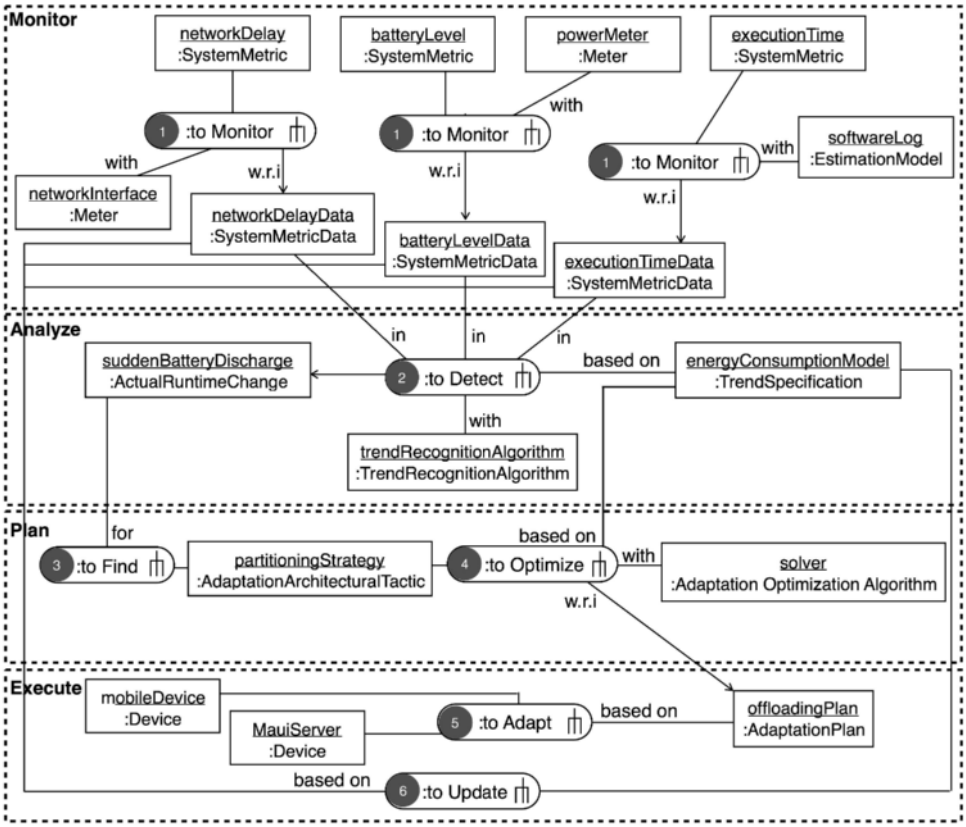


Figure 7.4: The instantiation of the runtime model for the Maui cyber-foraging system

the network delay are measured with the instrumented physical meters. Estimation models are implemented to predict the execution time of the computation tasks when executing locally or remotely.

2. Based on the energy consumption models, *a trend recognition algorithm* detects if a (possible) change occurs, which in this example is the *faster mobile battery discharge*, i.e. a battery discharge that is faster than the predicted trend.

3. Implemented *partitioning strategies* will be assessed for this change.

4. The solver will optimize the strategy by performing a cost-benefit analysis, which is a comparison between the predicted energy consumption of the mo-

bile device when offloading the task and when locally executing. The result of the algorithm is an *offloading plan* that will be configured accordingly in either the mobile device or the Maui server.

5. *The mobile device* and *the Maui server* will adapt based on the adaptation plan. If the plan is to offload the computation, a number of architectural components must be connected, such as the **Maui Runtime Controller** of the Maui server and additional software components in the mobile device, which will transfer the offload-able computation partition to the remote server.

6. If needed, *the energy consumption model* will be updated according to new system metric data.

7.3.2 Example 2: Energy Efficient Data Center Scheduling

We pick the data center example from the work presented by Dong *et al.* [76]. They present a data center scheduler to assign available resources in the infrastructure to the computation tasks.

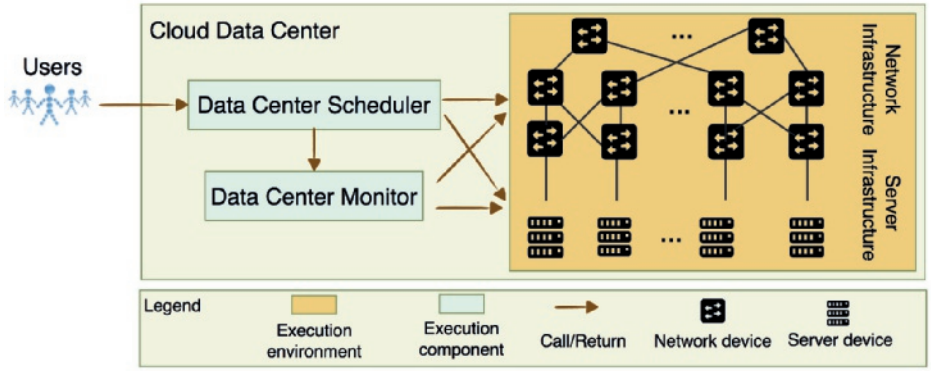


Figure 7.5: The components of a cloud data center that carries out computation tasks in an energy efficient way

As Figure 7.5 presents, the **Data Center Scheduler** receives the requests from the users. The **Data Center Monitor** provides the scheduler with some status data. Then, the scheduler finds an optimum arrangement for utilizing the **Servers** and the **Network Infrastructure**. In this example the objective is to simultaneously maximize the energy efficiency and the performance of the data center. We define energy efficiency as the energy consumed to execute the requested computation tasks.

Figure 7.6 shows the instantiated runtime model for the data center scheduler presented in [76]. We describe each activity in the order of realization:

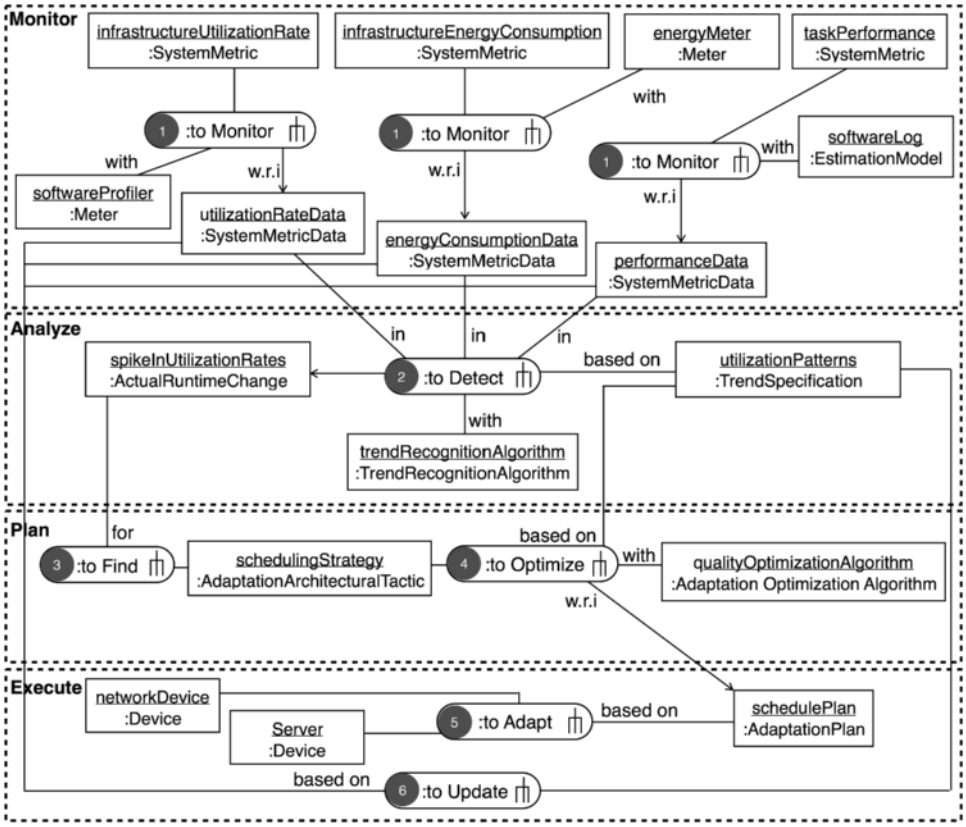


Figure 7.6: The instantiation of the runtime model for the data center scheduler presented in [76]

1. The system metrics are collected from the infrastructure: *the total energy consumption* of the data center environment, *the utilization rate* per device, and *the execution performance* of the computation tasks. Physical meters are instrumented to measure the energy consumption of the infrastructure and software meters are used to estimate the utilization rate of the infrastructure. Estimation models are implemented in the form of software logs to predict the performance to execute the requested tasks.

2. The anticipated changes for this example are rapid changes to resource utilization, which are typically caused by spikes in the computation requests. *A trend recognition algorithm* detects the occurrence of these type of changes based on *the utilization patterns*.

3. Different *scheduling strategies* are evaluated based on the severity of the changes.

4. *A quality optimization algorithm* will optimize a selected scheduling strategy based on utilization patterns. This will result in *a schedule plan* that will be configured in the infrastructure.

5. *The network devices* and *servers* will adapt based on the new schedule plan. Some servers and network devices might reduce their utilization rate, and consequently their energy consumption, by switching to their sleeping mode.

6. If the new adaptation introduces new patterns of resource utilization, *the utilization patterns* will be updated based on newly collected system metric data.

7.4 Discussion

Self-adaptability is a property that naturally manifests itself at runtime. Therefore, and not surprisingly, our example cases focus on how systems react to runtime changes and adapt by re-architecting. It is more challenging to "show" how the domain model supports architects in *designing* self-adaptive software in a more effective manner. To this aim, we discuss our arguments in the following.

We classify the relevant concepts for self-adaptive systems according to their contribution to the software lifecycle: for example *to Identify Potential Runtime Changes* is relevant at design time (drawn in light green in Figure **??**) and *to Monitor System Metrics* is an action occurring at runtime (drawn in black in Figure 7.1). This classification helps software architects recognize the link between "what shall happen" and "what can happen". Further, with the help of the concepts from the model that are relevant for both design time and runtime (drawn in white), the link between software design and system engineering is both explicit and integrated in the fact that some concepts exist throughout the full lifecycle.

In addition, to enable seamless adaptation, one first needs to identify potential runtime changes. Identifying *all* runtime changes at design time is challenging (if possible at all) as the runtime condition of the software systems is not completely known before execution. We argue that our domain model helps better-informed reasoning for the identification of potential runtime changes by offering the potentially-relevant contextual elements. In doing so, it can increase the number of identified changes. This resembles the work of Tang & Lau [239] showing that the reasoning behind the decisions can raise new design issues –

in our case new potential runtime changes. As the relation between design decisions is emphasized in our domain model, the improvement in the reasoning can eventually be extended to other decisions, such as selecting suitable adaptation mechanisms. For example, to achieve self-adaptation, software systems must employ the intelligence for an effective adoption of adaptation tactics. This means available adaptation mechanisms must be selected according to the runtime changes. An implemented optimization algorithm makes different trade-offs between quality requirements to minimize the gap between the current behavior and the expected behavior of the system. In our domain model, the activities realizing the "Plan" MAPE functionality both at runtime and design-time, deliver this trade-off analysis.

In particular, the improvement in reasoning can eventually improve the classification of anticipated and unanticipated changes. Software architects choose which identified potential changes should be anticipated for the system and which potential changes should remain unanticipated. Tools like "probability and impact matrix" are used to perform a risk assessment of each identified change [104]. For instance, one could leave changes with a low impact and with a small probability of occurrence unanticipated. Better-informed design decisions can link better "Potential Runtime Change" (a design-time concept) and "Actual Runtime Change" (a runtime concept) that are presented in our model.

7.5 Related Work

A number of survey studies introduce characteristics of existing self-adaptive solutions [58, 71, 214]. Our focus, however, is on self-adaptability at the architectural level. The essential role of software architecture in self-adaptability is investigated by Oreizy *et al.* [188], which outline an architecture-based approach that contains high-level processes to enable self-adaptation. Their approach describes the two aspects of a self-adaptive system at runtime: evolution management and adaptation management. In a more recent study, they discuss a number of architectural styles that can enable self-adaptation [189]. However, their focus is on the runtime adaptation by re-configuring software architecture. Their model does not include design decisions for enabling such reconfigurations in the form of the MAPE model functionalities.

Andersson *et al.* propose a classification of modeling dimensions for self-adaptive systems [19]. The idea is to gather a common vocabulary for engineers to specify self-$\star$ properties of self-adaptive systems. The Rainbow framework [98] adopts an architecture-based approach with the focus on re-usability. At the same time, it recognizes a number of mechanisms for specializing the infrastructure for the sake of applicability. Their focus is on self-adaptation at runtime with proposing an external architectural layer to monitor and reconfigure the system.

They do not address the design time activities such as change identification and the selection of adaptation mechanisms.

Weyns *et al.* present the FORMS (FOrmal Reference Model for Self-adaptation) reference model that includes specifications of modeling elements and relationships among them in the design of self-adaptive software systems [252]. They focus on separation of self-adaptation concerns (meta-level subsystem) and system functionality concerns (base-level subsystem) at the architectural level. However, FORMS does not support the trace-ability link between the design specification and the system's implementation.

A number of self-adaptation approaches targeting specific domain of wireless sensor networks (WSNs) exist. For instance, Ruiz *et al.* propose MANNA, which is a decentralized policy-based management architecture [209]. Based on policies, MANNA achieves the desired behavior of the system, while it minimizes the generated traffic by the management layer. Agilla is a middleware for WSNs that implements a mobile multi-agent approach to support self-adaptation [97]. Portocarrero *et al.* introduce a reference architecture for self-adaptive service-oriented WSNs [197]. These studies place their focus on specific runtime changes (e.g. undesirable energy level in sensors) and specific adaptation tactics (e.g. replacing a low energy node). In comparison, in our model, we emphasize the design-time activities for "change identification", "system metric identification", and "adaptation tactic selection", which potentially help to make better-informed design decisions that realize self-adaptability at runtime. We provide the fine-grained explicit link between the design decisions and the runtime behavior of the system. We present the design-time concepts and the runtime concepts in the form of MAPE model functionalities that enable seamless adaptation.

7.6 Conclusion

This chapter presents a domain model for self-adaptive software systems as an answer to our research question (RQ3.2), namely "Which architectural tactics can ensure energy efficiency of software systems?". The design of self-adaptive software systems is supported by introducing the relevant concepts at design time and runtime. Software architects can use our model to enable seamless adaptation that corresponds to the MAPE model functionalities.

To qualitatively illustrate our model, we used two examples from the literature that enable self-adaptation for the purpose of energy efficiency. We described how these example systems can benefit from our model to ensure their energy efficiency. In future work we plan to quantitatively evaluate self-adaptive systems designed as based on our domain model.

Although seamless adaptation is only evident at runtime, we argue that software architects can ensure self-adaptability with the help of our model by employ-

ing the runtime concepts of the model as elements of their architecture design. Future work will focus on evaluating in practice the power of our domain model for effective design decision making.

8 Conclusion

Software systems cause large amounts of energy consumption in ICT. Energy efficiency has become an essential quality attribute to minimize the impact of software on energy resources.

Currently, we realize that software systems are designed somewhat independent of the configuration of their underlying execution context. The dynamic contextual changes at runtime can influence the availability of resources. Therefore, software systems need to compete with each other over resources. With resource limitations, achieving higher efficiency of software quality requirements becomes more challenging. It is not a trivial task to maximize efficiency of software qualities simultaneously because of their possible conflicting nature. For instance, maximizing availability of a system can require duplicating hardware devices. This, in turn, can negatively influence the energy efficiency of the system.

In this thesis, we focus on self-adaptation architectural tactics for the purpose of energy efficiency. With self-adaptation tactics, software systems will be alert to runtime changes and recover from them. Throughout the chapters of the thesis, we distinguish between architectural and infrastructural solutions. In this concluding chapter, we answer our research questions with a short list of take-away messages. At last, we provide a summary of our research plans for future work.

8.1 Answers to Research Questions

The aim of this thesis is to investigate ways that enable self-adaptability of software systems for the purpose of energy efficiency. Our main research question is formulated in Chapter 1 as:

RQ: How can we improve energy efficiency of software systems by enabling self-adaptation?

To answer this question, we have defined a number of sub-questions that cover several aspects of our main research question more specifically:

RQ1: What are the emerging approaches in the field of energy-efficient self-adaptation?

In order to define the relationship between energy efficiency and self-adaptability, we systematically surveyed the state-of-the-art to answer RQ1. In Chapter 2, we carried out a systematic literature review to collect existing relevant studies. We generalized our findings in two types of self-adaptability approaches, namely software design approaches (SDA) and system engineering approaches (SEA). Our findings reveal patterns on adoption of the self-adaptation approaches in different years, application domains and systems types. The answer to RQ1 is a guideline to help solution providers choose the best-fitting self-adaptation approaches based on specific requirements.

RQ2: How can software-defined infrastructures help increasing energy efficiency of software?

After identifying the latest updates in the field, we opted to analyze the impact of infrastructure-specific solutions on the energy efficiency of running software systems. We specifically focused on software-defined infrastructure, which can be able to realize self-adaptability approaches. In Chapter 3, we surveyed energy efficiency solutions adopted for networking components in a systematic way to answer RQ2.1 namely, *What are energy efficient solutions for networking in the cloud?*. Our results are grouped into four types of solutions: devices, network architectures, routing/switching protocols, and decision frameworks. Among all the types, decision frameworks have attracted the most attention from the researchers. Decision frameworks have been the most promising regarding energy savings in the networks. It is because decision frameworks are in fact self-adaptive software systems that can employ optimization algorithms to utilize the underlying infrastructure in a very efficient way. The importance of optimization algorithms leads us to our next sub-question RQ2.2, namely *How can optimization algorithms utilize infrastructure in an energy efficient way?* In Chapter 4, we develop and evaluate an optimization algorithm for software-defined networks. Our simulation findings show that using different variations of our algorithm, we can achieve 15% to 45% power savings out of the maximum possible savings.

RQ3: Can we guide architectural tactics to ensure energy efficiency of software systems?

The previous research questions focused on the impact of adaptation of infrastructure on the energy efficiency. We complement our previous results with RQ3, which targets the impact of software design on energy efficiency. Therefore, we formulate RQ3.1 as *How can we evaluate the effectiveness of the self-adaptation architectural tactics at different usage contexts?* In Chapter 5, we empirically evaluate cyber-foraging architectural tactics and in Chapter 6, we present a sim-

ulation framework to enable self-adaptability of mobile applications. In both chapters we realize that self-adaptive solutions increase energy efficiency while other quality attributes such as performance is not impacted negatively. Finally, in Chapter 7, we address RQ3.2, namely *Which architectural tactics can ensure energy efficiency of software systems?* We present a domain model for self-adaptive software systems. The model encompasses both software architecture and system engineering concepts.

8.2 Lessons Learned

Addressing our main research question has lead us to a number of take away messages. Software engineers and software architects can benefit from the lessons listed below, when building energy efficient self-adaptive software systems. As highlighted in previous chapters, we define the self-adaptability as implementations of the MAPE functionalities. The MAPE functionalities can ensure and increase energy efficiency. The following points list the lessons learned regarding each of these functionalities:

1. (Monitor) *Find the right granularity level to measure the metrics at runtime.* Monitoring every measurable variable will add an undeniable overhead on performance of software systems. It is important to equip the systems with the minimum number of meters and profilers. Also, in some cases monitoring some metrics can help get insights on other related metrics. Identifying and analyzing the metrics at design time helps finding the right granularity level for monitoring software systems.

2. (Analyze) *Tune the observed patterns of resource consumption at runtime.* Self-adaptive software systems reconfigure themselves when runtime contextual changes happen. Reconfiguration can result in new patterns of resource consumption, which should be taken into account when detecting future contextual changes. Pattern recognition algorithms help identify the changes to the existing patterns at runtime.

3. (Plan) *Optimization algorithms should be optimized, too.* Optimization algorithms perform the most effective when they are adjusted based on the requirements of software systems and the configurations of underlying infrastructure. To benefit the most from optimization algorithms, one could employ some variations of optimization algorithms that can be realized in the best fitting situations.

4. (Execute) *Adaptation plans consume energy themselves.* Adaptation possibilities should be investigated at design time regarding their overhead at

runtime. In general, a sound approach to enable self-adaptability for software systems, is to categorize the functional features into multiple groups of resource utilization. Therefore, when runtime changes occur, software systems can switch between these groups that can result in disabling some features and enabling some others. Switching from one features group to another can be resource-consuming itself, which should be taken into account when employing the optimization algorithms.

8.3 The Road Ahead

Our work can be extended in at least three future research directions:

1. *Designing and performing empirical experimentation to quantify the impact of tactics for self-adaptation in different usage contexts.* Software systems of different domains may introduce new challenges and requirements to realize self-adaptability. This direction will help software architects have statistical evidence to support their design decisions.

2. *Providing variations of optimization algorithms.* Optimization algorithms should be able to adjust themselves according to contextual changes occurring at runtime. We plan to design and develop different optimization algorithms with differing priorities such as high scalability, low computation costs, high accuracy, etc. This will provide a catalog of algorithms for software engineers to pick the best-fitting algorithm for software systems based on their requirements.

3. *Studying the effectiveness of our domain model in case studies.* We quantify the energy efficiency of new software systems that are built according to our domain model. The results will help software architects sort the elements of self-adaptability from the most- to the least-influencing with quantified evidence, and make well-informed design decisions.

Bibliography

[1] http://www.nationmaster.com/country-info/stats/Energy/Electricity/Consumption#-amount. [Online; accessed 26-Feb-2019]. (Cited on page 1.)

[2] Number of mobile app downloads worldwide from 2009 to 2017 (in millions). http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/, September 2013. [Online; accessed 26-Feb-2019]. (Cited on page 83.)

[3] Cisco catalyst 2960-x, 2960-cx, and 3560-cx platforms: The greenest catalyst switches ever. 2015. [Online; accessed 26-Feb-2019]. (Cited on page 79.)

[4] 3GPP. Lte overview, 2017. [Online; accessed 26-Feb-2019]. URL: http://www.3gpp.org/technologies/keywords-acronyms/98-lte. (Cited on page 116.)

[5] Mohamed Abdelaal, Oliver Theel, Christian Kuka, Peilin Zhang, Yang Gao, Vasilisa Bashlovkina, Daniela Nicklas, and Martin Fränzle. Improving energy efficiency in qos-constrained wireless sensor networks. *International Journal of Distributed Sensor Networks*, 12(5):1576038, 2016. (Cited on pages 18, 19, and 20.)

[6] Saeid Abolfazli, Zohreh Sanaei, Ejaz Ahmed, Abdullah Gani, and Rajkumar Buyya. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials*, 16(1):337–368, 2014. (Cited on page 111.)

[7] Mohammed Abufouda. A framework for enhancing performance and handling run-time uncertainty in self-adaptive systems. *arXiv preprint arXiv:1402.2144*, 2014. (Cited on pages 18, 19, and 20.)

[8] Emmanuel Olufemi Urai Adeagbo. *Energy-Efficient Pattern Matching Methods on a Fine-Grained Many-Core Platform*. University of California, Davis, 2017. (Cited on pages 18 and 20.)

[9] Ashish Agrawal and TV Prabhakar. Towards a framework for building adaptive app-based web applications using dynamic appification. In *European Conference on Software Architecture*, pages 37–44. Springer, 2015. (Cited on page 111.)

[10] Hisham Ahmed and Othman Sidek. An energy-aware self-adaptive system-on-chip architecture for real-time harris corner detection with multi-resolution support. *Microprocessors and Microsystems*, 49:164–178, 2017. (Cited on pages 18 and 20.)

[11] Yousif EE Ahmed, Kondo H Adjallah, Sharef F Babikier, and Romuald Stock. Resiliency assessment of ndsc based lifetime maximization approach for heterogeneous wireless sensor network by monte carlo simulation. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on*, volume 1, pages 374–378. IEEE, 2017. (Cited on pages 18 and 20.)

[12] Baris Aksanli, Tajana Simunic Rosing, and Inder Monga. Benefits of green energy and proportionality in high speed wide area networks connecting data centers. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 175–180. EDA Consortium, 2012. (Cited on pages 37, 40, and 56.)

[13] Fahimeh Alizadeh Moghaddam, Robert Deckers, Giuseppe Procaccianti, Paola Grosso, and Patricia Lago. A domain model for self-adaptive software systems. In *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings*, pages 16–22. ACM, 2017. (Cited on pages 18 and 20.)

[14] Fahimeh Alizadeh Moghaddam and Paola Grosso. Linear programming approaches for power savings in software-defined networks. In *NetSoft Conference and Workshops*, pages 83–87. IEEE, 2016. (Cited on page 141.)

[15] Fahimeh Alizadeh Moghaddam, Patricia Lago, and Paola Grosso. Energy-efficient networking solutions in cloud-based environments: A systematic literature review. *ACM Computing Surveys (CSUR)*, 47(4):64, 2015. (Cited on page 21.)

[16] Fahimeh Alizadeh Moghaddam, Giuseppe Procaccianti, Grace A Lewis, and Patricia Lago. Empirical validation of cyber-foraging architectural tactics for surrogate provisioning. *Journal of Systems and Software*, 2017. (Cited on pages 18 and 20.)

[17] Raquel Almeida and Marco Vieira. Benchmarking the resilience of self-adaptive software systems: Perspectives and challenges. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, pages 190–195, New York, NY, USA, 2011. ACM. (Cited on page 92.)

[18] Ibrahim Alzamil, Karim Djemame, Django Armstrong, and Richard Kavanagh. Energy-aware profiling for cloud computing environments. *Electronic Notes in Theoretical Computer Science*, 318:91–108, 2015. (Cited on pages 18, 19, and 20.)

[19] Jesper Andersson, Rogerio De Lemos, Sam Malek, and Danny Weyns. Modeling dimensions of self-adaptive software systems. In *Software engineering for self-adaptive systems*, pages 27–47. Springer, 2009. (Cited on pages 142 and 152.)

[20] Django Armstrong, Karim Djemame, and Richard Kavanagh. Towards energy aware cloud computing application construction. *Journal of Cloud Computing*, 6(1):14, 2017. (Cited on pages 18 and 20.)

[21] AMEDEO Asnaghi, M Ferroni, and MD Santambrogio. Dockercap: A software-level power capping orchestrator for docker containers. In *Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), 2016 IEEE Intl Conference on*, pages 90–97. IEEE, 2016. (Cited on pages 18, 19, and 20.)

[22] Uwe Aßmann, Sebastian Götz, Jean-Marc Jézéquel, Brice Morin, and Mario Trapp. A reference architecture and roadmap for models@ run. time systems. In *Models@ run. time*, pages 1–18. Springer, 2014. (Cited on pages 18 and 20.)

[23] Fouad Bahrpeyma, Hassan Haghighi, and Ali Zakerolhosseini. An adaptive rl based approach for dynamic resource provisioning in cloud virtualized data centers. *Computing*, 97(12):1209–1234, 2015. (Cited on pages 18, 19, and 20.)

[24] Rajesh Krishna Balan, Darren Gergle, Mahadev Satyanarayanan, and James Herbsleb. Simplifying cyber foraging for mobile devices. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 272–285. ACM, 2007. (Cited on page 110.)

[25] Rajesh Krishna Balan, Mahadev Satyanarayanan, So Young Park, and Tadashi Okoshi. Tactics-based remote execution for mobile computing. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 273–286. ACM, 2003. (Cited on page 110.)

[26] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM, 2009. (Cited on page 114.)

[27] Jayant Baliga, Robert WA Ayre, Kerry Hinton, and RodneyS Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167, 2011. (Cited on page 24.)

[28] Kutty S Banerjee and Emmanuel Agu. Powerspy: fine-grained software energy profiling for mobile devices. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 2, pages 1136–1141. IEEE, 2005. (Cited on page 137.)

[29] Davide Basilio Bartolini. An autonomic operating system via applications monitoring and performance aware scheduling. 2011. (Cited on pages 18 and 20.)

[30] Victor R. Basili. Software modeling and measurement: The goal/question/metric paradigm. Technical report, University of Maryland at College Park, College Park, MD, USA, 1992. (Cited on page 91.)

[31] Victor R Basili, Richard W Selby, and David H Hutchens. Experimentation in software engineering. *IEEE Transactions on software engineering*, (7):733–743, 1986. (Cited on pages 6, 90, 112, 119, and 138.)

[32] Len Bass. *Software architecture in practice*. Pearson Education India, 2007. (Cited on page 143.)

[33] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 3rd edition, 2012. (Cited on pages 85 and 86.)

[34] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya, et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111, 2011. (Cited on page 25.)

[35] Nelly Bencomo, Paul Grace, and Peter Sawyer. Revisiting the relationship between software architecture and requirements: the case of dynamically adaptive systems. In *Self-Organizing Architectures SOAR 2009 Workshop at Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), WICSA/ECSA*, 2009. (Cited on pages 17, 18, and 20.)

[36] Elhadj Benkhelifa, Tom Welsh, Loai Tawalbeh, Yaser Jararweh, and Anas Basalamah. Energy optimisation for mobile device power consumption: A survey and a unified view of modelling for a comprehensive network simulation. *Mobile Networks and Applications*, 21(4):575–588, 2016. (Cited on page 114.)

[37] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010. (Cited on page 76.)

[38] Andreas Christoph Bergen. *Energy adaptive digital ecosystems*. PhD thesis, 2017. (Cited on pages 18 and 20.)

[39] Kamlesh Bhatia. Hype cycle for the telecommunications industry, 2014. *Gartner Research, Retrieved from Gartner database*, 2014. (Cited on pages 61 and 62.)

[40] Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and J Rougier. A survey of green networking research. *Communications Surveys & Tutorials, IEEE*, 14(1):3–20, 2012. (Cited on page 25.)

[41] Chris Bihary. The sdn story archype is (slowly) becoming reality. January 2017. [Online; accessed 28-November-2017]. (Cited on page 66.)

[42] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys & Tutorials, IEEE*, 13(2):223–244, 2011. (Cited on page 25.)

[43] Danyl Bosomworth. Mobile marketing statistics 2015. *Smart Insights site*, 2013. (Cited on page 83.)

[44] David J Brown and Charles Reams. Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58, 2010. (Cited on page 142.)

[45] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*, 2008. (Cited on page 2.)

[46] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer, 2009. (Cited on pages 4, 10, 114, and 143.)

[47] Jens Buysse, Konstantinos Georgakilas, Anna Tzanakaki, Marc De Leenheer, Bart Dhoedt, Chris Develder, and Piet Demeester. Calculating the minimum bounds of energy consumption for cloud networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7. IEEE, 2011. (Cited on pages 34, 40, 47, 56, and 66.)

[48] Radu Calinescu, Danny Weyns, Simos Gerasimou, Muhammad Usman Iftikhar, Ibrahim Habli, and Tim Kelly. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering*, 2017. (Cited on pages 18 and 20.)

[49] Chun Cao, Ping Yu, Hao Hu, and Jian Lv. Toward a seamless adaptation platform for internetware. *Science China Information Sciences*, 56(8):1–13, 2013. (Cited on page 142.)

[50] C Cappiello, M Fugini, B Pernici, and P Plebani. Green information systems for sustainable it. In *Information Technology and Innovation Trends in Organizations*, pages 153–160. Springer, 2011. (Cited on pages 18, 19, and 20.)

[51] Alessandro Carrega, Suresh Singh, Roberto Bruschi, and Raffaele Bolla. Traffic merging for energy-efficient datacenter networks. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS*, volume 12, pages 8–11. (Cited on pages 34, 36, 40, 44, 50, 51, and 56.)

[52] Derya Cavdar and Fatih Alagoz. A survey of research on greening data centers. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 3237–3242. IEEE, 2012. (Cited on page 25.)

[53] Isabella Cerutti, Pier Giorgio Raponi, Nicola Andriolli, Piero Castoldi, and Odile Liboiron-Ladouceur. Designing energy-efficient data center networks using space-time optical interconnection architectures. *Selected Topics in Quantum Electronics, IEEE Journal of*, 19(2):3700209–3700209, 2013. (Cited on pages 35, 40, 42, 44, 50, 51, 52, and 56.)

[54] Dave Chaffey. Mobile Marketing Statistics compilation. `http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/`, 2017. [Online; accessed 16-May-2017]. (Cited on page 1.)

[55] Yu-Shuo Chang and Shih-Hao Hung. Developing collaborative applications with mobile cloud-a case study of speech recognition. *J. Internet Serv. Inf. Secur.*, 1(1):18–36, 2011. (Cited on page 88.)

[56] Luxi Chen, Linpeng Huang, Chen Li, and Xiwen Wu. Self-adaptive architecture evolution with model checking: A software cybernetics approach. *Journal of Systems and Software*, 124:228–246, 2017. (Cited on pages 18 and 20.)

[57] Li Chenand, Wenwen Chen, Bin Wang, Xin Zhang, Hongyang Chen, and Dacheng Yang. System-level simulation methodology and platform for mobile cellular systems. *IEEE Communications Magazine*, 49(7), 2011. (Cited on pages 114 and 137.)

[58] Betty HC Cheng, Rogerio De Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer, 2009. (Cited on page 152.)

[59] Byung-Gon Chun and Petros Maniatis. Augmented smartphone applications through clone cloud execution. In *HotOS*, volume 9, pages 8–11, 2009. (Cited on page 110.)

[60] Tudor Cioara, Ionut Anghel, and Ioan Salomie. Methodology for energy aware adaptive management of virtualized data centers. *Energy Efficiency*, 10(2):475–498, 2017. (Cited on pages 16, 18, 19, and 20.)

[61] EC (European Commission). Energy efficiency plan 2011, 2011. URL: `https://ec.europa.eu/clima/sites/clima/files/strategies/2050/docs/efficiency_plan_en.pdf`. (Cited on page 9.)

[62] EINS Consortium et al. Overview of ict energy consumption (d8. 1). *Report FP7-2888021. European Network of Excellence in Internet Science*, 2013. (Cited on page 9.)

[63] T D Cook and D T Campbell. *Quasi-experimentation: Design & analysis issues for field settings*. Houghton Mifflin Company, Boston, 1979. (Cited on page 108.)

[64] Peter Corcoran and Anders Andrae. Emerging trends in electricity consumption for consumer ict. *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep*, 2013. (Cited on pages 1 and 2.)

[65] Nicola Cordeschi, Mohammad Shojafar, and Enzo Baccarelli. Energy-saving self-configuring networked data centers. *Computer Networks*, 57(17):3479–3491, 2013. (Cited on pages 35, 40, and 56.)

[66] Vlad Coroama and Lorenz M Hilty. Energy consumed vs. energy saved by ict–a closer look. In *Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools, 23rd International Conference on Informatics for Environmental Protection, Berlin*, pages 353–361, 2009. (Cited on page 1.)

[67] Javier Mendonca Costa and Guowang Miao. Context-aware machine-to-machine communications. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 730–735. IEEE, 2014. (Cited on pages 18, 19, and 20.)

[68] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *International Conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010. (Cited on pages 110, 146, and 147.)

[69] Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. Self-adaptive battery and context aware mobile application development. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 761–766. IEEE, 2014. (Cited on pages 18, 19, and 20.)

[70] Silvana de Gyvés Avila and Karim Djemame. Fuzzy logic based qos optimization mechanism for service composition. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 182–191. IEEE, 2013. (Cited on pages 18, 19, and 20.)

[71] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013. (Cited on page 152.)

[72] Frederico Alvares de Oliveira Junior. *Multi Autonomic Management for Optimizing Energy Consumption in Cloud Infrastructures*. PhD thesis, Université de Nantes, 2013. (Cited on pages 18, 19, and 20.)

[73] Jean-Christophe Deprez, Ravi Ramdoyal, and Christophe Ponsard. Integrating energy and eco-aware requirements engineering in the development of services-based applications on virtual clouds. In *First International Workshop on Requirements Engineering for Sustainable Systems*, 2012. (Cited on pages 18, 19, and 20.)

[74] Karim Djemame, Django Armstrong, Richard Kavanagh, Ana Juan Ferrer, David Garcia Perez, David Antona, Jean-Christophe Deprez, Christophe Ponsard, David Ortiz, Mario Macías Lloret, et al. Energy efficiency embedded service lifecycle: Towards an energy efficient cloud computing architecture. In *Joint Workshop Proceedings of the 2nd International Conference on ICT for Sustainability 2014*, pages 1–6. CEUR-WS. org, 2014. (Cited on pages 18, 19, and 20.)

[75] Karim Djemame, Raimon Bosch, Richard Kavanagh, Pol Alvarez, Jorge Ejarque, Jordi Guitart, and Lorenzo Blasi. Paas-iaas inter-layer adaptation in an energy-aware cloud environment. *IEEE Transactions on Sustainable Computing*, 2(2):127–139, 2017. (Cited on pages 18 and 20.)

[76] Jiankang Dong, Xing Jin, Hongbo Wang, Yangyang Li, Peng Zhang, and Shiduan Cheng. Energy-saving virtual machine placement in cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 618–624. IEEE, 2013. (Cited on pages 34, 36, 40, 47, 51, 56, 149, and 150.)

[77] Martin Dräxler, Frederic Beister, Stephan Kruska, Jörg Aelken, and Holger Karl. Using omnet++ for energy optimization simulations in mobile core networks. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 157–165. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012. (Cited on pages 114 and 136.)

[78] Witold Drytkiewicz, Steffen Sroka, Vlado Handziski, Andreas Köpke, and Holger Karl. A mobility framework for omnet++. In *3rd International OMNeT++ workshop*, volume 93, 2003. (Cited on page 136.)

[79] Daniele Joseph Dubois. Self-organizing methods and models for software development. 2010. (Cited on pages 18, 19, and 20.)

[80] Christopher Eibel and Tobias Distler. Towards energy-proportional state-machine replication. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware*, page 4. ACM, 2015. (Cited on pages 18, 19, and 20.)

[81] EN Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-efficient server clusters. In *International Workshop on Power-Aware Computer Systems*, pages 179–197. Springer, 2002. (Cited on page 3.)

[82] Eric Enge. Mobile vs desktop usage in 2018: Mobile takes the lead, 2018. [Online; accessed 26-Feb-2019]. URL: `https://www.stonetemple.com/mobile-vs-desktop-usage-study/`. (Cited on page 115.)

[83] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. Taming uncertainty in self-adaptive software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 234–244. ACM, 2011. (Cited on pages 18, 19, and 20.)

[84] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, and Naixue Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow

routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196, 2013. (Cited on pages 34, 40, 51, 52, 56, and 67.)

[85] Weiwei Fang, Xiangmin Liang, Yantao Sun, and Athanasios V Vasilakos. Network element scheduling for achieving energy-aware data center networks. *International Journal of Computers, Communications & Control*, 7(2), 2012. (Cited on pages 34, 35, 40, 46, 47, 54, and 56.)

[86] Weiwei Fang, Xiangmin Liang, Yantao Sun, and Athanasios V Vasilakos. Network element scheduling for achieving energy-aware data center networks. *International Journal of Computers Communications & Control*, 7(2):241–251, 2014. (Cited on page 67.)

[87] Mohammad Abdullah al Faruque. *Runtime Adaptive System-on-Chip Communication Architecture*. PhD thesis, Karlsruhe, Univ., Diss., 2009, 2009. (Cited on pages 18, 19, and 20.)

[88] Michael Feilen. *CMR12*. PhD thesis, Universität München, 2017. (Cited on pages 18 and 20.)

[89] Alexandre Mello Ferreira and Barbara Pernici. Using intelligent agents to discover energy saving opportunities within data centers. In *RE4SuSy@ RE*, 2013. (Cited on pages 18, 19, and 20.)

[90] Alexandre Mello Ferreira and Barbara Pernici. Managing the complex data center environment: an integrated energy-aware framework. *Computing*, 98(7):709–749, 2016. (Cited on pages 18, 19, and 20.)

[91] Antonio Filieri, Henry Hoffmann, and Martina Maggio. Automated design of self-adaptive software with control-theoretical formal guarantees. In *Proceedings of the 36th International Conference on Software Engineering*, pages 299–310. ACM, 2014. (Cited on pages 18, 19, and 20.)

[92] Jason Flinn. Cyber foraging: Bridging mobile and cloud computing. In Mahadev Satyanarayanan, editor, *Synthesis Lectures on Mobile and Pervasive Computing*. Morgan & Claypool Publishers, 2012. (Cited on page 85.)

[93] Jason Flinn, SoYoung Park, and Mahadev Satyanarayanan. Balancing performance, energy, and quality in pervasive computing. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 217–226. IEEE, 2002. (Cited on page 110.)

[94] Jason Flinn and Mahadev Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 2–10. IEEE, 1999. (Cited on page 137.)

[95] Jacqueline Floch, Svein Hallsteinsen, Erlend Stav, Frank Eliassen, Ketil Lund, and Eli Gjorven. Using architecture models for runtime adaptability. *IEEE software*, 23(2):62–70, 2006. (Cited on page 3.)

[96] Chien-Liang Fok. *Adaptive middleware for resource-constrained mobile ad hoc and wireless sensor networks*. Washington University in St. Louis, 2009. (Cited on pages 18, 19, and 20.)

[97] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):16, 2009. (Cited on page 153.)

[98] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, 37(10):46–54, 2004. (Cited on pages 142 and 152.)

[99] Guillaume Gauvrit, Erwan Daubert, and Francoise Andre. Safdis: A framework to bring self-adaptability to service-based distributed applications. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, pages 211–218. IEEE, 2010. (Cited on page 4.)

[100] Simos Gerasimou, Radu Calinescu, Stepan Shevtsov, and Danny Weyns. Undersea: an exemplar for engineering self-adaptive unmanned underwater vehicles. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 83–89. IEEE Press, 2017. (Cited on pages 18 and 20.)

[101] GeSI: Global e-Sustainability Initiative. SMARTer2030 – ICT solutions for 21st century challenges. page 134, 2015. (Cited on page 108.)

[102] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. Living with uncertainty in the age of runtime models. In *Models@ run. time*, pages 47–100. Springer, 2014. (Cited on pages 18, 19, and 20.)

[103] Ivan Glesk, Tolulope Osadola, and Siti Idris. Enhancing data centre networking using energy aware optical interconnects. In *Transparent Optical Networks (ICTON), 2013 15th International Conference on*, pages 1–4. IEEE, 2013. (Cited on pages 35, 40, 43, and 56.)

[104] Theodore J Gordon and Howard Hayward. Initial experiments with the cross impact matrix method of forecasting. *Futures*, 1(2):100–116, 1968. (Cited on page 152.)

[105] Sebastian Götz, Thomas Ilsche, Jorge Cardoso, Josef Spillner, Uwe Aßmann, Wolfgang Nagel, and Alexander Schill. Energy-efficient data processing at sweet spot frequencies. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 154–171. Springer, 2014. (Cited on pages 18, 19, and 20.)

[106] Sebastian Götz, Julian Mendez, Veronika Thost, and Anni-Yasmin Turhan. Owl 2 reasoning to detect energy-efficient software variants from context. In *OWLED*, 2013. (Cited on pages 18, 19, and 20.)

[107] Sebastian Götz, René Schöne, Claas Wilke, Julian Mendez, and Uwe Aßmann. Towards predictive self-optimization by situation recognition. In *2nd Workshop EASED@ BUIS 2013*, page 11, 2013. (Cited on pages 18, 19, and 20.)

[108] Gabriel Guerrero-Contreras, Jose Luis Garrido, Sara Balderas-Diaz, and Carlos Rodríguez-Domínguez. A context-aware architecture supporting service availability in mobile cloud computing. *IEEE Transactions on Services Computing*, 10(6):956–968, 2017. (Cited on pages 18 and 20.)

[109] Govind P Gupta, Manoj Misra, and Kumkum Garg. Towards scalable and load-balanced mobile agents-based data aggregation for wireless sensor networks. *Computers & Electrical Engineering*, 64:262–276, 2017. (Cited on pages 18 and 20.)

[110] László Gyarmati and Tuan Anh Trinh. How can architecture help to reduce energy consumption in data center networking? In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 183–186. ACM, 2010. (Cited on pages 35, 40, 51, 52, 56, and 67.)

[111] Ali Hammadi and Lotfi Mhamdi. A survey on architectures and energy efficiency in data center networks. *Computer Communications*, 40:1–21, 2014. (Cited on page 21.)

[112] Neil B Harrison and Paris Avgeriou. How do architecture patterns and tactics interact? a model and annotation. *Journal of Systems and Software*, 83(10):1735–1758, 2010. (Cited on page 111.)

[113] Lamis Hawarah, Stéphane Ploix, and Mireille Jacomino. User behavior prediction in energy consumption in housing using bayesian networks. In *Artificial Intelligence and Soft Computing*, pages 372–379. Springer, 2010. (Cited on page 146.)

[114] Keqiang He, Yi Wang, Xiaofei Wang, Wei Meng, and Bin Liu. Greenvlan: An energy-efficient approach for vlan design. In *Computing, Networking*

and Communications (ICNC), 2012 International Conference on, pages 522–526. IEEE, 2012. (Cited on pages 34, 35, 40, 49, and 56.)

[115] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010. (Cited on pages 24, 34, 35, 40, 51, 56, 57, 66, 67, and 68.)

[116] Seth N Hetu, Vahid Saber Hamishagi, and Li-Shiuan Peh. Similitude: Interfacing a traffic simulator and network simulator with emulated android clients. In *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, pages 1–7. IEEE, 2014. (Cited on pages 114 and 137.)

[117] Henry Hoffmann. Coadapt: Predictable behavior for accuracy-aware applications running on power-aware systems. In *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, pages 223–232. IEEE, 2014. (Cited on pages 18 and 20.)

[118] Henry Hoffmann, Martina Maggio, Marco D Santambrogio, Alberto Leva, and Anant Agarwal. Seec: A framework for self-aware computing. 2010. (Cited on pages 18 and 20.)

[119] Md Farhad Hossain. Traffic-driven energy efficient operational mechanisms in cellular access networks. 2013. (Cited on pages 18, 19, and 20.)

[120] Yoshihiko Hotta, Mitsuhisa Sato, Hideaki Kimura, Satoshi Matsuoka, Taisuke Boku, and Daisuke Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp. IEEE, 2006. (Cited on page 3.)

[121] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238. ACM, 2012. (Cited on pages 116, 117, and 118.)

[122] Lei Huang, Qin Jia, Xin Wang, Shuang Yang, and Baochun Li. Pcube: Improving power efficiency in data center networks. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 65–72. IEEE, 2011. (Cited on pages 34, 35, 40, 51, and 56.)

[123] Mahmoud Hussein, Reda Nouacer, and Ansgar Radermacher. Safe adaptation of vehicle software systems. *Microprocessors and Microsystems*, 52:272–286, 2017. (Cited on pages 18 and 20.)

[124] Syed MAH Jafri, Liang Guang, Ahmed Hemani, Kolin Paul, Juha Plosila, and Hannu Tenhunen. Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes. *Microprocessors and Microsystems*, 37(8):811–822, 2013. (Cited on pages 18, 19, and 20.)

[125] Philip N Ji, Christoforos Kachris, Ioannis Tomkos, and Ting Wang. Energy efficient data center network based on a flexible bandwidth mimo ofdm optical interconnect. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 699–704. IEEE, 2012. (Cited on pages 3, 35, 40, 43, 44, 50, 53, 54, 56, and 57.)

[126] Jiming Jiang and Christian Claudel. A high performance, low power computational platform for complex sensing operations in smart cities. *HardwareX*, 1:22–37, 2017. (Cited on pages 18 and 20.)

[127] Hao Jin, Tosmate Cheocherngngarn, Dmita Levy, Alex Smith, Deng Pan, Jason Liu, and Niki Pissinou. Joint host-network optimization for energy-efficient data center networking. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 623–634. IEEE, 2013. (Cited on pages 34, 40, 51, 56, 57, 66, and 67.)

[128] Christoforos Kachris and Ioannis Tomkos. Power consumption evaluation of hybrid wdm pon networks for data centers. In *Networks and Optical Communications (NOC), 2011 16th European Conference on*, pages 118–121. IEEE, 2011. (Cited on pages 3, 35, 40, 44, 50, 52, and 56.)

[129] Christoforos Kachris and Ioannis Tomkos. Power consumption evaluation of all-optical data center networks. *Cluster Computing*, 16(3):611–623, 2013. (Cited on pages 35, 40, 43, 53, and 56.)

[130] Burak Kantarci, Luca Foschini, Antonio Corradi, and Hussein T Mouftah. Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 708–713. IEEE, 2012. (Cited on pages 34, 40, 56, 66, and 67.)

[131] Burak Kantarci, Luca Foschini, Antonio Corradi, and Hussein T Mouftah. Design of energy-efficient cloud systems via network and resource virtualization. *International Journal of Network Management*, 2013. (Cited on pages 35, 40, 43, 56, and 66.)

[132] Burak Kantarci and Hussein T Mouftah. Optimal reconfiguration of the cloud network for maximum energy savings. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 835–840. IEEE Computer Society, 2012. (Cited on pages 35, 40, 43, and 56.)

[133] Michał P Karpowicz, Piotr Arabas, and Ewa Niewiadomska-Szynkiewicz. Design and implementation of energy-aware application-specific cpu frequency governors for the heterogeneous distributed computing systems. *Future Generation Computer Systems*, 2016. (Cited on pages 18, 19, and 20.)

[134] Ramesh Karri and Piyush Mishra. Modeling energy efficient secure wireless networks using network simulation. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 1, pages 61–65. IEEE, 2003. (Cited on pages 114 and 136.)

[135] Osama Khader. Autonomous framework for supporting energy efficiency and communication reliability for periodic data flows in wireless sensor networks. 2014. (Cited on pages 18, 19, and 20.)

[136] Osama Khader, Andreas Willig, and Adam Wolisz. Self-learning and self-adaptive framework for supporting high reliability and low energy expenditure in wsns. *Telecommunication Systems*, 61(4):717–731, 2016. (Cited on pages 18 and 20.)

[137] Samee Ullah Khan and Albert Y Zomaya. *Handbook on Data Centers*. Springer, 2015. (Cited on page 76.)

[138] Hyunwoo Kim, Euijong Lee, and Doo-kwon Baik. Self-adaptive software simulation: A lighting control system for multiple devices. In *Asian Simulation Conference*, pages 380–391. Springer, 2017. (Cited on pages 18 and 20.)

[139] Sungchan Kim and Hoeseok Yang. An energy-aware runtime management of multi-core sensory swarms. *Sensors*, 17(9):1955, 2017. (Cited on pages 18 and 20.)

[140] Ken-ichi Kitayama, Soumitra Debnath, Yuki Yoshida, Ryo Takahashi, and Atsushi Hiramatsu. Energy-efficient, high-performance optoelectronic packet switching for intra-data center network. In *Transparent Optical Networks (ICTON), 2013 15th International Conference on*, pages 1–4. IEEE, 2013. (Cited on pages 35, 40, 43, 49, 53, and 56.)

[141] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009. (Cited on pages 6, 10, 24, and 26.)

[142] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Dens: data center energy-efficient network-aware scheduling. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on &*

Int'l Conference on Cyber, Physical and Social Computing (CPSCom), pages 69–75. IEEE, 2010. (Cited on page 23.)

[143] Gerald Kristen. *Object-Orientation: The Kiss Method: From Information Architecture to Information System*. Addison-Wesley Longman Publishing Co., Inc., 1994. (Cited on pages 6 and 142.)

[144] Mads Daro Kristensen and Niels Olof Bouvin. Developing cyber foraging applications for portable devices. In *Portable Information Devices, 2008 and the 2008 7th IEEE Conference on Polymers and Adhesives in Microelectronics and Photonics. PORTABLE-POLYTRONIC 2008. 2nd IEEE International Interdisciplinary Conference on*, pages 1–6. IEEE, 2008. (Cited on page 110.)

[145] Philippe Kruchten, Patricia Lago, and Hans van Vliet. Building up and reasoning about architectural knowledge. In Christine Hofmeister, Ivica Crnkovic, and Ralf Reussner, editors, *Quality of Software Architectures*, volume 4214 of *Lecture Notes in Computer Science*, pages 43–58. Springer Berlin Heidelberg, 2006. (Cited on page 85.)

[146] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206, 2015. (Cited on pages 10 and 21.)

[147] Aruzhan Kulseitova and Ang Tan Fong. A survey of energy-efficient techniques in cloud data centers. pages 1–5, June 2013. (Cited on page 25.)

[148] Shin-ichi Kuribayashi. Reducing total power consumption method in cloud computing environments. *International Journal of Computer Networks & Communications*, 4(2), 2012. (Cited on pages 34, 40, and 56.)

[149] Hyun Jung La and Soo Dong Kim. A taxonomy of offloading in mobile cloud computing. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pages 147–153. IEEE, 2014. (Cited on page 111.)

[150] Patricia Lago and Paris Avgeriou. First workshop on sharing and reusing architectural knowledge. *SIGSOFT Software Engineering Notes*, 31(5):32–36, September 2006. (Cited on page 85.)

[151] Hyo-Cheol Lee and Seok-Won Lee. Towards knowledge-intensive software engineering framework for self-adaptive software. In *SEKE*, pages 30–35, 2015. (Cited on pages 18 and 20.)

[152] Aris Leivadeas, Chrysa Papagianni, and Symeon Papavassiliou. Energy aware networked cloud mapping. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 195–202. IEEE, 2013. (Cited on pages 34, 40, 56, and 66.)

[153] Adam Lella and Andrew Lipsman. The 2017 u.s. mobile app report, 2017. [Online; accessed 26-Feb-2019]. URL: `https://www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-2017-US-Mobile-App-Report?cs_edgescape_cc=NL`. (Cited on page 123.)

[154] Grace Lewis and Patricia Lago. Architectural tactics for cyber-foraging: Results of a systematic literature review. *Journal of Systems and Software*, 107:158–186, 2015. (Cited on pages 85, 86, and 87.)

[155] Grace Lewis and Patricia Lago. Architectural tactics for cyber-foraging: Results of a systematic literature review. *Journal of Systems and Software*, 107:158–186, 2015. (Cited on page 147.)

[156] Grace Lewis, Patricia Lago, and Giuseppe Procaccianti. Architecture strategies for cyber-foraging: Preliminary results from a systematic literature review. In Paris Avgeriou and Uwe Zdun, editors, *Proceedings of the 8th European Conference on Software Architecture (ECSA 2014)*, volume 8627 of *Lecture Notes in Computer Science*, pages 154–169. Springer International Publishing, 2014. (Cited on pages 84 and 86.)

[157] Grace A Lewis. Software architecture strategies for cyber-foraging systems. 2016. (Cited on page 86.)

[158] Grace A. Lewis and Patricia Lago. A catalog of architectural tactics for cyber-foraging. In *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, QoSA '15, pages 53–62, New York, NY, USA, 2015. ACM. (Cited on page 86.)

[159] Grace A. Lewis and Patricia Lago. Characterization of cyber-foraging usage contexts. In Danny Weyns, Raffaela Mirandola, and Ivica Crnkovic, editors, *Software Architecture*, volume 9278 of *LNCS*, pages 195–211. Springer International Publishing, 2015. (Cited on pages 97 and 108.)

[160] Grace A. Lewis, Patricia Lago, and Paris Avgeriou. A decision model for cyber-foraging systems. In *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA 2016)*, pages 51–60. IEEE, 2016. (Cited on pages 84, 87, 91, and 111.)

[161] Ding Li, Yingjun Lyu, Jiaping Gui, and William GJ Halfond. Automated energy optimization of http requests for mobile applications. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 249–260. IEEE, 2016. (Cited on page 114.)

[162] Zheng Ling, Zhang Bin, and Wang Jiye. Application of the snowflake structure in the network structure of electric power enterprise data center. In *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, pages 900–903. IEEE, 2012. (Cited on page 51.)

[163] Marin Litoiu, Mary Shaw, Gabriel Tamura, Norha M Villegas, Hausi A Müller, Holger Giese, Romain Rouvoy, and Eric Rutten. What can control theory teach us about assurances in self-adaptive software systems? In *Software Engineering for Self-Adaptive Systems III. Assurances*, pages 90–134. Springer, 2017. (Cited on pages 18 and 20.)

[164] Jie Liu, Feng Zhao, Xue Liu, and Wenbo He. Challenges towards elastic power management in internet data centers. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pages 65–72. IEEE, 2009. (Cited on page 65.)

[165] Jieyao Liu, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya, and Ahsan Qureshi. Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, 48:99–117, 2015. (Cited on page 111.)

[166] Ruoyan Liu, Huaxi Gu, Xiaoshan Yu, and Xiumei Nian. Distributed flow scheduling in energy-aware data center networks. *Communications Letters, IEEE*, 17(4):801–804, 2013. (Cited on pages 34, 35, 40, 51, 56, and 67.)

[167] Frank D Macías-Escrivá, Rodolfo Haber, Raul Del Toro, and Vicente Hernandez. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, 40(18):7267–7279, 2013. (Cited on pages 10 and 21.)

[168] Priya Mahadevan, Sujata Banerjee, Puneet Sharma, Amip Shah, and Parthasarathy Ranganathan. On energy efficiency for enterprise and data center networks. *Communications Magazine, IEEE*, 49(8):94–100, 2011. (Cited on pages 34, 40, and 56.)

[169] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. Energy aware network operations. In *INFOCOM Workshops 2009, IEEE*, pages 1–6. IEEE, 2009. (Cited on pages 34, 40, 56, and 57.)

[170] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *NETWORKING 2009*, pages 795–808. Springer, 2009. (Cited on page 76.)

[171] Sara Mahdavi-Hezavehi, Vinicius HS Durelli, Danny Weyns, and Paris Avgeriou. A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems. *Information and Software Technology*, 90:1–26, 2017. (Cited on page 21.)

[172] Vijay Mann, Avinash Kumar, Partha Dutta, and Shivkumar Kalyanaraman. Vmflow: leveraging vm mobility to reduce network power costs in data centers. In *NETWORKING 2011*, pages 198–211. Springer, 2011. (Cited on pages 34, 40, 51, and 56.)

[173] Rick McGeer, Priya Mahadevan, and Sujata Banerjee. On the complexity of power minimization schemes in data center networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010. (Cited on pages 3, 34, 40, 51, and 56.)

[174] Alexandre Mello Ferreira. *Energy aware service based information systems*. PhD thesis, Italy, 2013. (Cited on pages 18, 19, and 20.)

[175] Johannes Mey, René Schöne, Daniel Langner, and Christoff Bürger. Using reference attribute grammar-controlled rewriting for runtime resource management. In *RES4ANT@ DATE*, pages 13–17, 2016. (Cited on pages 18, 19, and 20.)

[176] Nikita Mishra, Huazhe Zhang, John D Lafferty, and Henry Hoffmann. A probabilistic graphical model-based approach for minimizing energy under performance constraints. In *ACM SIGPLAN Notices*, volume 50, pages 267–281. ACM, 2015. (Cited on pages 18, 19, and 20.)

[177] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 317–328. ACM, 2012. (Cited on page 113.)

[178] Rabeb Mizouni, Mohammad Abu Matar, Zaid Al Mahmoud, Salwa Alzahmi, and Aziz Salah. A framework for context-aware self-adaptive mobile applications spl. *Expert Systems with applications*, 41(16):7549–7564, 2014. (Cited on page 137.)

[179] Rabeb Mizouni, M Adel Serhani, Abdelghani Benharref, and Oubai Al-Abassi. Towards battery-aware self-adaptive mobile applications. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 439–445. IEEE, 2012. (Cited on pages 18, 19, and 20.)

[180] Mirko Morandini, Loris Penserini, Anna Perini, and Alessandro Marchetto. Engineering requirements for adaptive systems. *Requirements Engineering*, 22(1):77–103, 2017. (Cited on pages 18 and 20.)

[181] María V Moreno-Cano, José Santa, Miguel A Zamora-Izquierdo, and Antonio F Skarmeta. Future human-centric smart environments. In *Modeling and Processing for Next-Generation Big-Data Technologies*, pages 341–365. Springer, 2015. (Cited on pages 18, 19, and 20.)

[182] Henry Muccini, Mohammad Sharaf, and Danny Weyns. Self-adaptation for cyber-physical systems: a systematic literature review. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 75–81. ACM, 2016. (Cited on page 21.)

[183] Mohamad Najem, Pascal Benoit, Mohamad El Ahmad, Gilles Sassatelli, and Lionel Torres. A design-time method for building cost-effective runtime power monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(7):1153–1166, 2017. (Cited on pages 18 and 20.)

[184] Thanh Nguyen Huu, Nam Pham Ngoc, Huong Truong Thu, Thuan Tran Ngoc, Duong Nguyen Minh, Van Giang Nguyen, Hung Nguyen Tai, Thu Ngo Quynh, David Hock, and Christian Schwartz. Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks. *Simulation Modelling Practice and Theory*, 39:20–40, 2013. (Cited on pages 34, 40, and 56.)

[185] Xinyu Niu, Kuen Hung Tsoi, and Wayne Luk. Self-adaptive heterogeneous cluster with wireless network. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 306–311. IEEE, 2012. (Cited on pages 15, 18, 19, and 20.)

[186] Khamla Non-Alisavath, Somphone Kanthavong, Khanthanou Luangxaysana, and Xaythavy Louangvilay. Context-awareness application to control multiple sensors for monitoring smart environment. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2017 14th International Conference on*, pages 920–923. IEEE, 2017. (Cited on pages 18 and 20.)

[187] Institute of Electrical and NY USA Electronics Engineers (IEEE), New York. Ieee 802.11g-2003: Further higher data rate extension in the 2.4ghz band, 2003. [Online; accessed 17-Jul-2017]. URL: `https://standards.ieee.org/getieee802/download/802.11g-2003.pdf`. (Cited on page 116.)

[188] Peyman Oreizy, Michael M Gorlick, Richard N Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S Rosenblum, and Alexander L Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent systems*, 14(3):54–62, 1999. (Cited on pages 142 and 152.)

[189] Peyman Oreizy, Nenad Medvidovic, and Richard N Taylor. Runtime software adaptation: framework, approaches, and styles. In *Companion of the 30th international conference on Software engineering*, pages 899–910. ACM, 2008. (Cited on pages 142 and 152.)

[190] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys (CSUR)*, 46(4):47, 2014. (Cited on pages 21 and 25.)

[191] Gabriel Orsini, Dirk Bade, and Winfried Lamersdorf. Context-aware computation offloading for mobile cloud computing: Requirements analysis, survey and design guideline. *Procedia Computer Science*, 56:10–17, 2015. (Cited on page 111.)

[192] Cathryn Peoples, Gerard Parr, and Sally McClean. Energy-aware data centre management. In *Communications (NCC), 2011 National Conference on*, pages 1–5. IEEE, 2011. (Cited on pages 35, 40, 54, and 56.)

[193] Cathryn Peoples, Gerard Parr, Sally McClean, B Scotney, P Morrow, SK Chaudhari, and Ravi Theja. An energy aware network management approach using server profiling in'green'clouds. In *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, pages 17–24. IEEE, 2012. (Cited on pages 35, 40, and 56.)

[194] Eldad Perahia and Robert Stacey. *Next generation wireless LANs: 802.11 n and 802.11 ac*. Cambridge university press, 2013. (Cited on page 116.)

[195] Diego Perez-Palacin, Raffaela Mirandola, and José Merseguer. Qos and energy management with petri nets: A self-adaptive framework. *Journal of Systems and Software*, 85(12):2796–2811, 2012. (Cited on pages 15, 18, and 20.)

[196] Gustavo Pinto and Fernando Castor. Energy efficiency: a new concern for application software developers. *Communications of the ACM*, 60(12):68–75, 2017. (Cited on page 21.)

[197] Jesús MT Portocarrero, Flavia C Delicato, Paulo F Pires, and Thais V Batista. Reference architecture for self-adaptive management in wireless

sensor networks. In *Adaptive and Intelligent Systems*, pages 110–120. Springer, 2014. (Cited on page 153.)

[198] Jesús MT Portocarrero, Flavia C Delicato, Paulo F Pires, Bruno Costa, Wei Li, Weisheng Si, and Albert Y Zomaya. Ramses: a new reference architecture for self-adaptive middleware in wireless sensor networks. *Ad Hoc Networks*, 55:3–27, 2017. (Cited on pages 18, 19, and 20.)

[199] Bhanu Priya, Emmanuel S Pilli, and Ramesh C Joshi. A survey on energy and power consumption models for greener cloud. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 76–82. IEEE, 2013. (Cited on page 25.)

[200] Giuseppe Procaccianti, Patricia Lago, Antonio Vetro, Daniel Méndez Fernández, and Roel Wieringa. The green lab: Experimentation in software energy efficiency. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, pages 941–942, 2015. (Cited on page 10.)

[201] Moo-Ryong Ra, Anmol Sheth, Lily Mummert, Padmanabhan Pillai, David Wetherall, and Ramesh Govindan. Odessa: enabling interactive perception applications on mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 43–56. ACM, 2011. (Cited on page 110.)

[202] Massimiliano Raciti. *Anomaly detection and its adaptation: Studies on cyber-physical systems*. PhD thesis, Linköping University Electronic Press, 2013. (Cited on pages 18, 19, and 20.)

[203] Massimiliano Raciti, Jordi Cucurull, and Simin Nadjm-Tehrani. Energy-based adaptation in simulations of survivability of ad hoc communication. In *Wireless Days (WD), 2011 IFIP*, pages 1–7. IEEE, 2011. (Cited on pages 18 and 20.)

[204] C Raibulet, F Arcelli Fontana, R Capilla, and C Carrillo. An overview on quality evaluation of self-adaptive systems. 2016. (Cited on page 142.)

[205] Andres J Ramirez, Adam C Jensen, and Betty HC Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 99–108. IEEE Press, 2012. (Cited on pages 18 and 20.)

[206] Gérald Rocher, Jean-Yves Tigli, and Stéphane Lavirotte. Probabilistic models toward controlling smart-* environments. *IEEE Access*, 5:12338–12352, 2017. (Cited on pages 18 and 20.)

[207] Romain Rouvoy. *Contributions to the Autonomy of Ubiquitous Software Systems*. PhD thesis, Université de Lille 1, Sciences et Technologies, 2014. (Cited on pages 18, 19, and 20.)

[208] Diego Rughetti, Pierangelo Di Sanzo, and Alessandro Pellegrini. Adaptive transactional memories: Performance and energy consumption tradeoffs. In *Network Cloud Computing and Applications (NCCA), 2014 IEEE 3rd Symposium on*, pages 105–112. IEEE, 2014. (Cited on pages 18 and 20.)

[209] Linnyer Beatrys Ruiz, Jose Marcos Nogueira, and Antonio AF Loureiro. Manna: A management architecture for wireless sensor networks. *IEEE communications Magazine*, 41(2):116–125, 2003. (Cited on page 153.)

[210] Eric Rutten, Nicolas Marchand, and Daniel Simon. *Feedback Control as MAPE-K loop in Autonomic Computing*. PhD thesis, INRIA Sophia Antipolis-Méditerranée; INRIA Grenoble-Rhône-Alpes, 2015. (Cited on pages 18, 19, and 20.)

[211] Shivashis Saha, Jitender S Deogun, and Lisong Xu. Energy models driven green routing for data centers. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2529–2534. IEEE, 2012. (Cited on pages 34, 36, 40, and 56.)

[212] Mouna Ben Said, Yessine Hadj Kacem, Mickaël Kerboeuf, Nader Ben Amor, and Mohamed Abid. Design patterns for self-adaptive rte systems specification. *International Journal of Reconfigurable Computing*, 2014:8, 2014. (Cited on pages 18, 19, and 20.)

[213] Maria Salama and Rami Bahsoon. A taxonomy for architectural stability. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1354–1357. ACM, 2016. (Cited on page 3.)

[214] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):14, 2009. (Cited on page 152.)

[215] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, Aug 2001. (Cited on page 85.)

[216] Pete Sawyer, Raul Mazo, Daniel Diaz, Camille Salinesi, and Danny Hughes. Using constraint programming to manage configurations in self-adaptive systems. *Computer*, 45(10):56–63, 2012. (Cited on pages 18, 19, and 20.)

[217] Philipp Schleiss, Marc Zeller, Gereon Weiss, and Dirk Eilers. Safeadapt-safe adaptive software for fully electric vehicles. In *3rd Conference on Future Automotive Technology, CoFAT*, 2014. (Cited on pages 18 and 20.)

[218] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Aßmann. Toward a framework for self-adaptive workflows in cyber-physical systems. *Software & Systems Modeling*, pages 1–18, 2017. (Cited on pages 18 and 20.)

[219] Daniele De Sensi, Massimo Torquati, and Marco Danelutto. A reconfiguration algorithm for power-aware parallel applications. *ACM Transactions on Architecture and Code Optimization (TACO)*, 13(4):43, 2016. (Cited on pages 18, 19, and 20.)

[220] Chiyoung Seo, Sam Malek, and Nenad Medvidovic. Estimating the energy consumption in pervasive java-based systems. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 243–247. IEEE, 2008. (Cited on page 137.)

[221] Filippo Seracini. *A Proactive Top-Down Approach to Dynamic Allocation of Resources in Data Centers*. University of California, San Diego, 2014. (Cited on pages 18, 19, and 20.)

[222] Estefanía Serral, Paolo Sernani, and Fabiano Dalpiaz. Personalized adaptation in pervasive systems via non-functional requirements. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–15, 2017. (Cited on pages 18 and 20.)

[223] Li Shang, Li-Shiuan Peh, and Niraj K Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91–102. IEEE, 2003. (Cited on page 3.)

[224] Yunfei Shang, Dan Li, and Mingwei Xu. Energy-aware routing in data center network. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 1–8. ACM, 2010. (Cited on pages 34, 40, 46, 47, 51, 56, and 67.)

[225] Yunfei Shang, Dan Li, and Mingwei Xu. A comparison study of energy proportionality of data center network architectures. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 1–7. IEEE, 2012. (Cited on pages 34, 40, 47, 48, 51, and 56.)

[226] Yunfei Shang, Dan Li, and Mingwei Xu. Greening data center networks with flow preemption and energy-aware routing. In *Local & Metropolitan Area Networks (LANMAN), 2013 19th IEEE Workshop on*, pages 1–6. IEEE, 2013. (Cited on pages 34, 40, 47, and 56.)

[227] Mohsen Sharifi, Somayeh Kafaie, and Omid Kashefi. A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys & Tutorials*, 14(4):1232–1243, 2012. (Cited on pages 85 and 111.)

[228] Hiroki Shirayanagi, Hiroshi Yamada, and KONO Kenji. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. *IEICE TRANSACTIONS on Information and Systems*, 96(9):2055–2064, 2013. (Cited on pages 34, 40, 51, and 56.)

[229] Muhammad Shiraz, Abdullah Gani, Rashid Hafeez Khokhar, and Rajkumar Buyya. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys & Tutorials*, 15(3):1294–1313, 2013. (Cited on page 111.)

[230] Weisheng Si, Javid Taheri, and Albert Zomaya. A distributed energy saving approach for ethernet switches in data centers. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 505–512. IEEE, 2012. (Cited on pages 34, 36, 40, 44, 47, 56, and 57.)

[231] Soumya Simanta, Kiryong Ha, Grace Lewis, Ed Morris, and Mahadev Satyanarayanan. A reference architecture for mobile code offload in hostile environments. In *International Conference on Mobile Computing, Applications, and Services*, pages 274–293. Springer, 2012. (Cited on page 110.)

[232] Filippo Sironi. System support for adaptive performance and thermal management of chip multiprocessors. 2014. (Cited on pages 18, 19, and 20.)

[233] Mohammed Sourouri, Espen Birger Raknes, Nico Reissmann, Johannes Langguth, Daniel Hackenberg, Robert Schöne, and Per Gunnar Kjeldsberg. Towards fine-grained dynamic tuning of hpc applications on modern multicore architectures. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 41. ACM, 2017. (Cited on pages 18 and 20.)

[234] Statistica. Number of available applications in the google play store from december 2009 to june 2017, 2017. [Online; accessed 26-Feb-2019]. URL: `https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/`. (Cited on page 113.)

[235] Christian Stier, Henning Groenda, and Anne Koziolek. Towards modeling and analysis of power consumption of self-adaptive software systems in palladio. In *SOSP14 Symposium on Software Performance: Joint Descartes/Kieker/Palladio Days 2014*, page 28, 2014. (Cited on pages 18, 19, and 20.)

[236] Anselm L Strauss. *Qualitative analysis for social scientists*. Cambridge University Press, 1987. (Cited on pages 13, 24, and 30.)

[237] Gang Sun, Hongfang Yu, Vishal Anand, Dan Liao, and Lemin Li. Exploring power-efficient provisioning for online virtual network requests. In *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, pages 51–55. IEEE, 2012. (Cited on pages 34, 35, 40, and 56.)

[238] Ted H Szymanski. Maximum flow minimum energy routing for exascale cloud computing systems. In *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on*, pages 89–95. IEEE, 2013. (Cited on pages 3, 34, 40, 46, 48, and 56.)

[239] Antony Tang and Man F. Lau. Software architecture review by association. *Journal of Systems and Software*, 88:87–101, 2014. (Cited on page 151.)

[240] Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata. A virtual network to achieve low energy consumption in optical large-scale datacenter. In *Communication Systems (ICCS), 2012 IEEE International Conference on*, pages 45–49. IEEE, 2012. (Cited on pages 3, 35, 40, 44, 52, and 56.)

[241] Nguyen Huu Thanh, Pham Ngoc Nam, Thu-Huong Truong, Nguyen Tai Hung, Luong Kim Doanh, and Rastin Pries. Enabling experiments for energy-efficient data center networks on openflow-based platform. In *Communications and Electronics (ICCE), 2012 Fourth International Conference on*, pages 239–244. IEEE, 2012. (Cited on pages 34, 35, 40, 51, 56, and 76.)

[242] Sergey Tyurin and Anton Kamenskih. Green logic: models, methods, algorithms. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, pages 69–86. Springer, 2017. (Cited on pages 18 and 20.)

[243] Alejandro Valero, Negar Miralaei, Salvador Petit, Julio Sahuquillo, and Timothy M Jones. On microarchitectural mechanisms for cache wearout reduction. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):857–871, 2017. (Cited on pages 18 and 20.)

[244] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet, and Piet Demeester. Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014. (Cited on page 1.)

[245] Archana Venkatraman. Global census shows datacentre power demand grew 63% in 2012. October 2012. [Online; accessed 2-July-2014]. (Cited on page 23.)

[246] Antonio Vincenzo Taddeo, Pierpaolo Marcon, and Alberto Ferrante. Negotiation of security services: a multi-criteria decision approach. In *Proceedings of the 4th Workshop on Embedded Systems Security*, page 4. ACM, 2009. (Cited on pages 18 and 20.)

[247] Lin Wang, Fa Zhang, Jordi Arjona Aroca, Athanasios V Vasilakos, Kai Zheng, Chenying Hou, Dan Li, and Zhiyong Liu. Greendcn: A general framework for achieving energy efficiency in data center networks. *Selected Areas in Communications, IEEE Journal on*, 32(1):4–15, 2014. (Cited on page 67.)

[248] Lin Wang, Fa Zhang, Jordi Arjona Aroca, Athanasios V Vasilakos, Kai Zheng, Chenying Hou, Dan Li, and Zhiyong Liu. A general framework for achieving energy efficiency in data center networks. *arXiv preprint arXiv:1304.3519*, 2013. (Cited on pages 34, 35, 40, 41, 48, 51, 56, and 57.)

[249] Lin Wang, Fa Zhang, Athanasios V Vasilakos, Chenying Hou, and Zhiyong Liu. Joint virtual machine assignment and traffic engineering for green data center networks. *ACM SIGMETRICS Performance Evaluation Review*, 41(3):107–112, 2014. (Cited on page 67.)

[250] Xiaodong Wang, Yanjun Yao, Xiaorui Wang, Kefa Lu, and Qing Cao. Carpo: Correlation-aware power optimization in data center networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1125–1133. IEEE, 2012. (Cited on pages 34, 35, 38, 40, 56, 66, and 67.)

[251] Nicolas Weber. *GPU Array Access Auto-Tuning*. PhD thesis, Technische Universität, 2017. (Cited on pages 18, 19, and 20.)

[252] Danny Weyns, Sam Malek, and Jesper Andersson. Forms: Unifying reference model for formal specification of distributed self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):8, 2012. (Cited on pages 143 and 153.)

[253] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty HC Cheng, and Jean-Michel Bruel. Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, 15(2):177–196, 2010. (Cited on pages 18, 19, and 20.)

[254] Weihang Wu and Tim Kelly. Safety tactics for software architecture design. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 368–375. IEEE, 2004. (Cited on page 111.)

[255] Simon Wunderlich, Juan A Cabrera, Frank HP Fitzek, and Martin Reisslein. Network coding in heterogeneous multicore iot nodes with dag scheduling of parallel matrix block operations. *IEEE Internet of Things Journal*, 4(4):917–933, 2017. (Cited on pages 18 and 20.)

[256] Bin Xu, Jin Qi, Xiaoxuan Hu, Kwong-Sak Leung, Yanfei Sun, and Yu Xue. Self-adaptive bat algorithm for large scale cloud manufacturing service composition. *Peer-to-Peer Networking and Applications*, pages 1–14, 2017. (Cited on pages 18 and 20.)

[257] Mingwei Xu, Yunfei Shang, Dan Li, and Xin Wang. Greening data center networks with throughput-guaranteed power-aware routing. *Computer Networks*, 57(15):2880–2899, 2013. (Cited on pages 34, 35, 40, 56, 67, and 76.)

[258] Siyuan Xu and Benjamin Carrion Schafer. Approximate reconfigurable hardware accelerator: Adapting the micro-architecture to dynamic workloads. In *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pages 113–120. IEEE, 2017. (Cited on pages 18 and 20.)

[259] Kun Yang, Shumao Ou, and Hsiao-Hwa Chen. On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications. *IEEE communications magazine*, 46(1), 2008. (Cited on page 110.)

[260] Lei Yang, Jiannong Cao, Yin Yuan, Tao Li, Andy Han, and Alvin Chan. A framework for partitioning and execution of data stream applications in mobile cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 40(4):23–32, 2013. (Cited on page 2.)

[261] Zhuoqun Yang, Zhi Jin, and Zhi Li. A model-based fuzzy control approach to achieving adaptation with contextual uncertainties. *arXiv preprint arXiv:1704.00417*, 2017. (Cited on pages 18 and 20.)

[262] Zhuoqun Yang, Zhi Jin, and Zhi Li. Modeling uncertainty and evolving self-adaptive software: A fuzzy theory based requirements engineering approach. *arXiv preprint arXiv:1704.00873*, 2017. (Cited on pages 18 and 20.)

[263] Zhuoqun Yang, Zhi Li, Zhi Jin, and Yunchuan Chen. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 55–71. Springer, 2014. (Cited on pages 10 and 21.)

[264] Yasir Zaki, Liang Zhao, Carmelita Goerg, and Andreas Timm-Giel. Lte mobile network virtualization. *Mobile Networks and Applications*, 16(4):424–432, 2011. (Cited on page 136.)

[265] Di Zhang, Yaoxue Zhang, Yuezhi Zhou, and Hao Liu. Leveraging the tail time for saving energy in cellular networks. *IEEE Transactions on Mobile Computing*, 13(7):1536–1549, 2014. (Cited on page 114.)

[266] Guangda Zhang, Wei Song, Jim Garside, Javier Navaridas, and Zhiying Wang. Handling physical-layer deadlock caused by permanent faults in quasi-delay-insensitive networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(11):3152–3165, 2017. (Cited on pages 18 and 20.)

[267] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES/ISSS '10, pages 105–114, New York, NY, USA, 2010. ACM. (Cited on pages 102, 106, and 109.)

[268] Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications*, 16(3):270–284, 2011. (Cited on page 110.)

[269] Yi Zhang, Pulak Chowdhury, Massimo Tornatore, and Biswanath Mukherjee. Energy efficiency in telecom optical networks. *Communications Surveys & Tutorials, IEEE*, 12(4):441–458, 2010. (Cited on page 25.)

[270] Steffen Ziegert and Heike Wehrheim. Temporal plans for software architecture reconfiguration. *Computer Science-Research and Development*, 30(3-4):303–320, 2015. (Cited on pages 18 and 20.)

Acknowledgements

Now, it is time to take a look back to the journey I made to finish my PhD thesis. It would have not been possible without the support of so many people in the past years.

First and foremost, I would like to show my gratitude to my supervisors and promotors: Dr. Paola Grosso, Prof. dr. Patricia Lago and Prof. dr. Cees de Laat. With Paola, I had many fruitful discussions on countless topics. She has taught me how to enable the "systems thinking" in myself and be a pragmatic engineer. Patricia has taught me to see the "big picture" and be a right-to-the-point designer. Since my master studies, I had the privilege to learn many things from Cees, both research-related and work-related. I can not thank them enough for the support and education they have provided me. I am very grateful to my thesis committee for their valuable feedback. Thanks to my paranymphs, Eoin Grua and Pieter Hjima for their help through the defense.

To perform this research, I had the opportunity to work with many talented colleagues from two research groups S2 and SNE: Paolo, Guiseppe, Ivano, Eoin, Ameneh, Maryam, Grace, Nelly, Roberto, Pieter, Robert, Ralph, Hao and many others. I enjoyed very much to spend time with them in social activities and our meetings. I would like to thank all my friends from all over the world. I am so grateful to have the chance to meet each and any of them. My Persian friends back from home and in the Netherlands have encouraged and supported me all way long. They have been there for me in the ups and downs of my PhD years.

I am very thankful to my parents, Mahnaz and Alireza. I am so grateful for the love and support they have given me all my life. They have taught me to see the positive side in any situation, and to work hard for my goals. I would like to thank my brother and my sister-in-law, Hossein and Shide. Hossein and I have been the best buddies all our lives. I wish Hossein and our kind Shide many more happy years. I would like to thank my parents' in-law and brother-in-law, Yousef, Zahra, and Ali. Many thanks for their support.

Sepehr and Ava, the two highlights of my life! You have brought me so much joy and love that I can not even describe. I feel so grateful to have you both. Mohammad, my dear husband, we started this journey together. Many thanks for believing in me and supporting me in my decisions.

Summary

The ICT sector consumes a large portion of the total energy supply in the world. The increasing number of ICT users, services, and infrastructures also suggests that the energy consumption of the ICT sector will grow even more significantly in the coming years. As a matter of fact, software defines how ICT infrastructure should be utilized. Inefficiencies in software propagate easily throughout the entire system. Therefore, software should be the main focus of energy efficiency solutions in the ICT sector. Ideally, software systems should be alert to their own energy consumption during the execution, and if the resources availability changes, they must adapt themselves to the new situation.

In this dissertation, we explore the relationship between energy efficiency and self-adaptability of software systems. We distinguish between architectural solutions and infrastructural solutions. As for the former, we use software architecture as the main instrument to carry over energy-related design decisions. As for the latter, we evaluate the effectiveness of optimization algorithms and software-defined infrastructures. Lastly, we introduce a domain model for self-adaptive software systems. The model includes both architectural and infrastructural concepts, which provides the reader with a clearer image of all ingredients to enable self-adaptability.

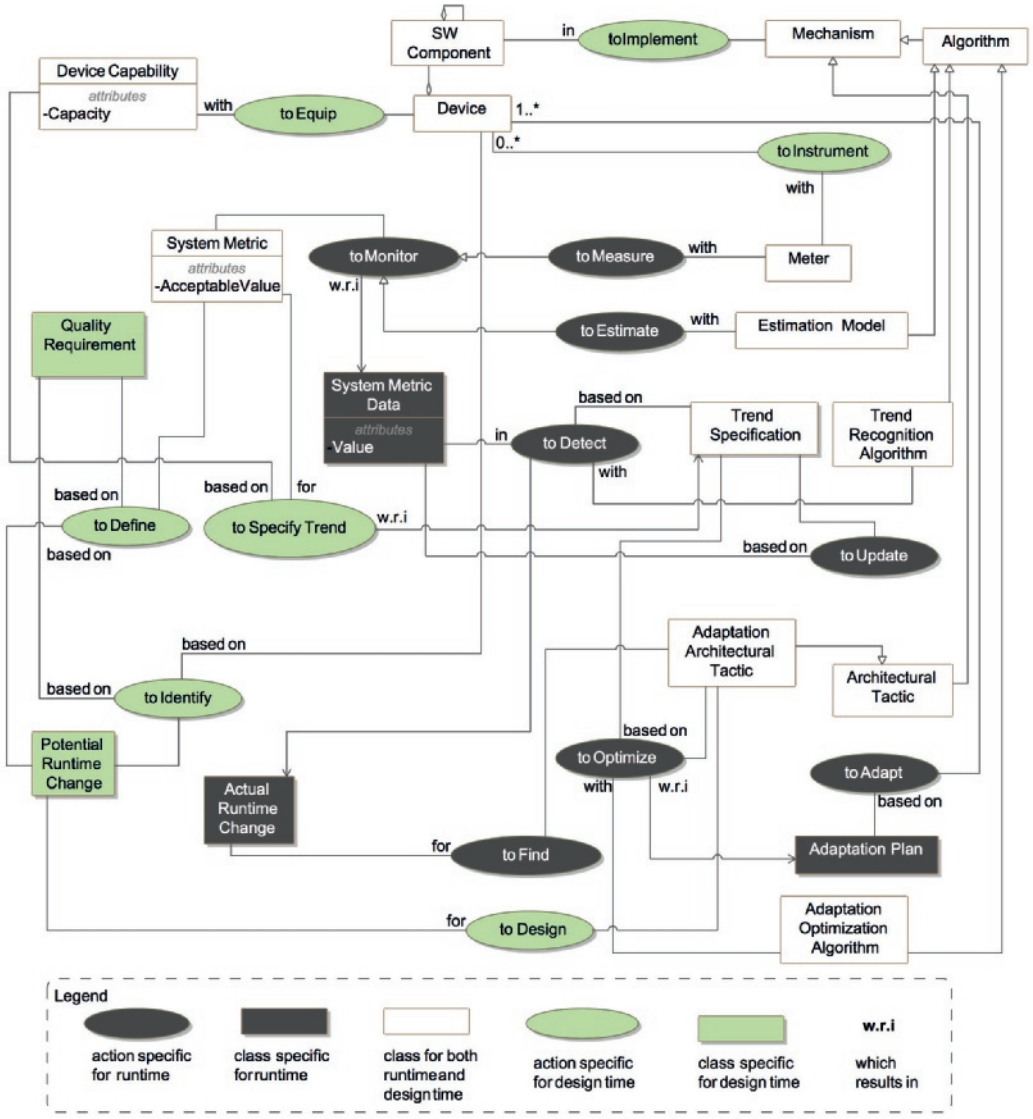
Samenvatting

De ICT-sector verbruikt een groot deel van de totale energievoorziening in de wereld. Het toenemende aantal ICT-gebruikers, -diensten en -infrastructuren suggereert ook dat het energieverbruik van de ICT-sector de komende jaren nog sterker zal stijgen. Software definieert feitelijk hoe ICT-infrastructuur moet worden gebruikt. Inefficinties in software verspreiden zich gemakkelijk door het hele systeem. Daarom moet software de belangrijkste focus zijn van oplossingen voor energie-efficintie in de ICT-sector. Idealiter moeten softwaresystemen alert zijn op hun eigen energieverbruik tijdens de uitvoering, en als de beschikbaarheid van hulpbronnen verandert, moeten ze zich aanpassen aan de nieuwe situatie.

In dit proefschrift onderzoeken we de relatie tussen energie-efficintie en zelfaanpassing van softwaresystemen. We onderscheiden architecturale oplossingen en infrastructurele oplossingen. Wat de eerste betreft, gebruiken we software- architectuur als het belangrijkste instrument om ontwerpbeslissingen met betrekking tot energie over te nemen. Wat dit laatste betreft, evalueren we de effectiviteit van optimalisatie-algoritmen en softwaregedefinieerde infrastructuren. Ten slotte introduceren we een domeinmodel voor zelf-adaptieve softwaresystemen. Het model bevat zowel architecturale als infrastructurele concepten, waardoor de lezer een duidelijker beeld krijgt van alle ingredinten om zichzelf aan te passen.

Appendix A

The domain model for self-adaptive software systems



2019-01. Rob van Eijk (UL),Comparing and Aligning Process Representations
2019-02. Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
2019-03. Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
2019-04. Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
2019-05. Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
2019-06. Chris Dijkshoorn (VUA), Niche sourcing for Improving Access to Linked Cultural Heritage Datasets
2019-07. Soude Fazeli (OUN), Recommender Systems in Social Learning Platforms
2019-08. Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
2018-01. Han van der Aa (VUA), Comparing and Aligning Process Representations
2018-02. Felix Mannhardt (TUE), Multi-perspective Process Mining
2018-03. Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
2018-04. Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
2018-05. Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process
2018-06. Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
2018-07. Jieting Luo (UU), A formal account of opportunism in multi-agent systems
2018-08. Rick Smetsers (RUN), Advances in Model Learning for Software Systems
2018-09. Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
2018-10. Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
2018-11. Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks
2018-12. Xixi Lu (TUE), Using behavioral context in process mining
2018-13. Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
2018-14. Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
2018-15. Naser Davarzani (UM), Biomarker discovery in heart failure
2018-16. Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
2018-17. Jianpeng Zhang (TUE), On Graph Sample Clustering
2018-18. Henriette Nakad (UL), De Notaris en Private Rechtspraak
2018-19. Minh Duc Pham (VUA), Emergent relational schemas for RDF
2018-20. Manxia Liu (RUN), Time and Bayesian Networks
2018-21. Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
2018-22. Eric Fernandes de Mello Araujo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
2018-23. Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
2018-24. Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
2018-25. Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
2018-26. Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
2018-27. Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis

2018-28. Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
2018-29. Yu Gu (UVT), Emotion Recognition from Mandarin Speech
2018-30. Wouter Beek, The "K" in "semantic web" stands for "knowledge": scaling semantics to the web
2017-01. Jan-Jaap Oerlemans (UL), Investigating Cybercrime
2017-02. Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
2017-03. Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
2017-04. Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
2017-05. Mahdieh Shadi (UVA), Collaboration Behavior
2017-06. Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
2017-07. Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
2017-08. Rob Konijn (VU), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
2017-09. Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
2017-10. Robby van Delden (UT), (Steering) Interactive Play Behavior
2017-11. Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
2017-12. Sander Leemans (TUE), Robust Process Mining with Guarantees
2017-13. Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
2017-14. Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
2017-15. Peter Berck (RUN), Memory-Based Text Correction
2017-16. Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
2017-17. Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
2017-18. Ridho Reinanda (UVA), Entity Associations for Search
2017-19. Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
2017-20. Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
2017-21. Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
2017-22. Sara Magliacane (VU), Logics for causal inference under uncertainty
2017-23. David Graus (UVA), Entities of Interest — Discovery in Digital Traces
2017-24. Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
2017-25. Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
2017-26. Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
2017-27. Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
2017-28. John Klein (VU), Architecture Practices for Complex Contexts
2017-29. Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT
2017-30. Wilma Latuny (UvT), The Power of Facial Expressions
2017-31. Ben Ruijl (UL), Advances in computational methods for QFT calculations
2017-32. Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
2017-33. Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
2017-34. Maren Scheffel (OU), The Evaluation Framework for Learning Analytics

2017-35. Martine de Vos (VU), Interpreting natural science spreadsheets
2017-36. Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
2017-37. Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
2017-38. Alex Kayal (TUD), Normative Social Applications
2017-39. Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
2017-40. Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
2017-41. Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
2017-42. Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
2017-43. Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
2017-44. Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
2017-45. Bas Testerink (UU), Decentralized Runtime Norm Enforcement
2017-46. Jan Schneider (OU), Sensor-based Learning Support
2017-47. Jie Yang (TUD), Crowd Knowledge Creation Acceleration
2017-48. Angel Suarez (OU), Collaborative inquiry-based learning
2016-01. Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
2016-02. Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
2016-03. Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
2016-04. Laurens Rietveld (VU), Publishing and Consuming Linked Data
2016-05. Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
2016-06. Michel Wilson (TUD), Robust scheduling in an uncertain environment
2016-07. Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
2016-08. Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
2016-09. Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
2016-10. George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
2016-11. Anne Schuth (UVA), Search Engines that Learn from Their Users
2016-12. Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
2016-13. Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
2016-14. Ravi Khadka (UU), Revisiting Legacy Software System Modernization
2016-15. Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
2016-16. Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
2016-17. Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
2016-18. Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
2016-19. Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
2016-20. Daan Odijk (UVA), Context & Semantics in News & Web Search
2016-21. Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
2016-22. Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems

2016-23. Fei Cai (UVA), Query Auto Completion in Information Retrieval
2016-24. Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
2016-25. Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
2016-26. Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
2016-27. Wen Li (TUD), Understanding Geo-spatial Information on Social Media
2016-28. Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
2016-29. Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
2016-30. Ruud Mattheij (UvT), The Eyes Have It
2016-31. Mohammad Khelghati (UT), Deep web content monitoring
2016-32. Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
2016-33. Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example
2016-34. Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
2016-35. Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
2016-36. Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
2016-37. Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
2016-38. Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
2016-39. Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
2016-40. Christian Detweiler (TUD), Accounting for Values in Design
2016-41. Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
2016-42. Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
2016-43. Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
2016-44. Thibault Sellam (UVA), Automatic Assistants for Database Exploration
2016-45. Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
2016-46. Jorge Gallego Perez (UT), Robots to Make you Happy
2016-47. Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
2016-48. Tanja Buttler (TUD), Collecting Lessons Learned
2016-49. Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
2016-50. Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains
2015-01. Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
2015-02. Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
2015-03. Twan van Laarhoven (RUN), Machine learning for network data
2015-04. Howard Spoelstra (OUN), Collaborations in Open Learning Environments
2015-05. Christoph Bösch (UT), Cryptographically Enforced Search Pattern Hiding

2015-06. Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
2015-07. Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis
2015-08. Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
2015-09. Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems
2015-10. Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning
2015-11. Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins
2015-12. Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks
2015-13. Giuseppe Procaccianti (VU), Energy-Efficient Software
2015-14. Bart van Straalen (UT), A cognitive approach to modeling bad news conversations
2015-15. Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation
2015-16. Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork
2015-17. André van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs
2015-18. Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories
2015-19. Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners
2015-20. Lois Vanhée (UU), Using Culture and Values to Support Flexible Coordination
2015-21. Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning
2015-22. Zhemin Zhu (UT), Co-occurrence Rate Networks
2015-23. Luit Gazendam (VU), Cataloguer Support in Cultural Heritage
2015-24. Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation
2015-25. Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection
2015-26. Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure
2015-27. Sándor Héman (CWI), Updating compressed colomn stores
2015-28. Janet Bagorogoza (TiU), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO
2015-29. Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains
2015-30. Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning
2015-31. Yakup Koç (TUD), On the robustness of Power Grids
2015-32. Jerome Gard (UL), Corporate Venture Management in SMEs
2015-33. Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources
2015-34. Victor de Graaf (UT), Gesocial Recommender Systems
2015-35. Jungxao Xu (TUD), Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction
2014-01. Nicola Barile (UU), Studies in Learning Monotone Models from Data
2014-02. Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method
2014-03. Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions
2014-04. Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation
2014-05. Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability

2014-06. Damian Tamburri (VU), Supporting Networked Software Development
2014-07. Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior
2014-08. Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints
2014-09. Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language
2014-10. Ivan Salvador Razo Zapata (VU), Service Value Networks
2014-11. Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support
2014-12. Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
2014-13. Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
2014-14. Yangyang Shi (TUD), Language Models With Meta-information
2014-15. Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
2014-16. Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
2014-17. Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
2014-18. Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations
2014-19. Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
2014-20. Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
2014-21. Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments
2014-22. Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training
2014-23. Eleftherios Sidirourgos (UvA/CWI), Space Efficient Indexes for the Big Data Era
2014-24. Davide Ceolin (VU), Trusting Semi-structured Web Data
2014-25. Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction
2014-26. Tim Baarslag (TUD), What to Bid and When to Stop
2014-27. Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
2014-28. Anna Chmielowiec (VU), Decentralized k-Clique Matching
2014-29. Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software
2014-30. Peter de Cock (UvT), Anticipating Criminal Behaviour
2014-31. Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support
2014-32. Naser Ayat (UvA), On Entity Resolution in Probabilistic Data
2014-33. Tesfa Tegegne (RUN), Service Discovery in eHealth
2014-34. Christina Manteli (VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
2014-35. Joost van Ooijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach
2014-36. Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models
2014-37. Maral Dadvar (UT), Experts and Machines United Against Cyberbullying
2014-38. Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing.
2014-39. Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
2014-40. Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education

2014-41. Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text
2014-42. Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
2014-43. Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments
2014-44. Paulien Meesters (UvT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.
2014-45. Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
2014-46. Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
2014-47. Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval
2013-01. Viorel Milea (EUR), News Analytics for Financial Decision Support
2013-02. Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
2013-03. Szymon Klarman (VU), Reasoning with Contexts in Description Logics
2013-04. Chetan Yadati (TUD), Coordinating autonomous planning and scheduling
2013-05. Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
2013-06. Romulo Goncalves (CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
2013-07. Giel van Lankveld (UvT), Quantifying Individual Player Differences
2013-08. Robbert-Jan Merk (VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
2013-09. Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications
2013-10. Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.
2013-11. Evangelos Pournaras (TUD), Multi-level Reconfigurable Self-organization in Overlay Services
2013-12. Marian Razavian (VU), Knowledge-driven Migration to Services
2013-13. Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
2013-14. Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning
2013-15. Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
2013-16. Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation
2013-17. Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid
2013-18. Jeroen Janssens (UvT), Outlier Selection and One-Class Classification
2013-19. Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling
2013-20. Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval
2013-21. Sander Wubben (UvT), Text-to-text generation by monolingual machine translation
2013-22. Tom Claassen (RUN), Causal Discovery and Logic
2013-23. Patricio de Alencar Silva (UvT), Value Activity Monitoring
2013-24. Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning
2013-25. Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
2013-26. Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning
2013-27. Mohammad Huq (UT), Inference-based Framework Managing Data Provenance
2013-28. Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience
2013-29. Iwan de Kok (UT), Listening Heads
2013-30. Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support

2013-31. Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications
2013-32. Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
2013-33. Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere
2013-34. Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search
2013-35. Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction
2013-36. Than Lam Hoang (TUe), Pattern Mining in Data Streams
2013-37. Dirk Börner (OUN), Ambient Learning Displays
2013-38. Eelco den Heijer (VU), Autonomous Evolutionary Art
2013-39. Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
2013-40. Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games
2013-41. Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
2013-42. Léon Planken (TUD), Algorithms for Simple Temporal Reasoning
2013-43. Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts
2012-01. Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
2012-02. Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
2012-03. Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
2012-04. Jurriaan Souer (UU), Development of Content Management System-based Web Applications
2012-05. Marijn Plomp (UU), Maturing Interorganisational Information Systems
2012-06. Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
2012-07. Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
2012-08. Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories
2012-09. Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
2012-10. David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
2012-11. J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
2012-12. Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
2012-13. Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
2012-14. Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
2012-15. Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
2012-16. Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
2012-17. Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
2012-18. Eltjo Poort (VU), Improving Solution Architecting Practices
2012-19. Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
2012-20. Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
2012-21. Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval

2012-22. Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
2012-23. Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
2012-24. Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
2012-25. Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
2012-26. Emile de Maat (UVA), Making Sense of Legal Text
2012-27. Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
2012-28. Nancy Pascall (UvT), Engendering Technology Empowering Women
2012-29. Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
2012-30. Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
2012-31. Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
2012-32. Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
2012-33. Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
2012-34. Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
2012-35. Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
2012-36. Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
2012-37. Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
2012-38. Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
2012-39. Hassan Fatemi (UT), Risk-aware design of value and coordination networks
2012-40. Agus Gunawan (UvT), Information Access for SMEs in Indonesia
2012-41. Sebastian Kelle (OU), Game Design Patterns for Learning
2012-42. Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
2012-43. Withdrawn
2012-44. Anna Tordai (VU), On Combining Alignment Techniques
2012-45. Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
2012-46. Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
2012-47. Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
2012-48. Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
2012-49. Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
2012-50. Steven van Kervel (TUD), Ontologogy driven Enterprise Information Systems Engineering
2012-51. Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching
2011-01. Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models
2011-02. Nick Tinnemeier (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
2011-03. Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems
2011-04. Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference

2011-05. Bas van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.
2011-06. Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage
2011-07. Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction
2011-08. Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues
2011-09. Tim de Jong (OU), Contextualised Mobile Media for Learning
2011-10. Bart Bogaert (UvT), Cloud Content Contention
2011-11. Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective
2011-12. Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining
2011-13. Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling
2011-14. Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
2011-15. Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval
2011-16. Maarten Schadd (UM), Selective Search in Games of Different Complexity
2011-17. Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness
2011-18. Mark Ponsen (UM), Strategic Decision-Making in complex games
2011-19. Ellen Rusman (OU), The Mind's Eye on Personal Profiles
2011-20. Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach
2011-21. Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems
2011-22. Junte Zhang (UVA), System Evaluation of Archival Description and Access
2011-23. Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media
2011-24. Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
2011-25. Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics
2011-26. Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots
2011-27. Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
2011-28. Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure
2011-29. Faisal Kamiran (TUE), Discrimination-aware Classification
2011-30. Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions
2011-31. Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
2011-32. Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science
2011-33. Tom van der Weide (UU), Arguing to Motivate Decisions
2011-34. Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
2011-35. Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
2011-36. Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
2011-37. Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
2011-38. Nyree Lemmens (UM), Bee-inspired Distributed Optimization

2011-39. Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
2011-40. Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
2011-41. Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
2011-42. Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
2011-43. Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
2011-44. Boris Reuderink (UT), Robust Brain-Computer Interfaces
2011-45. Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
2011-46. Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
2011-47. Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
2011-48. Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
2011-49. Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

VU VRIJE UNIVERSITEIT AMSTERDAM