Doctoral theses at NTNU, 2024:349

Doctoral thesis

Anders Eivind Bråten

Synthesis of a Conceptual Architecture for Cognitive IoT Device Management

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Anders Eivind Bråten

Synthesis of a Conceptual Architecture for Cognitive IoT Device Management

Thesis for the Degree of Philosophiae Doctor

Trondheim, September 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ISBN 978-82-326-8288-1 (printed ver.)
ISBN 978-82-326-8287-4 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2024:349

Printed by NTNU Grafisk senter

Abstract

The Internet of Things (IoT) is characterised by diverse, dynamic and distributed deployments of sensors, actuators, tools and gadgets. These devices often have limited resources like energy, memory, and processing power. Moreover, they are frequently located in unstable environments where conditions change over time.

The limitations of IoT nodes often require them to delegate resource-intensive tasks to a device management platform that can act on their behalf and ensure optimal performance. However, in addition to the limited resources of the devices and the context in which they operate, these platforms also have to deal with challenges related to the scale of the deployment, the network topology, and the type of management that is performed. The size and complexity of IoT deployments often require device management platforms to operate autonomously.

In this thesis, we study and explore how to autonomously manage a multitude of distinct and constrained IoT equipment that operate in ever-changing contexts. To solve this problem, we synthesised a generalised conceptual architecture for autonomous management of IoT devices deployed in non-stationary environments. The conceptual architecture is based on the theoretical foundation of two distinct research fields: IoT device management and cognitive architectures.

The research was guided by the design science research methodology and followed both a bottom-up and top-down approach. Initially, we conducted six case studies, where we focused on designing specific parts of the proposed model. Following that, we carried out a structured literature review, where we analysed architectural models from 32 case studies in the field of autonomous IoT device management.

Based on the insights that we gathered through the work of this thesis, we synthesised a conceptual architecture for cognitive IoT device management that fulfilled our research goal. The model describes adaptive behaviour on three levels. On the highest level, there are three system components: The device, the device manager and the system manager. On the second level, we have five distinct adaptation components that are contained within the system components. They are responsible for handling perception, action, adaptation process, declarative knowledge and procedural knowledge, respectively. The adaptation process component is further detailed in five types of adaption processes, namely, monitor, analyse, learn, predict and plan. On the lowest levels, we find adaptation mechanisms and triggers, which describe the data flow between the components.

Apart from the conceptual architecture for cognitive IoT device management, this thesis has three additional contributions. First, we present a comprehensive taxonomy of adaptation mechanisms for cognitive IoT device management. Second, we describe a model of cognitive planning. Third, we provide a list of best practices to guide the design and implementation of cognitive IoT device management platforms, along with recommendations for when and how to apply them. These contributions will be useful for those who aim to develop such solutions.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* in Information Security and Communication Technology at the Faculty of Information Technology and Electrical Engineering (IE), Norwegian University of Science and Technology (NTNU). The research presented here was conducted at the Department of Information Security and Communication Technology under the supervision of Professor Frank Alexander Kraemer and the co-supervision of Professor Peter Herrmann and Associate Professor David Palma.

The thesis is composed of two parts. Part one provides the main results from the thesis, while part two is a paper collection. The included papers have been published in scientific conferences or journals, and have been reformatted to have consistent formatting within the thesis. Thus, they may deviate visually from the published versions.

Disclaimer

This thesis was written without the assistance of ChatGPT or similar tools, but Grammarly was used to some degree to check spelling and grammar.

Acknowledgements

Above all, I would like to thank Frank Alexander Kraemer for the guidance, dedication, support, patience and constructive feedback I have received during this PhD period. Your expertise on the subject of communication technology, research methodology and academic writing has been a great inspiration and paramount for the completion of this thesis. Next, I want to show my appreciation to Peter Hermann for the good advice I have received during the work on this thesis. We have had many fruitful conversations on IoT and other topics these years. I am also grateful for the contributions provided by David Palma. Your knowledge, insights and feedback have been invaluable concerning the papers we have written together and thus for this thesis. I have thoroughly enjoyed working with all of you these years.

Further, I want to acknowledge my colleagues at the Department of Information Security and Communication Technology at NTNU for their kindness and generosity during my time there. In particular, I'd like to thank Yuming Jiang and Bjarne Helvik for valuable feedback on my midterm report, and Mona Nordaune and Pål Sturla Sæther for assisting me with any practical issues that arose during this period. I'd also like to extend a special thanks to my office mates during this time, namely Doreid, Katrien, Nattachart, Ruxandra, Murad, Faiga and Sonu. Our discussions on various topics, although not necessarily related to my thesis, were a great stress reliever and helped me stay sane throughout this process.

One particular group of friends has been instrumental during my academic career. In order of appearance, I would like to send my thanks to Harald, Finn Olav, Jardar, Stein Eldar, Tord F, Pernille, Rune, Tord R, Morten and Jenny. Since we first met at Steinan Studentby in the late 90s, we have first and foremost shared a lot of fun, but also ideas, knowledge, insights and support. I am the fifth member of the group to obtain a Ph.D. That is quite an accomplishment for a group of ragtag friends who originally were bound together by the love of role-playing games.

Finally, I express my gratitude to my family who has always been there for me. First, and foremost, I want to thank my beloved wife, Oddrun. I could not have made it without your exceptional support and endless patience throughout this process. We are still a great team. Next, I want to thank my mother Gerd Inger, my father Arne, who sadly passed away in 2020, and my in-laws, Marit and Anders. Your trust in me has helped me stay strong when I needed it the most. I also want to thank my siblings, Jan Ivar, Geir Arne, and Lillian. Your encouragement was of great help to me throughout my journey. A very special thanks go to my amazing children, Brage, Idunn, and Jenny, who mean the world to me. You have always been a source of cheer and support, even during my most stressful periods.

Anders Eivind Bråten
Trondheim, August 2024

List of Papers

Paper A

Dirk Ahlers, Frank Alexander Kraemer, Anders Eivind Braten, Xiufeng Liu, Fredrik Anthonisen, Patrick Driscoll, and John Krogstie, “Analysis and visualization of urban emission measurements in smart cities”. In: *Proceedings of the 21st International Conference on Extending Database Technology (EDBT)*, Vienna, Austria, 2018, March, pp. 682-685.

Paper B

Anders Eivind Braten, Nattachart Tamkittikhun, Frank Alexander Kraemer, and Doreid Ammar, “Towards cognitive device management: A testbed to explore autonomy for constrained IoT devices”. In: *Simon Mayer, Stefan Schneegass, Bernhard Anzengruber, Alois Ferscha, Gabriele Anderst-Kotsis, Joe Paradiso (Ed.): Proceedings of the Seventh International Conference on the Internet of Things*, Linz, Austria 2017, October, pp. 1-2

Paper C

Frank Alexander Kraemer, Doreid Ammar, Anders Eivind Braten, Nattachart Tamkittikhun, and David Palma, “Solar energy prediction for constrained IoT nodes based on public weather forecasts”. In: *Simon Mayer, Stefan Schneegass, Bernhard Anzengruber, Alois Ferscha, Gabriele Anderst-Kotsis, Joe Paradiso (Ed.): Proceedings of the Seventh International Conference on the Internet of Things*, Linz, Austria 2017, October, pp. 8-15

Paper D

Frank Alexander Kraemer, David Palma, Anders Eivind Braten and Doreid Ammar, “Operationalizing solar energy predictions for sustainable, autonomous IoT device management”. In: *IEEE Internet of Things Journal* (2020), 7(2), pp. 11803-11814

Paper E

Anders Eivind Braten and Frank Alexander Kraemer, “Towards cognitive IoT: Autonomous prediction model selection for solar-powered nodes”. In: *IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 118-125.

Paper F

Anders Eivind Braten, Frank Alexander Kraemer, and David Palma, “Adaptive, correlation-based training data selection for IoT device management”. In: *Mohammad Alsmirat, Yaser Jararwe (Ed.): 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Granada, Spain, 2019, October, pp. 169-176.

Paper G

Anders Eivind Braten, Frank Alexander Kraemer, and David Palma, “Autonomous IoT device management systems: Structured review and generalized cognitive model”. In: *IEEE Internet of Things Journal* (2021), 8(6), pp. 4275-4290

Contents

Chapter 1

Introduction and Motivation

The Internet of Things (IoT) is characterised by diverse, dynamic and distributed deployments of devices. This means that IoT systems must be able to handle a myriad of distinct sensors, actuators, tools and gadgets that are connected in different ways and are located in a large variety of environments. As shown by Siddiqui et al. [1], this raises concerns on an architectural level since traditional monolithic software architectures are not designed to cope with this variance of scalability, deployability, adaptability, and maintainability.

For *constrained* IoT devices, which typically have limited resources like energy, memory and processing power, this challenge is even greater. Nikoui et al. [2] stress the importance of taking these constraints into account on an architectural level when developing wireless technologies aimed at the IoT. To complicate things even further, such equipment is often located in environments where conditions are unstable and change over time. This makes it hard to predict future events, which in turn complicates how to plan for perpetual operation.

The constraints often require IoT devices to delegate resource-intensive tasks to a device management platform located either on the edge or in the cloud that can act on their behalf and ensure performance requirements. The purpose of such platforms is twofold: To collect, aggregate and filter the data from these gadgets, and to monitor and control their operation. The latter is critical since it allows remote upgrades and operational adjustments. This can increase functionality, reliability and dependability, and even prevent waste of system resources, as demonstrated by Jantunen et al. [3]. However, the size and complexity of IoT deployments often require device management platforms to operate autonomously. This raises a series of challenges related to the architecture, that is, the overall system behaviour that emerges from the organisation of the components in the system.

In traditional management theory, autonomy is defined as “the degree to which one may make significant decisions without the consent of others” [4]. When transferred to the IoT, autonomous management implies the use of independent equipment that has the capability and authority to act on an unexpected event or a change in context, i.e., they can govern themselves. Through the work of this thesis, we have found that autonomous IoT device management (AIDM) can be categorised into three levels of autonomous behaviour which reflect the architectural complexity. These are autonomic, context-aware and cognitive IoT device management. We will discuss this distinction in detail in Chapter 2.

In many device management platforms the actual hardware is represented by agents, also known as digital twins. These agents are usually guided by a common strategy, objective or policy enacted by a central manager. In those cases, AIDM implies a division of tasks, where a central manager controls the strategic direction of the system as a whole, and an agent decides how to reach the goals of the system, given the individual context and constraints of a particular device [5].

The goal of autonomous IoT device management is to achieve maintenance-free systems, as explained by Sifakis et al. [6]. Interest in this research field has increased in recent years. Although some aspects have been extensively researched, such as automatically connecting different types of equipment [7] and supporting communication between them [8], other areas still lack research. For instance, how to guarantee a stable and reliable operation of devices that run in a heterogeneous and unstable environment once they are deployed and connected

We have also found that current research in this domain often is sporadic and rather unstructured with proposed solutions typically being specialised toward solving a single use case, with few attempts at discussing the architectural challenges of device management in general terms. In particular, the architecture of the employed components and mechanisms that allow *cognitive* behaviour is rarely discussed, and alternative approaches are seldom considered. This implies that cognitive IoT device management is still a maturing field in need of a standardised and unified architecture that can support researchers and practitioners in their work.

1.1 Challenges

Through the work of this thesis, we have found that autonomous IoT device management is a composite problem that consists of challenges related to the resources that the gadgets possess, the scale of the deployment, the network topology, the operational context, and the type of management that is employed. In the following, we will examine these five challenges in more detail, from an architectural point of view.

The challenge of constrained resources. Constrained devices are limited in terms of available energy, memory, processing capability or communication [9, 10]. They also lack contextual information, such as weather forecasts or local traffic patterns. Because of these limitations, they are unable to analyse the current situation, learn from experience, predict future events, or plan corrective actions effectively. To ensure optimal performance, tasks that require significant resources are often moved to nodes that have better access to these resources [11, 12, 13]. This raises the following question: **How can devices with limited resources offload tasks, such as learning, predicting, and planning, to equipment that has better access to those resources?**

The challenge of large-scale deployments. Traditional sensing applications typically employ high-quality components to measure a specific effect within a limited area. However, the IoT infrastructure enables the deployment of numerous low-cost sensors that can provide high-density coverage over a much larger area, such as an entire city [14]. The sheer number of gadgets means that maintaining them throughout their full lifecycle, i.e., the processes of planning, configuring, deploying, running, repairing, and finally recycling them, is a major challenge [15, 16]. For obvious reasons, maintaining such deployments manually is not a feasible option. Another question is then: **How can large-scale deployments of IoT devices operate autonomously with a minimum of human intervention?**

The challenge of network topologies. When deploying a wireless sensor network (WSN) that consists of wireless devices, deciding which topology to implement is a key design decision, as this defines how the devices are connected, how they communicate and how they collaborate [17]. This decision will in turn influence the robustness of the system [18]. During the work of this thesis, we have found that there are in essence three types of topologies employed in IoT: The distributed topology, where each node is responsible for all operative and adaptive actions; the clustered topology, where two or more parent nodes share the responsibility for managing adaptive behaviour for separate subgroups of devices; and the star topology, where a single central node manages the adaptive processes for all devices. Each of these topologies comes with different strategies when it comes to how the nodes handles their operation.

Topology requirements for WSNs depend on both technical, environmental and contextual conditions. For instance, the topology of a few nodes placed in proximity of each other in a stable or homogeneous environment might differ a great deal from that of a system consisting of many nodes distributed over a large area that operates in a volatile or heterogeneous environment. If we look at how devices in a wireless sensor network collaborate, the latter case is of particular interest. Nodes that are operating under different conditions might benefit from sharing this knowledge. This is a concept in IoT that is known as 'edge knowledge' [19]. From this challenge, we ask: **How can constrained devices share relevant knowledge?**

The challenges of operational context. As shown in the examples above, IoT equipment can be deployed in different settings. Many devices operate in an environment where conditions are different from node to node or change over time, i.e., a context that is dynamic in the spatial or temporal dimension. This can be especially challenging for energy-harvesting equipment. Thus, when operating in a context of high variance and volatility each device must have the ability to adapt its operation individually, as demonstrated by Zheng et al. [20].

We identified two distinct types of dynamic environments through the work of this thesis. In stationary environments, we deal with variables where the values follow a specific distribution, known or unknown to us. This means we can usually predict any changes in variation stochastically. Non-stationary environments are on the other hand characterised by unstable conditions and distributions that change over time. This means that an action applied to devices under different conditions can yield completely different outcomes. In turn, this makes it challenging for a device manager to plan and execute proper corrective actions to a previously unseen event [21]. From this challenge, we identify another question: **How can constrained devices adapt to varying conditions that they experience in their environment?**

The challenge of resource management. The purpose of a system is often closely connected to the type of management that is performed. In this thesis, we have identified three management tasks that are commonly employed by autonomous device managers, namely network management, application management and resource management. In short, network management handles the infrastructure of a wireless sensor network, application management oversees the applications that run on the nodes and resource management is concerned with maintaining the operation itself. We will go into more detail about these three types of device management in Chapter 3.

Through our research, we have discovered that managing resources to ensure high data quality and perpetual operation of constrained devices can be particularly challenging. This problem is interconnected with the challenges we have previously discussed. For example, constrained devices operating in a non-stationary environment often lack the resources needed to make necessary plans for preserving or conserving their energy [22, 23]. Additionally, the chosen network topology and deployment scale are linked to factors such as network orchestration and internal complexity, which can impact how resources are managed and allocated among the nodes in the network [24]. The challenges related to resource management in this context lead to yet another question: **How can constrained IoT devices handle their resource management, given the limited resources they have at their disposal?**

1.2 Use Case: Solar Energy Harvesting Devices

To explore and investigate the challenges identified above on an architectural level, we have performed six case studies related to the use of solar energy harvesting and energy planning for constrained IoT devices.

We selected this particular use case for three main reasons: Firstly, while working on Paper A, we discovered that when deploying solar-powered IoT devices in environments where the amount of energy available for harvesting varies significantly, they must be able to plan their energy budget. By doing so, they can determine how much energy they can safely consume given the amount of energy that is available to harvest from the environment. Secondly, due to the location of the nodes and the environmental variance between them, each device must have the ability to plan according to its context. For instance, the direction of the solar panels will significantly impact the amount of energy that can be harvested. Thirdly, the volatile and non-stationary weather patterns in the northwestern part of Norway are a good use case for exploring how a cognitive device manager copes with events that are hard to predict.

Together, these factors introduce the necessity of individual learning in addition to reasoning and planning. An architecture aimed at solving the problem of cognitive and perpetual management for constrained IoT devices operating under these conditions is thus a good candidate for realising the research goal, which is further described in Section 1.3.

1.3 Research goal

As explained in the introduction, describing a general architecture for ensuring perpetual operation for large-scale deployments of constrained IoT devices placed in non-stationary environments is still an open challenge. This thesis aims to study how a central manager can assist constrained devices in adapting to new situations that they may face at run-time, independently and based on their context, from an architectural perspective.

Thus we formulate the following research goal:

Research goal

Synthesise and describe a generalised architectural model for autonomous and perpetual management of IoT devices deployed in a non-stationary environment.

We solve the research goal by following the design science research methodology detailed in Chapter 5 and answer the research questions described in Section 1.4.

1.4 Research questions

To realise the research goal and guide the research, we formulate a set of research questions (RQs), based on the questions raised in Section 1.1. RQ 1 addresses the investigation of the artefact design goal, in the context of the problem space, while RQ 2 to 5 address the artefact design goal in the context of the knowledge space.

- **RQ 1: Identify challenges**
  What are the contextual and architectural challenges that must be addressed to realise the research goal?

- **RQ 2: Identify system components and adaptation components**
  Which system components and adaptation components are required to reach the research goal?

- **RQ 3: Identify adaptation mechanisms**
  Which adaptation mechanisms can be used to achieve the research goal?

- **RQ 4: Identify patterns**
  Which patterns can we identify in existing autonomous architectural models, in industry or recent research, that can guide us toward the research goal?

- **RQ 5: Identify best practices**
  Which best practices can we identify by analysing existing implementations of autonomous architectures, in industry or recent research, that can help us reach the research goal?

We examine RQ 1 in the six case studies in Papers A through F and in the structured literature review in Paper G. For RQ 2 and RQ 3, we adopt both a bottom-up and top-down approach. Firstly, we analyze the architectural models explained in our six case studies. Secondly, we scrutinize the existing architectural models and descriptions in the recent literature of self-adaptable architectures and platforms, which can tackle the challenges identified by RQ 1. We answer RQ 4 and RQ 5 through the structured literature review, taking into account the insights gathered by answering RQ 1, RQ 2 and RQ 3.

1.5 Design Problems

Answering knowledge questions requires data. To produce this data, we defined a set of design problems (DPs) which in turn resulted in a set of corresponding design artefacts. These were aimed at helping us explore and analyse architectural aspects and challenges raised by the research questions. The insights gathered through the analysis then helped us synthesise knowledge that was put to use when designing the generalised architecture for cognitive IoT management.

In the following, we briefly describe the design problems, the corresponding design artefacts and how the insights contributed towards the research questions and the research goal. Note that all case studies are based on constrained, solar-powered IoT devices located in a non-stationary environment.

- **DP 1: Design a platform that supports data collection and analysis.** This artefact, detailed in Paper A, was used to study the operation of collecting and analysing CO2 measurements from a wireless sensor network. This study gave valuable insights into the architectural challenges on a contextual level.

- **DP 2: Design a platform that supports resource offloading.** This artefact was the focus of Paper B. In this study, we described the setup for a platform that collects both environmental and contextual data from a set of IoT nodes. Further, we explored the case of outsourcing planning to a central manager located in the cloud, which could remotely override the sensing cycles of the devices. This validated the concept of architectural resource offloading.

- **DP 3: Design a mechanism that supports resource optimisation.** This artefact is described in Paper C, and is a continuation of Paper B. Here we created a pipeline for exploring how different machine-learning models can support energy buffer predictions for solar-powered devices. The use case validated the underlying mechanisms for learning and prediction that need to be in place to design an architectural model for cognitive IoT device management.

- **DP 4: Design a mechanism that supports self-tuning of learning processes.** In this case study, found in Paper D, we implemented algorithms that helped us explore different mechanisms that can support the autonomous tuning of machine-learning models for predicting future energy intake. This artefact helped us to understand the mechanisms needed for self-tuning on an architectural level.

- **DP 5: Design a mechanism that supports self-management when deploying a new system.** In paper E, we designed an algorithm that could autonomously switch between different prediction models, to increase prediction accuracy in the first days after deployment. This case study explored an architectural pattern that can provide self-management for agents that mirror IoT equipment deployed in a new WSN and validated the architectural concept.

- **DP 6: Design a mechanism that supports self-management when deploying new devices into an existing IoT deployment.** The case study detailed in Paper F demonstrates an autonomous learning data selection algorithm, which improves prediction accuracy by using the principle of transfer learning. The case study provided knowledge about how to organise system components in an IoT architecture and validated an architectural pattern that can provide self-management capabilities for agents when new nodes are introduced into a deployment of existing IoT devices.

1.6 Scope

The main focus of this thesis is on the general architectural challenges that are associated with the implementation of cognitive IoT device management platforms, as explained in Section 1.3. These challenges were explored both from a bottom-up approach through a series of case studies where we simulated isolated parts of the architecture, and from a top-down approach through a structured literature review where we analysed the architecture of 32 case studies found in research. This resulted in the design of a generalised *conceptual* architectural model of cognitive IoT device management. Building or deploying an actual implementation of the complete architecture is therefore out of the scope of the thesis.

A key topic in IoT is the edge-cloud continuum, which refers to managing distributed computing and network infrastructures at the edge of the cloud. This involves optimizing performance, resource usage, energy consumption, security and financial costs [25]. We acknowledge that these attributes are highly relevant to the topic of this thesis. Nevertheless, since we aim to design a technology-agnostic generalised architectural model that in theory can be applied at any level in an ecosystem of edge devices, we did not study this type of system further.

End-of-life commissioning is a crucial aspect in the overall management process of an IoT system [26]. IoT devices powered by batteries contain chemicals and toxins that can become a hazard if not properly recycled when they reach the end of their lifecycle [27]. This is particularly true for outdoor equipment placed in volatile conditions. However, since this is mostly a manual task and we focus on the autonomous part of IoT operation and management in this thesis, we chose to not include this phase in the study.

Research within autonomous IoT device management overlaps with Industry 4.0 in several aspects, both concerning theory and terminology. However, there is an important distinction between the two areas of research. While AIDM aims to achieve maintenance-free perpetual operation for the sensor nodes themselves, Industry 4.0 usually employs sensory instruments in conjunction with artificial intelligence to avoid interruptions in the production lines and optimise the production run time [28]. In Industry 4.0 this is known by the term ‘predictive maintenance’. Thus, even though these domains share many similarities, the findings in this thesis may not be generalisable into the field of Industry 4.0.

1.7 Contributions

Through the work of six case studies and one structured literature review, this thesis provides insights that can guide the design and implementation of platforms for cognitive IoT device management. In particular, the thesis contributes to this field on the following accounts:

- **A generalised architectural model for cognitive management of IoT devices.** The proposed model is essentially a blueprint of a cognitive, service-oriented architecture that describes the interaction between the stored knowledge in a system and the adaptation processes that are needed to provide cognitive behaviour, i.e., reasoning, learning and planning. The model consists of three different system components, namely *device*, *device manager* and *system manager*. Within these containers, we have identified five adaptation components. These are ‘Perception’, ‘Action’, ‘Adaptation process’, ‘Declarative knowledge’ and ‘Procedural knowledge’. The adaptation process is again divided into five categories: ‘Monitor’, ‘Analyse’, ‘Learn’, ‘Predict’ and ‘Plan’.

  Since the notation is general, it has the potential to progress the implementation of *any* platform that requires cognitive and autonomous decision-making. Energy budgeting for solar-powered IoT devices is just one of many possible use cases.

- **A taxonomy of adaptation mechanisms for autonomous IoT device management.** This taxonomy describes the full range of adaptation mechanisms that are used to support autonomous device management in IoT and how they relate to each other, on three levels. First, it shows that cognitive device management incorporates components that include adaptation mechanisms to observe an event, analyse the situation, reason about the implications, predict an outcome and plan a corrective action, if necessary. Second, it categorises the different types of reasoning mechanisms into model-driven, semantic and data-driven mechanisms, which reflect the underlying principles that are used to infer situational awareness. Third, within those categories, it identifies eight different types of implementation.

  Our taxonomy can help information architects and developers decide what type of mechanism to include and where to place it in their architecture, based on their particular challenges.

- **A model of cognitive planning.** This model is a refinement of Vernon’s Cognitive Cycle, first presented in [29]. In the model, we introduce the planning component and place it in the centre. It consists of an autonomic and a situation-aware subsystem, which are executed through two separate control loops. The autonomic loop registers an event and initiates a response per its active policy, which ensures that the device can operate even if the connection to the rest of the system is lost or broken. The adaptive loop analyses the event and changes the policy if the situation calls for adaptation. It also supports continuous learning from experience, based on the stimuli provided by the autonomic loop. The distinction between the two sub-systems increases both the autonomy and the robustness of the system.

This model can help information architects and developers to get a better understanding of the planning process when designing top-level architectures of cognitive systems.

- **A list of best practices to guide the design and implementation of cognitive IoT device management platforms.** This is a description of five best practices for designing cognitive IoT device management systems coupled with recommendations for when and how to apply them. These practices can help information architects and developers when they need to make architectural decisions regarding such systems.

In addition to the main contributions listed above, the case studies have made individual advances towards the studied domain.

- In Paper A, we demonstrated a quick and flexible way to prototype a platform for data management and data analytics from wireless sensor nodes.
- In Paper B, we demonstrated that constrained devices can outsource research-intensive machine learning to a manager located in the cloud and that it is possible to estimate the energy consumption of applications running on a constrained device with high precision.
- In Paper C, we ran a simulation which showed that predicting solar energy is possible even with limited access to data, progressively improving as the system runs.
- In Paper D, we ran a simulation that showed how individual sensor nodes can be trained autonomously by employing automatic tuning of learning models. Our results showed that this approach can improve the median prediction scores by more than 20% compared to state-of-the-art predictors for IoT energy prediction.
- In Paper E, we developed a mechanism for autonomous prediction model selection which can mitigate the bootstrapping problem for constrained devices and help them stay in operation in periods when training data is missing.
- In Paper F, we developed a mechanism that uses a correlation algorithm to select appropriate training data for a given node. Our results showed that this approach can improve the accuracy of the predictions of a new node by 14%.

The novelty of each contribution is further detailed and discussed in Chapter 6 and in the respective papers.

1.8 Thesis structure

This thesis is divided into two parts. The first part includes seven chapters that detail the main results of the thesis. In Chapter 1, we introduce the problem domain, explain the motivation behind conducting the study and present the research goal. Chapter 2 provides background information, and Chapter 3 shows related work within IoT management from an evolutionary perspective. We then proceed to an overview of each of the publications that are part of the thesis in Chapter 4 before offering a description of the research methodology in Chapter 5. In Chapter 6, we present the results and show the contributions in each paper. Finally, we conclude the thesis in Chapter 7 with an overview of the main achievements and future directions.

Part two contains the seven main papers published as part of the thesis work and one auxiliary journal paper.

Chapter 2

Background

This thesis delves into the topic of autonomous IoT device management (AIDM), from an architectural viewpoint. The goal is to advance research on cognitive management of IoT devices by designing a model that depicts the interaction between adaptation components in an architectural component model, later presented in Chapter 6.

The conceptual architectural model proposed in this thesis is based on the theoretical foundation of two distinct research fields: IoT device management and cognitive architectures. Research on cognitive architectures is in turn rooted in the study of autonomic computing, cognitive science and big data analytics. However, it is also possible to see this domain as a blend of context-aware computing and cognitive computing. Figure 2.1 shows how cognitive IoT device management can be found in the cross-section of these research fields.

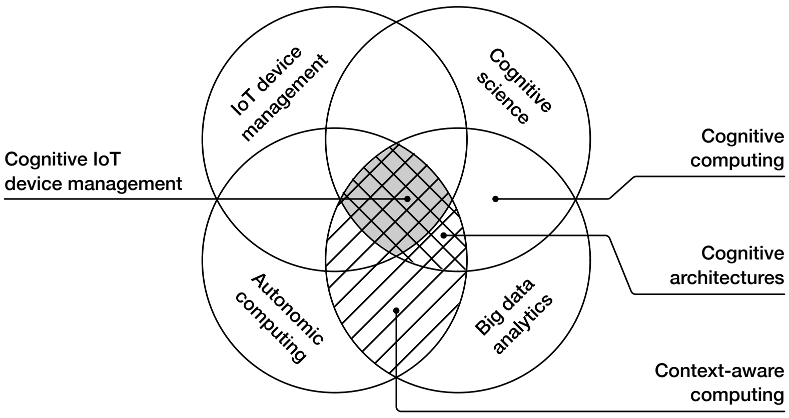

Figure 2.1: Venn-diagram showing how cognitive IoT device management can be found in the cross-section of IoT device management and cognitive architectures.

In the following sections, we will first briefly describe the relevant research fields. Next, we will present the characteristics of an autonomous system. Lastly, we will look at AIDM, on autonomic, context-aware and cognitive levels.

2.1 A Brief Description of Related Research Fields

As explained in Chapter 1, IoT device management aims to ensure optimal performance and perpetual operation of the managed IoT devices. Deployed IoT devices are usually managed by device management platforms, which provide various functionalities such as status monitoring, fault detection, system configuration, collection of performance data and operation control [8, 30]. They also enable data exchanges between the manager and managed devices [31].

Autonomic computing is the study of self-adaptive systems. These are systems that can adjust their behaviour through adaptation processes to reach a goal, without human intervention, even if they encounter unexpected events or changes in conditions [32, 33]. Adaptation processes, triggered by some internal or external stimuli, are responsible for deciding if there is a need for adaptation, which adaptation strategy to follow, and the best course of action [34].

Context-aware computing expands autonomic computing by also considering and analysing contextual data like time, identity, history, location and environment when making decisions [35]. Having access to such knowledge means they are usually better equipped to reason about the best way to adapt to the effect of the observed event. This can help the system to solve more complex tasks [36].

Cognitive computing builds on cognitive science and big data analytics [37]. According to Modha et al. [38], cognitive computing is concerned with the mechanisms that define cognitive behaviour in computer systems in general. Such behaviour can be modelled as reasoning, learning and planning mechanisms [39], which in turn can drive the adaptation processes.

Research on cognitive architectures is guided by theories and practices found in cognitive science and neuroscience [40]. In contrast to cognitive computing, the focus is on modelling how specific agents imbued with artificial intelligence can display intelligent behaviour [41]. Thus, it combines elements from autonomic, context-aware and cognitive computing. In this thesis, we adopt the cognitivist perspective within this field of research, which aims to capture the underlying functions of human cognition, such as reasoning, control, learning, memory, adaptivity, perception, and action at the computational level [42].

2.2 Autonomous Behaviour in IT Systems

Sifakis et al. [43] list five factors that are required to achieve full autonomy: 1) perception; 2) reflection; 3) goal management; 4) planning; and 5) self-adaptation. The characteristics of each factor are listed in Table 2.1.

Table 2.1

Factor	*Ability*
Perception	Interpret a stimuli from the environment
Reflection	Reason about the stimuli in the environmental context
Goal management	Decide if the current goal is still valid
Planning	Decide which action(s) to take to achieve the goal
Self-adaptation	Adjust the behaviour through learning and reasoning

The spectrum of possible solutions between no autonomy and full autonomy is quite wide, and most autonomous systems are located somewhere in between.

Often, the adaptive behaviour is a reflection of the complexity of the challenges that they encounter or the kind of problems they solve.

As mentioned in Chapter 1, we have found that AIDM can be categorised into autonomic, context-aware and cognitive IoT device management, based on the level of autonomous behaviour and architectural complexity. In the next three sections, we will look at the characteristics of each of these categories, from an architectural perspective.

2.3 Autonomic IoT Device Management

In neuroscience, an autonomic system is characterised by an automatic response to stimuli [44]. An example from nature is a dandelion that grows shorter and smaller buds as a response to mowing [45]. By this definition, autonomic IoT device management is the planning and executing a pre-defined or designated response of an IoT device that adapts to or mitigates the effect of an observed event.

Most autonomic device management systems are concerned with the internal state of the device. They are typically based on the feedback control loop principle [46] and perform basic dynamic adaptation based on high-level policies and pre-defined adaptive actions [47, 48].

Within IoT, many device management architectures employ the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) pattern, described by IBM Corporation in their 2006 white paper [49]. A general model of this adaptive feedback control loop is shown in Figure 2.2. This model enables self-adaptation through four steps: first, data is collected and monitored; second, the collected data is analyzed, which may trigger a need for adaptation; third, a plan is made, based on a set of adaptation rules; fourth, a corrective action is executed, if necessary. New knowledge can be integrated into the control loop by feeding the newly adapted state back into the system or by manipulating and expanding data sets and rules.

For systems that outsource device management to a central manager, it is possible to expand the model by applying two separate control loops: one for the managed subsystem (the device) and one for the managing subsystem (the device manager) as discussed by Arcaini et al. [50].

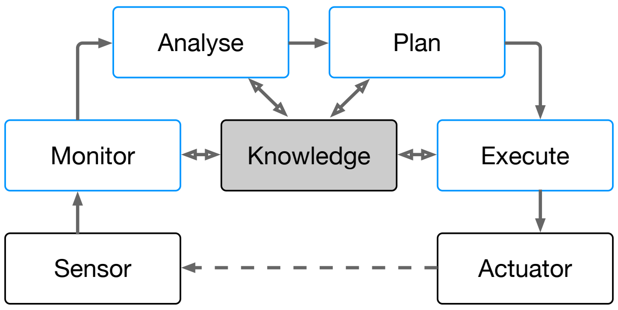

Figure 2.2: Model of the basic MAPE-K autonomic control loop. Adapted from [49]

During the work conducted for Paper G, we observed that 1) autonomic architectures are typically used to solve tasks that can be predicted either statistically or stochastically, i.e., changes that occur within known distributions; 2) a common strategy is to employ basic feedback control loops in combination with probabilistic methods, dynamic programming, knowledge graphs, semantic rules or fuzzy logic to adapt to changes in the environment; and 3) goal management, planning and adaptation are usually also automated processes.

An example of the autonomic pattern can be seen in the work of Romero-Garces et al. [51], where MAKE-K loops in combination with fuzzy logic are used to give a service robot in a warehouse self-adaptive properties. In fuzzy logic, pre-defined fuzzy sets and fuzzy rules stored in linguistics variables define the behaviour of the system. These rules are then used on the sets to infer a decision from the model, which again might cause the system to adapt to a new state. In systems that employ fuzzy logic, it is possible to expand the knowledge base by manipulating fuzzy sets or by applying new fuzzy rules. However, fuzzy systems usually lack the capability of recognising complex patterns that change over time.

2.4 Context-aware IoT Device Management

A common goal in autonomous IoT device management is to help systems adapt to new situations autonomously through self-management [32]. According to Kephart et al. [52] such systems are characterised by the ability to configure, optimise, heal and protect themselves. The response of an anthill that is damaged by an animal looking for food can be used as an analogy for this type of autonomous behaviour [53]. The ants will typically respond to this event by repairing the damage, often in a different pattern to reflect new conditions.

Self-adaptation is often a necessity for systems that operate under changing conditions. This requires that the system can reflect on *how* the environment affects the devices [54]. Self-management relies on two features. Firstly, the system must have context awareness, which means it can understand its status within the environmental situation. Sheth et al. [55] argue that this requires an intelligent mechanism that can interpret data and make it meaningful in the context of the current conditions. Secondly, the system should be capable of performing adaptive actions, such as self-configuration, self-optimisation, self-healing and self-protection [7, 56], in response to internal or external events or changes.

Context-aware systems commonly adapt to changes in their conditions by adjusting their environmental model, expanding their policies or changing their goals. Such adaptation is driven by situation-based reasoning, that is, a system will adjust its behaviour based on past experiences. This implies that a system can recognise an event it has previously encountered and choose its next action accordingly. Learning is based on storing data about new experiences, that is, situations that have not been previously encountered, for future reference.

Gonzales et al. [58], illustrate the context-aware pattern when they employ case-based reasoning (CBR) as a subsystem in their architecture to adjust the energy consumption in an office building. CBR systems typically analyse the current situation and compare it with data stored in a set of cases to decide the next course of action. A model of the CBR cycle is shown in Figure 2.3.

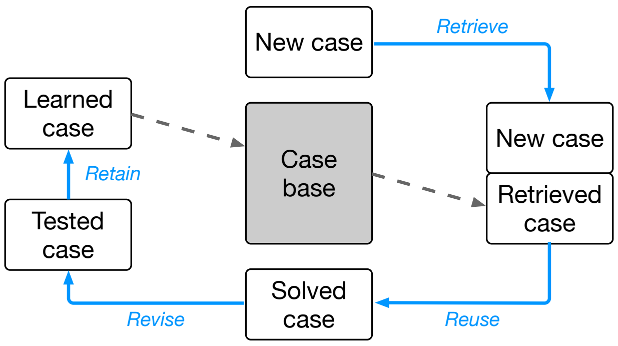

Figure 2.3: Model of the CBR cycle. Adapted from [57]

Aamodt et al. [57] have described how a context-aware system can follow the CBR cycle to solve problems. Firstly, it retrieves the case that is most similar to the current situation. Secondly, it reuses the knowledge in that case to solve the problem. Thirdly, it revises the solution that was chosen. Lastly, it retains important parts of the current experience to use for future reference. If the current situation resembles a known case, the next action will depend on what the system previously experienced and the outcome of the action it performed. If it cannot find a similar case, the system will store the current situation, the responsive action, and the outcome of that action as a new case. This way, the system will gradually expand its knowledge base as it encounters new situations, and hence learn from experience.

2.5 Cognitive IoT Device Management

The concept of cognition is closely connected to self-reflection, which in cognitive science is the active evaluation of one's thoughts, feelings, and behaviours [59]. Transferring this concept to autonomous IoT device management implies that to achieve self-reflection a device needs to be able to see itself in the context of the environment, in its past, present and future states.

Cognitive IoT device management combines elements from autonomic, context-aware and cognitive computing. Information is represented by structured or unstructured data and adaptive behaviour is achieved by combining meta-knowledge with advanced decision-making policies [60, 61]. This way, management systems can adapt to dynamic changes through situation awareness, iterative self-learning and predictions [32]. This means that cognitive device managers can predict future behaviour and consequences based on previous experience even in situations that they have not encountered before. In the wild, an example of such behaviour is the way a herd of reindeer will alter their migration pattern as a reaction to a change in the environment, like a newly constructed human infrastructure or local effects due to climate change [62].

Cognitive IoT device management systems commonly combine statistical learning with reasoning mechanisms. This allows them to predict the effect of new stimuli, events or situations through statistical inference. Based on the result of the prediction, the system can then decide on its next action. To achieve this, machine learning mechanisms are often employed in the system's architecture. These mechanisms enable the system to go beyond simple data analysis and perform complementary tasks such as testing correlations, identifying anomalies and searching for patterns in the data [55, 63]. The learning process involves comparing predictions to actual observations and feeding this data back to the statistical prediction model through model training mechanisms.

The adaptation process is usually triggered by an event that has been observed, a change in the situation or a forecast. In Paper G, we proposed a classification system for the mechanisms that drive these processes. We identified reasoning, learning, and planning as the three fundamental mechanisms that enable cognitive behaviour. *Reasoning mechanisms* analyze the situation, reflect on the context, control the data flow, and determine if there is a need for adaptation or adjustments. *Learning mechanisms* ensure that the knowledge base is up to date and that the prediction models are properly trained. *Planning mechanisms* are responsible for deciding on corrective actions, producing action plans, and synchronizing the device manager with the device. On an architectural level, cognitive behaviour thus emerges from the interaction between the adaptation components and adaptation mechanisms (detailed in Section 6.2.2 and Section 6.2.3).

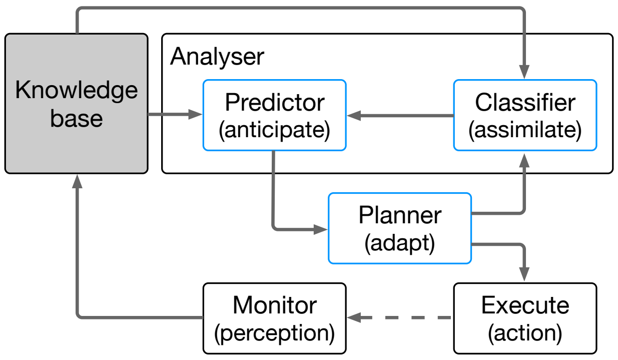

Figure 2.4: Adapted model of the cognitive pattern in the framework proposed by Faraji-Mehmandar et al. [64].

Faraji-Mehmandar et al. [64] propose a framework for optimising workload balance in an edge computing scenario that demonstrates the cognitive pattern. The framework is based on the MAPE-K loop and uses machine-learning mechanisms coupled with active planning policies based on fuzzy logic to obtain an optimal state in workload balance. An adapted model of their framework can be seen in Figure 2.4. The framework is centred around two adaptive components: An analyser component which uses machine learning to predict future workload for the system and handle any violations of the service-level agreement, and a planner component that allows dynamic resource provisioning using fuzzy logic.

The organisation of components in their architecture resembles the model for a *cognitive planner* pattern that we proposed in Paper G. This is in line with two of the core observations we made in our analysis of the 32 reviewed architectural models in this paper. Firstly, many architectures handle complexity with multiple, independent control loops, which often extend the autonomic MAPE-K pattern. Secondly, a common strategy for cognitive IoT device management systems is to combine several adaptation mechanisms. A possible reason for this is that distributed processes, microservice architectures and modularisation help break the problem down into smaller chunks, which are easier to solve [65].

Chapter 3

State of the Art

This chapter provides an overview of the state-of-the-art research in autonomous IoT device management (AIDM). We will evaluate the research in this domain from two perspectives. The first perspective is the gradual shift in focus from basic autonomic solutions to context-aware platforms and advanced cognitive architectures. The second perspective is the three primary areas of research within this domain: network management, which deals with the infrastructure of wireless sensor networks [66], application management, which oversees the maintenance of applications running on the devices [67], and resource management, which ensures perpetual operation [68].

In the following sections, we examine the history of AIDM from an evolutionary perspective and provide examples of research in each of the three management areas. We have chosen to divide the research into four periods, based on complexity level. These periods are named the autonomic period, the early context-aware period, the late context-aware period, and the early cognitive period. Additionally, we briefly touch upon the current status of AIDM in commercial and open-source IoT platforms towards the end of this chapter.

3.1 The Autonomic Period (2001-2009)

When Petr Jan Horn introduced the concept of autonomic computing in an IBM white paper [69] back in 2001, it was soon picked up by researchers in various fields. Among early practitioners, we find Kepart et al. [52], who published a groundbreaking paper in 2003 where they depicted their vision on this topic in general terms.

In the field of AIDM we have found some examples from this research period that were inspired by these papers. In 2003, Milenkovic et al. [70] wrote about autonomic network management for wireless sensor networks (WSNs). They showed how self-configuration and self-optimising can improve networking properties like scalability and resiliency for nodes and resources in distributed computing. The year after, Marsh et al. [71] demonstrated autonomic application management for distributed and computationally challenged devices connected through WSNs. They proposed using multi-agent systems to apply autonomic mechanisms for self-configuration and self-optimising to improve scalability and resiliency. In 2006, Kang et al. [72] described a platform for centralised and autonomous resource management on the device level. In a simulation, the authors demonstrated how a power-aware cluster of self-managed devices can employ activation mechanisms based on a feedback control loop to help balance the energy consumption in a network of sensor nodes.

Upon reviewing surveys and case studies from the autonomic period, we see that the majority of the discussion focuses on using autonomic mechanisms to solve isolated problems. Examples of such mechanisms within networking include device discovery [73], service discovery [74], network resiliency [75], and connectivity restoration [76].

On the application level, we observe that many researchers focus on computational optimisation, energy consumption modelling, and sensing capabilities of the devices, as demonstrated in the survey by Vieira et al [77]. Another topic was architectural challenges for wireless sensor networks concerning clustering and task orientation on the architectural level for wireless sensor networks, as shown by Duan et al. [78]. We also find security issues related to autonomic mechanisms, like securing communication channels [79], intrusion detection [80] and software updates [81].

For autonomic resource management, we see an early example of autonomic mechanisms in the work of Kansal et al. [82] from 2007, where they demonstrated a duty-cycling algorithm for energy-neutral power management in energy harvesting sensor networks. Another example from the same year is Shah et al. [83], who employed Q-learning and probabilistic actions on the devices for adaptive and autonomous resource and task management by optimising global system-wide parameters like total energy usage and network lifetime.

3.2 The Early Context-aware Period (2010 - 2013)

In the early context-aware period we see that the term "Internet of Things" is increasingly used to describe networks of inter-connected wireless sensor nodes. Researchers also discuss architectural aspects on a more complex level and stress that devices need to be managed in a context. For instance,Tan et al. [84], defined in 2010 the IoT as "things [that] have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user context".

A year later, Bandyopadhyay et al. [85], discussed how smart devices with inter-device communication capabilities lead to smart systems with a high degree of autonomy, which in turn enables new applications and services. Using dynamic network discovery mechanisms and energy harvesting methods as examples of enabling technology, they described a vision where independent constrained devices in federated edge networks are capable of adapting to changes in the environment and performing tasks like run-time configurations, data capture, event transfer and network connectivity autonomously.

Also in 2011, Lopez et al. [86] published a paper concerning the benefits of context-aware clustering. They proposed that it could enhance the lifetime, scalability, and robustness of autonomous intelligent objects. These objects were based on the concepts of intelligent and adaptive agents, respectively. There is a distinction between these types of agents: While intelligent agents can perceive dynamic changes, use reasoning to interpret data and decide on actions [87], adaptive agents can sense and act autonomously in complex environments and adapt to situations to achieve a set of goals [88].

Another pattern for the early context-aware period is that research diverges into different aspects of *self-management*. Atzori et al. [89] wrote in their survey from 2010 that a central issue for IoT architectures is to enable adaptation and autonomous behaviour without compromising on networking, application or resource aspects, like scalability, security, or computation and energy capacity, respectively. In 2011, Horre [90] made a clear distinction between the different types of management in his PhD. Here, he stated that due to 1) resource constraints of wireless sensor nodes; 2) the need to avoid manual intervention; 3) the need to facilitate multi-purpose usage; and 4) the need for flexible reconfiguration, architectures for device management solutions must take resource efficiency, reconfigurability, multi-application support and autonomy into account. In 2013, Vlacheas et. al. [91] showed that the context of the physical environment in which the device is located has a strong influence on the autonomous management framework they proposed. This framework could select behaviour dynamically through self-management and self-configuration, based on domain knowledge and knowledge about the context of the operation itself. However, their paper had a strong focus on addressing service provisioning in a setup of heterogeneous devices and did not look at operational management beyond this topic.

Examples of context-aware AIDM can also be seen within *network management*. In 2011, Tsagkaris et al. [92] discussed architectural challenges in the field of autonomous wireless networking related to context awareness, decision-making and learning, while one year later Movahedi et al. [93] described how to support self-management to reduce the cost and the complexity of managing network infrastructures by employing MAPE-K control-loops in their architecture.

Within *application management*, Rajan et al. [94] employed context awareness in their paper from 2011, which showed a simplified architecture that takes environmental parameters, network state and application context measurements into consideration to run auto-configurations of WSNs. The architecture made use of a control loop, an optimisation engine and policies to achieve the goals of the system. In 2013, Liu et al. [95] presented an autonomous, context-aware solution for application management that used ontology models to represent and facilitate context sharing between WSN nodes, while Brown et al. [96] worked on how to enable autonomous over-the-air software updates for resource-constrained WSNs.

As for *resource management*, we observe that research from the early context-aware period still was mostly focused on autonomic mechanisms, usually located in the devices themselves. For instance, Ferry et al. [97] described in 2011 an architecture that employed a dynamic power management policy that takes environmental factors like weather forecasts into account, to identify hotspots for energy consumption. Nonetheless, their simulation was aimed at decision support during system design, like where to place devices and how to scale the capacity of energy harvesting and energy buffering of the individual sensor nodes. The authors did not consider using such mechanisms for actual autonomous resource management. Another example from this period is the discussion on energy harvesting for autonomous WSNs [98, 99, 100]. The same pattern is evident in broader research on resource management for IoT, for instance in the survey by Wan et al. [101] from 2011. Here, they presented a review where energy harvesting and energy management technologies were explored to enable dynamic power management for wireless sensor networks. However, their focus was on enabling technologies, not how to manage the energy budget for the devices autonomously.

The examples from the early context-aware period show that researchers to a larger extent are combining perspectives to promote autonomous behaviour in the field of IoT management. Even so, there are still few discussions about truly cognitive features such as predicting future states of wireless networks or devices and devising strategies to meet predicted changes.

3.3 The Late Context-aware Period (2014-2017)

In the late context-aware period the discussion turns towards adding *knowledge* and *intelligence* to AIDM systems.

An example from 2014 is a paper by Wu et al. [61], which discussed the challenge of giving objects connected to IoT the capability to learn, think, and understand their physical and social environments independently. The authors proposed a theoretical framework for supporting semantic derivation and knowledge discovery, by employing cognitive mechanisms like perception-action cycles, massive data analytics, intelligent decision-making and on-demand service provisioning. In their framework, interconnected physical and virtual objects behave as intelligent agents. This allows for adaptation to changes with a minimum of human intervention or interaction through context-awareness mechanisms and automated learning processes. Although their work shared many of the same ideas that are presented in this thesis, their component models studied knowledge discovery, decision-making and learning in isolation. They did not provide a complete architecture showing interaction between the components.

The same theme was also present in the work of Kibria et al. [102] from 2015. Here, they proposed a context-aware architecture based on virtual objects, using a context-matching algorithm to provide user-centric features in IoT. In addition, the authors showed how new knowledge can be added to the system by using semantic ontology. However, this study only addressed the management of services, not the actual devices in the network.

In this period, we also see signs that researchers incorporate cognitive patterns for IoT device management. For instance, in 2016 Sheth et al. [55] used the term 'cognitive IoT' to connect the field of IoT with the field of cognitive computing. They proposed to use perceptual computing to add context awareness, cognitive computing to discover patterns in data, hypothesise about correlations and quantify uncertainty, and semantic computing to provide meaning. However, their paper only discussed this idea in general and theoretical terms and did not mention autonomous management explicitly, which shows that the research field was still maturing.

Research that explored the relationship between IoT and cyber-physical systems was another trend in the late context-aware period. A common practice was to employ digital twins, that is, virtual copies of physical devices, where self-awareness was combined with reasoning mechanisms to provide recommendation engines. For instance, Alam et al. [103] described in their paper from 2017 an architecture that used digital twins, bayesian networks and rules derived from fuzzy logic to analyse the current context of the system and recommend control actions for adaptation, if necessary. A year later, in 2018, Boschert et al. [104] argued that digital twins, embedded in the devices themselves, on the edge or in the cloud, represented a loop between the physical and virtual worlds.

Researchers also start recognising the challenges concerning architectural complexity. In 2015, Derhamy et al. identified in their survey [105] different approaches that were used by commercial platforms. One identified approach was data-centric and utilised the cloud for collection, analysis and visualisation of data.

These platforms were typically used for asset tracking, logistics and predictive maintenance. Another approach made use of smart objects that enable processing at the endpoints. The focus for these platforms was often to support distributed automation tasks either on the edge or in the cloud for devices that required a high level of device independence. As a consequence, these architectures required adaptation mechanisms on different levels of complexity. However, the paper did not go into detail about how management platforms can support such devices autonomously.

In this period, autonomous *network management* research made significant progress, and we see several examples of researchers adding cognitive elements to their architecture. For instance, Meriem et al. [106] published a white paper in 2016 proposing a reference model for cognitive networking and self-management of networks and services. The paper outlined requirements for autonomic management and control, using decision-making elements that employ cognitive control loops to enable autonomous behaviour. They argued that reasoning, learning and decision-making control mechanisms can dynamically assign network resources to adapt to changes in network policies, context, or events that may affect the availability, reliability, or quality of network services. However, this generic white paper only covered the design principles of autonomous network management without describing the interaction between components or the required adaptation mechanisms at an architectural level.

Another trend in the late context-aware period was to apply software-defined networking (SDN) to achieve self-organising networks. This can for instance be seen in the works of Tsagkaris et al. [107], who employed control-loops to control autonomous functions in each network layer, and Neves [108], that used data mining and stochastic algorithms to provide intelligence to diagnose problems, decide actions and enforce adaptation. In 2017, Zhao et al. [109] argued that SDN in combination with machine learning applied to large data sets has great potential for self-management and automatic adaptation. Even so, their paper did not explore that topic any further. We also see the SDN theme in broader research. One example is a survey from 2017, where Ndiaye et al. [110] looked at SDN in light of management aspects regarding network configuration, topology, QoS, energy, security and maintenance. However, they did not touch upon the autonomous aspects of management. So, although SDN could be a promising mechanism for enhancing AIDM, we see few examples of research exploring this topic in any depth.

We also see that the term 'cognitive IoT' was sometimes used when discussing purely autonomic networking aspects. An example from 2015 is the paper from Afzal et al. [111], where the authors described an architecture that aimed at improving connectivity for large-scale deployments of solar-powered IoT devices by using probabilistic methods. However, their connection to context-aware or cognitive computing was limited. This line of research is thus of little relevance to this thesis.

For autonomous *application management*, we can see that middleware was sometimes introduced to connect back-end systems to the devices and provide context awareness, knowledge management and intelligent decision support. In a survey from 2014, for example, Pererera et al. [36] showed that middleware can provide solutions to technical challenges related to heterogeneity, interoperability, security and dependability. The same year, Fortino et al. [112] analysed and discussed the requirements for a general middleware platform. They argued that adaptation at run-time is crucial to obtain effective management for the system, due to heterogeneity related to devices, interfaces, data streams and context. This requires the middleware to do knowledge management. Middleware frameworks were also used to enable the different application components to communicate and exchange data about themselves and their sensed environment to other agents in the system. In 2015, Venkatesh et al. [113] addressed the challenge of scalability for context-aware applications and provided a middleware architecture based on context engines that use general statistical learning to translate heterogeneous sensor data into high-level context data. This enabled the system to take contextual information about the devices in the system into account for reasoning. In 2017, Nascimento et al. [114] presented a framework where adaptive agents worked in collaboration with observer agents. The main task of the observer agents was to monitor the adaptive agents and decide when the actions deviated from what was expected. This would trigger a learning method that generated new configurations for the adaptive agents and test how this affected their operational behaviour.

Within autonomous resource management, we see that the field matured significantly compared to just a few years earlier. In 2017, Delicato et al. [68] discussed context-aware resource management concerning the IoT-edge-cloud ecosystem. They argued that resource monitoring and adaptive resource allocation are prerequisites for providing high-quality services, due to the dynamic nature of the IoT and variations related to the users, the network, the physical environments and the devices themselves. This implies that resource managers must have access to adaptive decision-making mechanisms. However, even if the authors discussed the challenges related to adaptive resource management in depth, they did not address how to solve this autonomously on an architectural level. The same year, Khazaei et al. [115] proposed an autonomic IoT platform that used a MAPE-K loop and micro-service architecture to support edge data processing and an autonomic manager that optimised QoS and utilisation of resources at run time. Still, in 2017, Bacciu et al. [116] proposed a conceptual architecture for an IoT learning service. Their architecture included an adaptive task component centred around a resource scheduling mechanism which was based on machine learning. However, Khan et al. did not go into detail about how to implement the actual mechanisms that are needed for adaptation, or how to predict future states of the system.

To conclude this section, we found many examples of AIDM architectures, platforms and frameworks in the late context-aware period that showed improvements regarding combining context-awareness with knowledge management and intelligent decision-making. However, we did not find significant evidence of cognitive features, like adaptive learning or prediction mechanisms.

3.4 The Early Cognitive Period (2018-2023)

Since 2018 we have seen rapid evolvement in research on cognitive IoT device management. In the last years, an increasing number of researchers have published papers showcasing frameworks, platforms and architectures that use reasoning, learning and planning mechanisms to provide cognitive behaviour. We will now look at some recent examples of IoT platforms that employ cognitive features within network, application and resource management, respectively.

3.4.1 Cognitive Network Management for the IoT

In 2018, Khan et al. [117] published a paper where they traced the history of autonomous network management from policy-based management, via ontological reasoning, agent-based networking and self-organised networks to control-loop-driven management. The authors looked at three different ways to apply artificial intelligence for reasoning and learning, namely probabilistic models, game-theoretic models and machine learning. They concluded that while traditional autonomic networking aims at achieving self-management by the use of automated control loops, the future lies with cognitive networking that can perceive its environment, adapt to new situations, and learn from experience using artificial intelligence. This is in line with the categorisation of AIDM that we use in this thesis, as described in Chapter 2. However, their paper did not show any actual architectural models that describe the adaptation components that are needed to realise such systems or the interaction between them.

Also in 2018, Hernandez [66] addressed in his thesis the challenge of gathering insights about events manually in a heterogeneous IoT network. He proposed the use of machine-learning models on big data sets and autonomic computing based on a MAPE-K loop to analyse incoming network data and increase the performance and reliability of data analytics for network monitoring. Nevertheless, his work focused mostly on autonomous monitoring, analysis and diagnostics of system data to provide decision support for human operators. The challenge of autonomous management of the network resources was only described superficially.

In 2021, Arzo et al. [118] conducted a review of autonomic network management since 2004. Here, they proposed a multi-agent-based network automation system using cognitive agents organised in a micro-service architecture to handle the challenges associated with software-defined networking. Their architecture was based on the reference model published by Meriem et al. in 2016 [106], which we mentioned in Section 3.3. The authors used a stochastic approach based on Markov processes for the analysis and mathematical formulation of the multi-agent system behaviour but stated that the intelligence could be realised by other mechanisms, like machine learning, deep learning or computational intelligence. However, their validation was based on a simplified case, and although their preliminary analysis was promising, they concluded that further research would be needed to validate that autonomous network management using multi-agents can provide scalable, flexible, dynamic, and resource-efficient systems.

The following year, Abbas et al. [119] presented an autonomous network management platform. Their architecture was built around a closed-loop model based on an intent-based networking mechanism for autonomous control, orchestration and management of edge-to-edge network slicing, and a mechanism for data analysis that employed ensemble learning to detect and predict anomalies and activate and execute mitigation policies. However, the proposed architecture was specialised towards network slicing management and can thus not be generalised to other use cases.

3.4.2 Cognitive Application Management for the IoT

In 2018, Suh et al. [120] proposed an autonomous agent-based IoT middleware framework that supports dynamic adaptation and learning, to manage applications for ambient intelligence. Here, they presented an architecture that models individual agents and the collaboration between them, and a software framework to support the development of ambient intelligent applications based on the proposed architecture. However, their work mostly focused on the interaction between agents, not the resources needed to support them.

In their 2021 paper, Dzeparoska et al. [121] described the architecture of a self-driving intent-based WSN that can predict changes and adapt autonomously. Their model demonstrates how a network receives high-level goals in the form of intents and then uses intelligent mechanisms to convert these intents into policies that can execute MAPE-K control loops to achieve these goals. The pipeline is controlled by an autonomous policy-based manager that can employ techniques like device programmability, massive scale monitoring, big data analytics, and machine learning. Their work is in line with the ideas outlined in this thesis but does not consider how to perform resource management on the devices.

Agyemang et al. [122] proposed in 2022 a general distributed software-defined management framework for IoT devices that employed autonomic computing and embedded knowledge engines to adapt autonomously to dynamic changes. The knowledge engines ran lightweight machine learning, fuzzy rule-based systems, and control functions to enable adaptation processes, like starting learning processes, initiating machine-to-machine communication, performing self-management or executing event-driven tasks. This architecture shares many of the core concepts that we propose. That said, their architecture is aimed at running on the devices themselves. Thus, it might not be usable for constrained devices. Also, their architectural model is not described on a component level, nor does it explain the interaction between them through concrete mechanisms.

In 2023, Alfonso et al. [123] presented a multi-layered architecture for the execution of self-adaptive rules to improve the Quality of Service of IoT applications. Their runtime framework was built around a MAPE-K loop, where pre-defined monitoring, analyzing, and planning operations are used for the deployment, configuration and execution of the adaptation engine and other software components. However, none of the included platforms include a dedicated learning mechanism. Thus, their architecture lacks a key component for cognitive management.

3.4.3 Cognitive Resource Management for the IoT

Resource management has matured more slowly compared to network and application management, but it has been catching up in the later years. This is evident from the increasing number of reviews being published in this field since 2018. For instance, in their review from 2019, Chatterjee et al. [124] argued that IoT devices require platforms that support resource-aware operation and autonomous adaptation, owing to their heterogeneity, resource constraints, context-variability, and security. Similarly, Wang et al. [125] reviewed in their 2021 paper 30 case studies where they concluded that energy management mechanisms based on techniques like deep reinforcement learning, artificial neural networks and machine learning are well suited to support cognition and learning. The authors also observed that autonomous power management mechanisms combined with energy harvesting techniques are often employed to maximize energy efficiency. The same year, Li et al. [126] carried out a review which focused on the use of decision support systems (DSS) for energy management in the IoT. They identified five predominant decision-making models in use, namely supervised machine learning algorithms, unsupervised recommender algorithms, fuzzy logic and methods, natural-inspired and evolutionary algorithms, and deep learning. Their analysis concluded that DSS can improve energy efficiency, reduce energy consumption, faults, and errors, and enhance performance and accuracy in smart applications. These findings are consistent with the results from our research, as outlined in Section 6.2.

We have found some examples of cognitive architectures and frameworks that are used for resource management in the IoT. One example is from 2020, when Bharti et al. [127] demonstrated a framework for IoT resource discovery and selection. In their framework, knowledge is represented by using a shared virtual composite ontology, while resource discovery is achieved using fuzzy control rules on the knowledge base. Then, in 2022, Liao et al. [128] proposed a cognitive balancing architecture for IoT resources, centred around knowledge acquisition and sharing. Their architecture included separate components for cognitive monitoring, processing and storage. In 2023, Kumar et al. [129] presented a framework based on a MAPE-K loop for autonomic workload prediction and allocation of resources. Their architecture employed an auto-encoded deep-learning model for cognitive analysis and prediction of workload, and a meta-heuristic crow search algorithm for resource allocation. However, none of these case studies detailed the interactions between components or the specific mechanisms that drive the cognitive processes.

3.5 Commercial and Open-source Platforms for Autonomous IoT Device Management

As we have seen, there are multiple case studies and reviews that discuss AIDM from a specialised perspective in recently published research. However, we have not seen any evidence of research on comprehensive platforms for cognitive management of IoT devices yet. To determine whether this is also true for open-source and commercial actors, we conducted a brief market review of commercial platforms to examine the autonomous services which are offered for IoT device management.

We found three recent studies that look at the general traits of commercial IoT platforms. These studies were analysed to find information regarding the provided device management services. The first study is from 2019. Here, Asemani et al. [130] surveyed 13 commercial platforms and 11 open-source platforms. In the review they looked at several factors, device management and service management being two of them. While their definition of device management includes both network and application management, their description of service management maps well to our definition of resource management. Their results showed that while all the reviewed commercial platforms supported basic network and application management, only six of thirteen platforms were able to deliver services for resource management. For open-source platforms, nine out of eleven platforms included services for network and application management, while none provided any kind of resource management.

The second study is from 2020, when Mijuskovic et al. [131] made a comprehensive survey on how five IoT platforms scored for a variety of factors and functional requirements. The authors stated that a typical commercial IoT platform enables provisioning, management and automation for devices connected via the IoT. Here, network, platform, and resource management were among the included perspectives. The results showed that all the reviewed platforms used a centralised approach and provided services in the cloud to manage the devices. Mijuskovic et al. also found that for resource management, three of the five reviewed platforms supported energy awareness on the device level. However, none of them included automated services for balancing the energy budget.

In the third study, published in 2021, Babun et al. [132] surveyed eight IoT platforms to study their support toward topology, programming languages, event handling, third-party support, security, and privacy. Although the authors did not look into the management aspect specifically, they stated that many platforms allow third-party support that can extend the functionality in that direction. Also, some platforms, e.g., the Watson IoT Platform from IBM, have built-in cloud service integrations that make it possible to connect management services to for instance machine-learning environments via an API or SDK. In theory, that makes it possible to achieve some level of autonomous management. However, such integrations, to either third-party applications or built-in services in the platform itself, must still be implemented manually.

We did not find indications that any of the surveyed platforms provided autonomous services for network, application or resource management above autonomic level. To validate this observation, we looked at the descriptions, feature listings and documentation of some commercial and open-source IoT platforms. These were Microsoft Azure IoT Central [133], AWS IoT Device Management [134], SAS IoT Analytics Solution [135], IBM Watson IoT Platform [136], Particle [137], Kaa Enterprise IoT Platform [138] and Friendly One-IoT Device Management [139].

Our analysis showed that the focus is still mostly on providing basic features, like deployment, discovery, monitoring, remote configuration and updating of devices. SAS seems to be the sole exception to this rule. Their documentation state that the concept of ‘artificial intelligence of things’ (AIoT) will become more prominent, that learning, collective intelligence and value of information will drive the future IoT, and that future systems will have the capability to optimise themselves for any scenario. However, it did not provide any examples to prove that this platform can provide cognitive device management as an out-of-the-box service.

Chapter 4

Overview of publications

This chapter provides an overview of the publications that are part of this thesis, plus one auxiliary paper. Papers A to F are case studies. Each of these papers includes an architectural model that describes the components and mechanisms in the implemented system. Paper G is a structured literature review. It contains a generalised architectural model based partly on the six case studies, and partly on patterns and best practices identified in that paper. The auxiliary paper is a review of fog computing in healthcare. Although it did not contribute directly to this thesis, it helped identify the direction and scope of the research.

To visualize the evolution of our proposed cognitive architectural model, we describe the architecture in each paper using the notation developed in Paper G. The notation used in the model is based on the principles of service-oriented architectures (SoA), where each container represents a component that either provides or consumes a discrete service. There are two types of components: System components and process components. We have identified four system components in our examples: 'Device,' 'Device manager,' 'System manager,' and 'Server.' These act as containers for five process components: 'Perception,' 'Action,' 'Adaptation process,' 'Declarative knowledge' and 'Procedural knowledge'. 'Adaptation process' is further divided into five categories, namely, 'Monitor', 'Analyze', 'Learn', 'Predict' and 'Plan.' Arrows in the notation represent service calls, triggers, or data flows. The notation is further detailed in Paper G and in Section 6.1.

In the following, we will describe each paper and how, retrospectively, the respective architectural model contributed to the research goal.

4.1 Summary of included papers

Paper A (Ahlers et al. 2018)

Title:
Analysis and Visualization of Urban Emission Measurements in Smart Cities
Presented at:
The 21st International Conference on Extending Database Technology (EDBT), Vienna, Austria, March 2018

In Paper A we present a case study that describes the architecture of a wireless sensor network (WSN) for data collection, management and analytics. The WSN consists of several solar-powered sensor nodes deployed in the cities of Trondheim, Norway and Vejle, Denmark. Apart from the sensor data, several publicly available data sets are collected and included in the data analytics and visualisation to improve data quality and support analysis.

The study highlights two challenges related to the energy management of the devices and the data integration process. First, traditional, fixed energy management fails on two accounts: In the summer months, the sensor nodes have access to an abundance of energy which they potentially could have utilised to collect more data of higher quality. However, the high variance in energy due to seasonality and volatile weather conditions makes it difficult for the devices to stay in operation throughout the winter. Second, data integration is challenging when the sources contain highly heterogeneous data, with different timescales, measurement frequencies, spatial distributions and granularities. This requires that each data stream is tailor-made, which is a big hindrance to the scalability of smart city systems and applications. Thus, the paper identifies the need for a management platform that can autonomously handle heterogeneous data streams in a non-stationary environment.

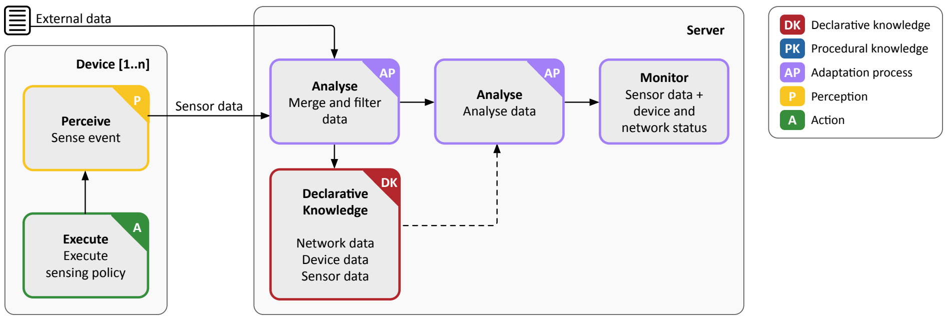

Figure 4.1: Component diagram of the data collection platform. Adapted from Paper A.

The architectural model included in the paper can be seen in Figure 4.1. The model makes a distinction between sensor data and meta-knowledge, i.e., collected data that shows the status of the system. The model also defines four fundamental process components in the generalised model, namely perception, action, adaptation process and declarative knowledge.

Paper B (Braten et al. 2017)

Title:
Towards Cognitive Device Management: A Testbed to Explore Autonomy for Constrained IoT Devices

Presented at:
The Seventh International Conference on the Internet of Things, Linz, Austria, October 2017

Paper B is a demonstration paper where we describe the setup for a platform that is responsible for collecting environmental and contextual data that can be used to study principles and mechanisms for autonomous resource management for IoT devices using machine learning.

Through the work of the testbed, we found that resource-intensive tasks can be outsourced to a device manager located in the cloud even when the communication lines are constrained. In addition, devices must be able to receive commands and change their operations accordingly. Finally, an autonomous IoT manager who oversees devices that operate under volatile and non-stationary conditions must be able to identify previously unseen events and learn from new experiences. Hence, we defined cognitive device management as an evolution of device management systems that can manage the learning process on their own.

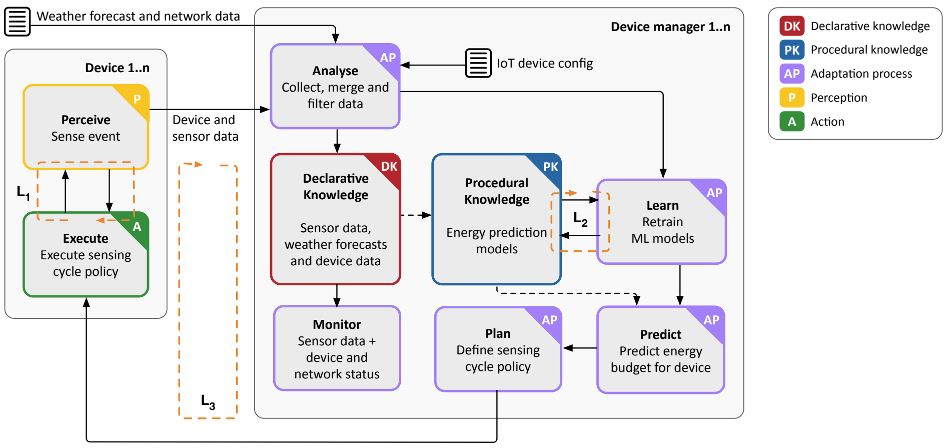

Figure 4.2: Component diagram of the autonomous sensor testbed lab. Adapted from Paper B.

The architectural model presented in the paper is shown in Figure 4.2. It introduces several important concepts: First, resource management is a separate entity that is distinct from network management and application management. Second, the model shows that an autonomous IoT system requires a separate cognitive device manager for each device, to support individual adaptation. Third, there are three types of data present in the system, namely application data, contextual data and meta-data. Fourth, knowledge can be either declarative or procedural. Fifth, autonomous adaptation is driven by three types of control loops; one for autonomous device *operation*, one for autonomous device *adaptation*, and one for autonomous *learning*. These loops are in turn based on four different adaptation mechanisms: analysing, learning, predicting and planning.

Paper C (Kraemer et al. 2017)

Title:
Solar Energy Prediction for Constrained IoT Nodes based on Public Weather Forecasts

Presented at:
The Seventh International Conference on the Internet of Things, Linz, Austria October 2017

Paper C is a continuation of Paper B. Here, we present a case study where we focus on the challenges related to the operation of constrained, energy-harvesting devices located in a non-stationary environment. More precisely, we examine the influence that seasonal variations in sun angle and short-term effects of volatile weather have on the energy output from a photovoltaic (PV) panel. We also discuss the problem of optimising data output from a solar-powered IoT device versus the need to avoid device failure due to over-consumption of energy.

To investigate this problem, we study how different machine learning methods can be used to predict the future energy intake of constrained solar energy harvesting devices, based on publicly available weather data. The evaluation is based on data provided by commercially available IoT hardware, demonstrating the feasibility of the proposed solution in a real deployment. Our results show that predicting solar energy is possible even with limited access to data and that the predictions can progressively improve as the system runs.

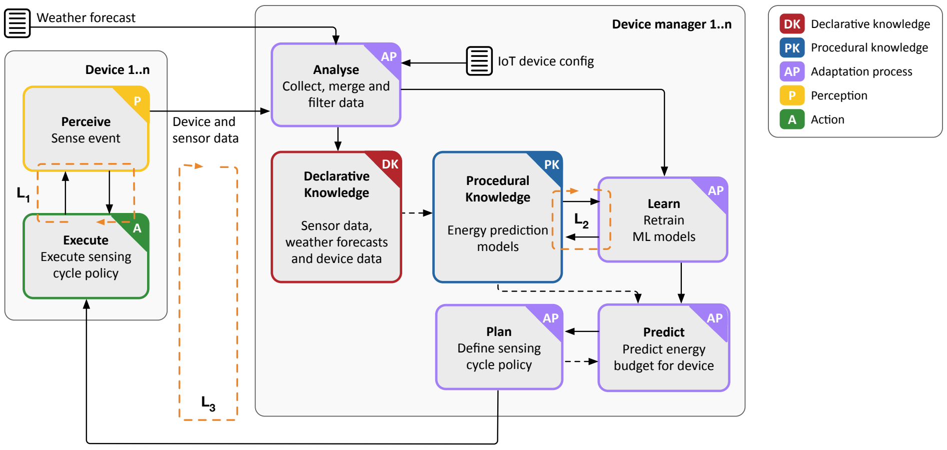

Figure 4.3: Component diagram of the proposed platform for autonomous operation of constrained IoT devices. Adapted from Paper C.

The paper identifies three requirements that must be fulfilled to enable optimal operation for such devices. First, sensor nodes must be aware of their energy budget and be able to plan their future energy consumption accordingly. Second, due to the scale of many IoT deployments, maintenance and adaptation must happen autonomously. Third, sensors that are placed in non-stationary environments must be able to adapt individually.

The architectural model, shown in Figure 4.3, is essentially the same as the one shown in Paper B. The main difference is that the Monitor process is removed since it is only used to visualise data to a human observer, and thus is not necessary for autonomous adaptation.

Paper D (Kraemer et al. 2020)

Title:
Operationalizing Solar Energy Predictions for Sustainable, Autonomous IoT Device Management
Published in:
IEEE Internet of Things Journal (2020), 7(2), pp. 11803-11814

In Paper D we explore how to select and optimise machine-learning models to improve the prediction accuracy of solar power that is harvested by IoT devices. In the paper, we examine which machine learning models, feature sets and sampling rates gain the best results for a medium-term forecasting horizon.

The study used one year of observational data from Trondheim in a simulation to evaluate IoT devices deployed in an operational setting without any prior data. The results from the simulation were then analyzed to determine which hyperparameters to use. The study also obtained empirical estimates for the amount of training data required for sufficiently accurate solar forecasts for energy budget planning.

Our results indicate that device managers can benefit from optimising energy harvesting models since this opens up more strategic energy planning. Further, manual tuning for individual devices in a large-scale deployment is unnecessary even if the devices have different configurations, as individual and autonomous machine learning models can be employed easily for each device individually.

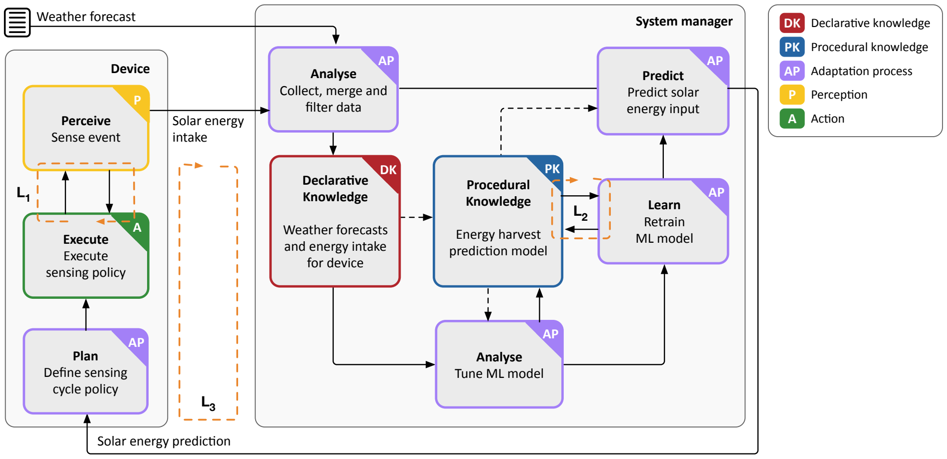

Figure 4.4: Component diagram of the proposed IoT device management platform. Adapted from Paper D.

The architectural model, shown in Figure 4.4, introduces the system manager. This component handles the processes that do not influence the individual devices. It also shows the importance of taking parallelism into account in autonomous systems, as the internal meta-processes of an autonomous device manager might not run sequentially.

Paper E (Braten et al. 2018)

Title:
Towards Cognitive IoT: Autonomous Prediction Model Selection for Solar-Powered Nodes
Presented at:
IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, USA, July 2018

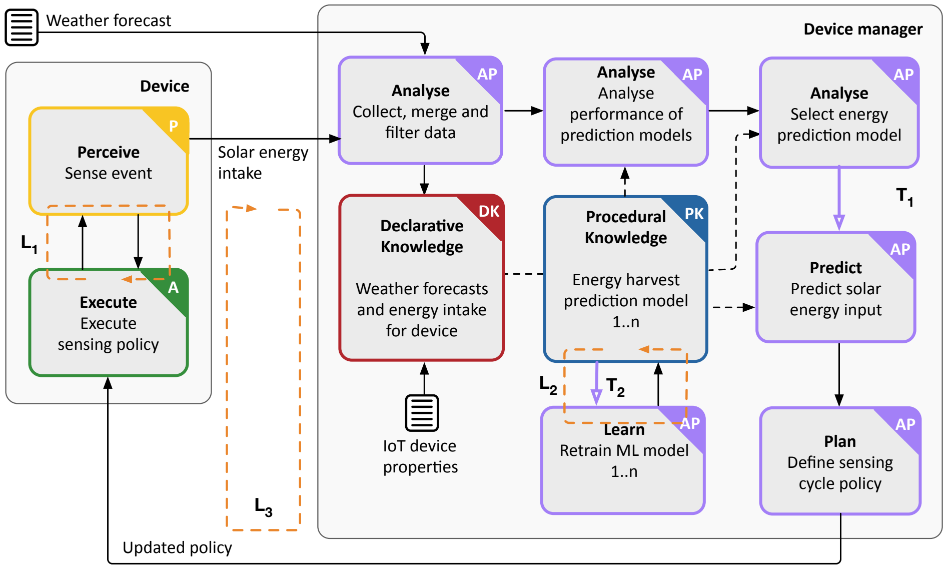

Figure 4.5: Component diagram showing the architecture needed for the energy planning workflow. Adapted from Paper E.

In Paper E we do a case study to investigate how prediction model selection can increase the accuracy of autonomous energy management of solar-powered sensor devices in an environment with volatile weather and seasonal changes. Further, we argue that cognitive device management requires that a system takes contextual data about the operation of the devices into account when managing IoT devices.

For our study, we ran a simulation based on real data gathered in realistic settings. Here, a device manager was provided with the ability to autonomously evaluate several prediction models and then select the one with the highest historical accuracy when predicting the future energy intake of a device.

The statistical analysis of the results from the simulation showed that a classical machine learning model can learn quickly when data becomes available. However, simple physical models are more suitable until the machine learning model has enough data to solve the problem. Thus, applying a set of models can help solve the bootstrapping problem during the start of a new system when training data is scarce or non-existent. In addition, having the ability to switch between models can contribute to the robustness of a system by providing some degree of self-healing.

Finally, the paper suggests how to select a decaying parameter for the model selection algorithm to manage the trade-off between reactivity and stability.

The architectural model in the paper, shown in Figure 4.5, stresses the importance of using triggers to control processes, e.g., to run a machine learning process when a significant amount of new training data becomes available to the device manager. It also validates the concept of context-aware device management for IoT, as it includes a component that can provide support for both self-optimisation and self-healing.

Paper F (Braten et al. 2019)

Title:
Adaptive, Correlation-Based Training Data Selection for IoT Device Management
Presented at:
The Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, October 2019

Paper F is a case study aimed at understanding how autonomous device management can enhance large-scale deployments of IoT nodes operating in non-stationary environments. In particular, we examined the effect that transfer learning has on the initial prediction accuracy for new devices that are introduced into an existing deployment. To this end, we designed a conceptual cognitive architecture that modelled the behaviour of an autonomous device manager who is responsible for energy planning for solar-powered constrained devices.

For validation, we constructed a mechanism where we simulated a device manager that can identify suitable training data for a given node by selecting data from nodes with correlated data. The statistical analysis of the results shows that the proposed training data selection can support self-management in several ways. First, adding relevant learning data when training a model can improve self-optimisation, since this improves the prediction accuracy in the first period after deployment. Third, self-protection can be improved by analysing data from correlated nodes, for instance, if the data suddenly and unexpectedly deviates this might indicate that a device is faulty or hi-jacked.

The case study thus illustrates how employing mechanisms for training data selection and learning management can improve the ability of constrained IoT devices to adapt to changes and act more autonomously. This will in turn decrease the need for manual maintenance and enhance the perpetual operation of the deployed devices in the wireless sensor network.

On an architectural level, we employ both a system manager and a device manager, which in turn reflects the division between general and device-specific knowledge. In Figure 4.6, we can see that the system manager employs a transfer learning data selection policy to identify which previously deployed device correlates most with the newly deployed device. Then the system manager takes the data from the older device and transfers it to the device manager for the newly deployed device. This transfer learning data is then utilised by the device manager when it retrains the prediction models for the device. This means that the device manager acts as a

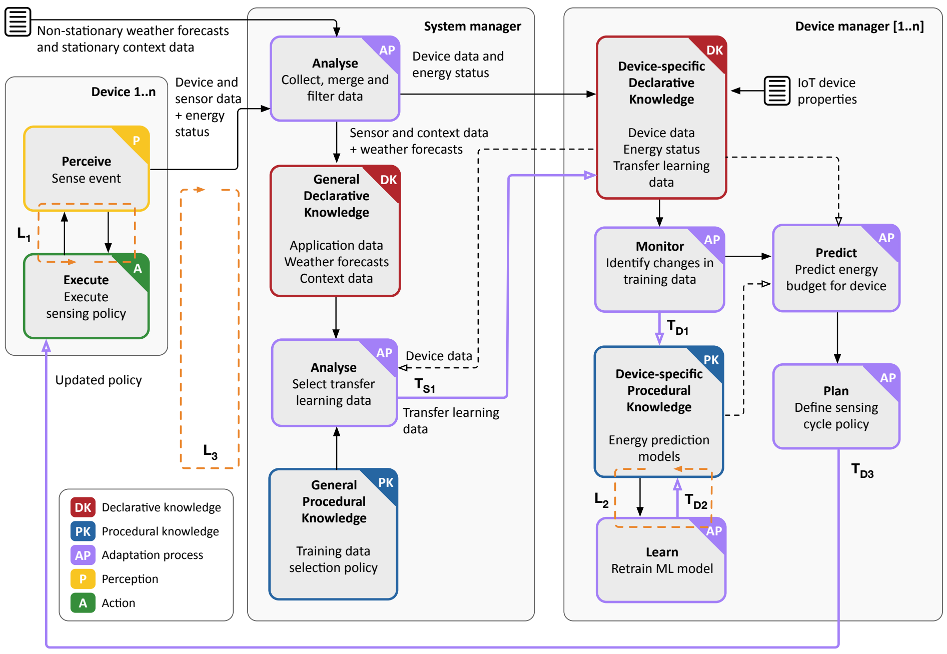
Non-stationary weather forecasts and stationary context data
System manager
Device manager [1..n]
Device 1..n
Device and sensor data + energy status
Analyse
Collect, merge and filter data
Device data and energy status
Device-specific Declarative Knowledge
Device data
Energy status
Transfer learning data
IoT device properties
Perceive
Sense event
Sensor and context data + weather forecasts
General Declarative Knowledge
Application data
Weather forecasts
Context data
L1
Execute
Execute sensing policy
Monitor
Identify changes in training data
Predict
Predict energy budget for device
TD1
Analyse
Select transfer learning data
Device data
TS1
Transfer learning data
Updated policy
L3
Device-specific Procedural Knowledge
Energy prediction models
Plan
Define sensing cycle policy
TD3
L2
TD2
General Procedural Knowledge
Training data selection policy
Learn
Retrain ML model
DK Declarative knowledge
PK Procedural knowledge
AP Adaptation process
P Perception
A Action

Figure 4.6: Architecture of a cognitive device manager responsible for energy planning for constrained IoT devices. Adapted from Paper F.

digital twin of that device, while the system manager models a cognitive control system that handles the general aspects of autonomous operational management on the system level.

Paper G (Braten et al. 2021)

Title:
Autonomous IoT Device Management Systems: Structured Review and Generalized Cognitive Model

Published in:
IEEE Internet of Things Journal, vol 8, no. 6, pp. 4275-4290

Paper G is a structured literature review where we present a comprehensive study on mechanisms for autonomous device management of constrained IoT devices, in the light of management tasks, operational environment, network topology, resource constraints, scalability and management categories.

In the study, we first analysed state-of-the-art models and descriptions of autonomic, context-aware and cognitive architectures from 32 cases. From these cases, we extracted relevant data, which was organised, analysed and synthesised according to a proposed taxonomy of observed adaptation mechanisms. We then studied the autonomic MAPE-K loop as described by IBM in 2005 [49], the cognitive model presented by Vernon [29] and the standard model of human-like minds described by Laird et al. [40], in light of the results from the 32 case studies, to identify common patterns of autonomous cognitive management for constrained IoT devices on component and process levels and best practices for designing and implementing solutions around adaptation mechanisms. Based on the gathered insights and knowledge we then designed a generalised cognitive model for autonomous management of constrained IoT devices.

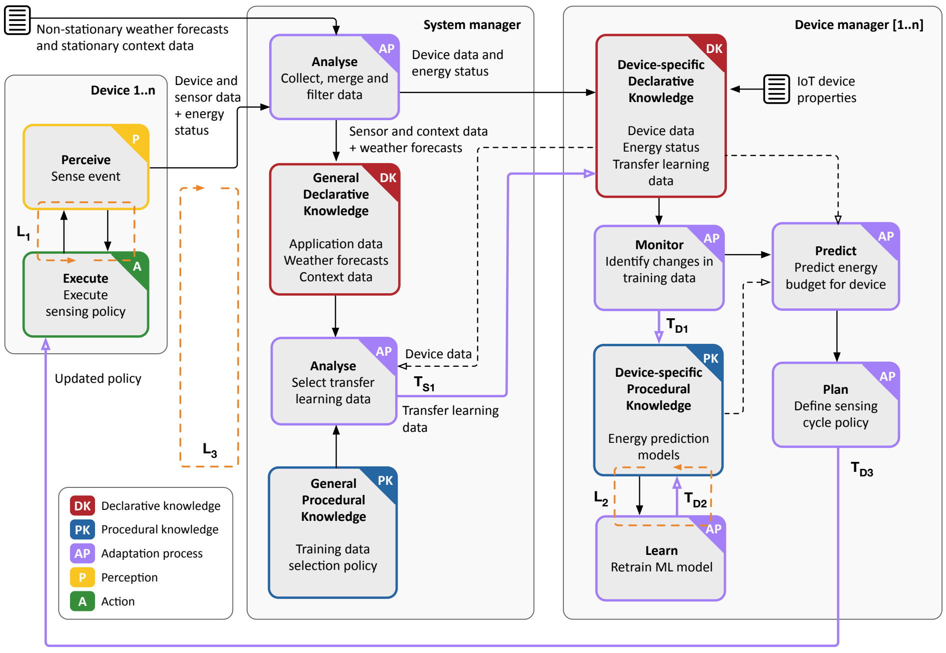
Non-stationary weather forecasts and stationary context data
System manager
Device manager [1..n]
Device 1..n
Device and sensor data + energy status
Analyse
Collect, merge and filter data
Device data and energy status
Device-specific Declarative Knowledge
Device data
Energy status
Transfer learning data
IoT device properties
Perceive
Sense event
Sensor and context data + weather forecasts
General Declarative Knowledge
Application data
Weather forecasts
Context data
L1
Execute
Execute sensing policy
Monitor
Identify changes in training data
Predict
Predict energy budget for device
TD1
Analyse
Select transfer learning data
Device data
TS1
Transfer learning data
Updated policy
Device-specific Procedural Knowledge
Energy prediction models
Plan
Define sensing cycle policy
L3
TD3
General Procedural Knowledge
Training data selection policy
L2
TD2
Learn
Retrain ML model
DK Declarative knowledge
PK Procedural knowledge
AP Adaptation process
P Perception
A Action

Figure 4.7: Component diagram of a generalised cognitive model for autonomous management of constrained IoT devices. Adapted from Paper G

When we look at the architectural models in Papers A to F as an evolutionary process, we observe that all components and mechanisms included in the architectural model in Figure 4.7 can be found in earlier case studies. This shows that the generalised cognitive model for autonomous IoT device management that we present in this thesis unifies the insights from the included case studies with the theoretical study in the structured literature review, into one cohesive archetype.

4.2 Summary of auxiliary paper

In addition to the main publications listed above, the following auxiliary paper helped us to decide on a topic, define the scope of this thesis and learn about system architecture in IoT.

Paper H (Kraemer et al. 2017)

Title:
Fog Computing in Healthcare – A Review and Discussion
Published in:
IEEE Access, vol. 5, pp. 9206-9222

In this literature review, we present the first review on fog computing within healthcare informatics and explore, classify, and discuss different use cases presented in the literature.

We categorise applications into use case classes and list an inventory of application-specific tasks that can be handled by fog computing. In addition, we discuss on which level of the network such fog computing tasks can be executed and provide tradeoffs concerning requirements relevant to healthcare.

Our review indicated that when implementing systems in healthcare two opposite factors move computation either away from the device or away from the cloud: First, processing on higher network tiers is required due to constraints in wireless devices and the need to aggregate data. Second, privacy concerns and dependability prevent computation tasks from being moved to the cloud. Thus, there are a significant number of computing tasks in healthcare that require or can benefit from employing fog computing principles.

Chapter 5

Methodology

This thesis is based on the design science research methodology (DSRM) as it provides a sound structure for answering the research goal described in Section 1.3. We will now describe how we applied the research methodology in the PhD project.

5.1 Design Science Research Methodology

According to Wieringa [140], design science aims at improving a studied problem, by investigating an artefact as it interacts with the problem in its context. Hevner et al. [141] state further that the purpose of DSRM is to acquire more knowledge and a better understanding of the design problem and the prescribed solution, through the study of concrete artefacts. The methodology employed in this thesis is thus based on answering knowledge questions by investigating a set of design artefacts in a given context through a series of case studies and a structured literature review. However, similar to most creative exercises, DSRM is an iterative and unstructured process. This means that the activities were typically repeated several times and might not have followed exactly in the presented order.

DSRM is guided by a technical research goal, which is broken down into knowledge goals and design goals. These goals correspond in turn to research questions (RQs) and design problems (DPs), respectively. We will now describe the relationship between the research goal, the research questions, the design problems and the design artefacts.

5.1.1 Research Goal

The research goal of this thesis is to synthesise and describe a generalised architectural model for autonomous and perpetual management of IoT devices deployed in a non-stationary environment. This corresponds to the technical research goal.

5.1.2 Research Questions

In addition to the research goal, most research projects have knowledge goals. A knowledge goal can be phrased as a knowledge question, which asks for knowledge of the artefact in its context. Empirical knowledge questions require either qualitative or quantitive data to answer them, while analytical knowledge questions are based on a conceptual or logical analysis [140].

We observe that all our research questions fit the description for empirical knowledge questions.

5.1.3 Design Problems

In contrast to knowledge questions, a design problem is solved by (re)designing an artefact to achieve a design goal. A design goal can typically be classified either as an instrument design goal or an artefact design goal. In the former case, we design a research instrument aimed at answering a knowledge question, while in the latter case, we design an artefact aimed at studying a problem in a given context [140].

We observe that all our design problems are aimed at answering RQ 2 and 3, which are knowledge questions. However, by their nature, they are also studying a specific problem in a specified context. Thus our design problems can be classified both as instrument design goals and artefact design goals.

5.1.4 Design artefacts

In this thesis, three types of artefacts are of particular interest, namely architectural models, components and mechanisms. These artefacts are found on different levels in the architecture [140], where they model the behaviour of the system in different ways.

The architectural model describes the system-level phenomena, that is, the purpose of the system. This means that the system architecture models the overall system behaviour through the *organisation* of the components in the system.

At the middle layer, we find the components. They are characterised by their capability to respond to a stimulus, an event or some change in the environment. The system behaviour at this level can be modelled by the *interaction* between the components, that is, the sequence in which the components are activated.

At the lowest level, we find the mechanisms that guide the interaction between the components in the architecture. They produce stimuli, which again creates a pattern of data that flows between the system components. Thus, the behaviour can also be modelled by following the *data flow*, that is, the input and output from components.

5.2 Research activities

DSRM includes a set of activities that guide the research toward the goal. The first activity is to identify the problem context. Then comes the identification of research objectives. Next is the design and investigation of artefacts. Finally, we have knowledge extraction and communication of results. A framework for DSRM is illustrated in Figure 5.1.

In the following subsections, we will recount how we applied DSRM in our research, and how the methodology helped us synthesise the knowledge needed to reach the goal.

5.2.1 Step 1: Identification of problem context

The first phase in DSRM is the identification of challenges found in a specific problem context. This is an iterative process that we performed throughout the PhD project.

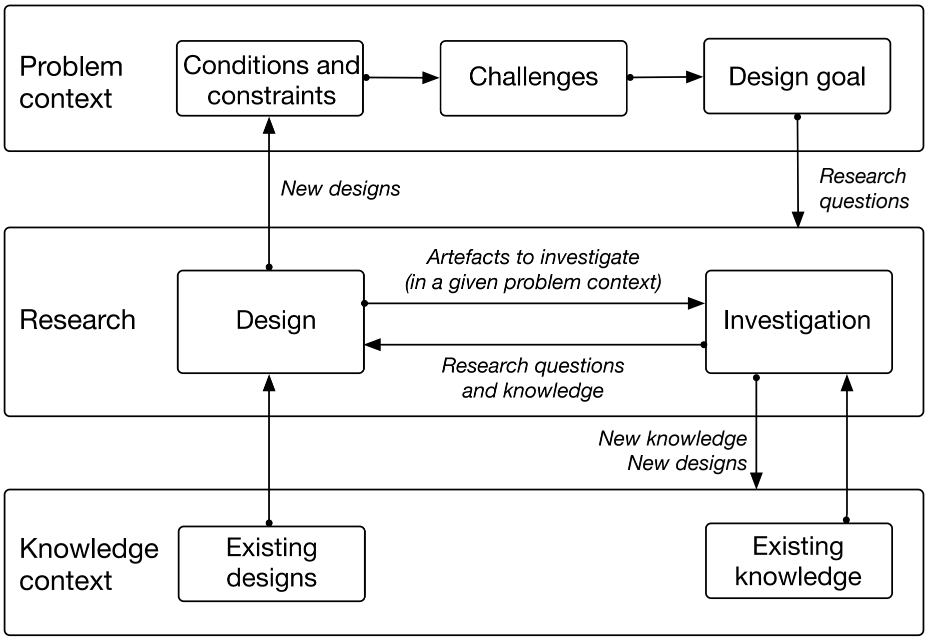

Figure 5.1: A framework for Design Science, adapted from [140]

In Paper A we identified the initial challenge, that is, how to keep constrained and energy-harvesting IoT devices in operation in a volatile and non-stationary environment. This particular challenge was further addressed in Papers B to D. In Papers E to G, we went beyond the initial challenge and discovered that cognitive management of constrained IoT devices is a complex problem involving several composite and interwoven challenges, as described in Section 1.1.

5.2.2 Step 2: Identification of research objectives

The next phase in DSRM is the process of identifying the research objectives, i.e., defining the technical research goal and the corresponding research questions.

The research goal emerged as a result of an iterative process. Already while working on the analytics engine in Paper B and C we decided that the main goal of this PhD project would be to define an architecture for cognitive IoT device management. Our initial idea was to explore different components and mechanisms through a series of case studies, and then put them together in a simulation. The strategy was thus based on the idea that the architecture would emerge as a result of the process in a bottom-up fashion.

However, while writing Papers E and F, we identified a gap in the existing literature regarding fundamental architectural questions related to autonomous IoT device management. For example, we found no clear answers to questions such as which components are required to impart the necessary intelligence to an autonomous IoT device manager, which adaptation mechanisms are most effective in assessing and analyzing a given situation or event, and how the internal communication should be described in such an architecture. Additionally, we could not find clear guidelines on the best practices for designing an AIDM management system.

To fill this knowledge gap we therefore decided to do a structured literature study where we explored, described and synthesised patterns and practices for autonomous IoT device management platforms and architectures in recent research literature. Based on this knowledge we could then design a *conceptual* model for a cognitive IoT device management solution, using a top-down approach.

As with the research goal, the knowledge goals and their respective research questions were gradually defined during the PhD project. RQ 1 followed naturally from the iterative identification of challenges in the specified problem context, as explained in Step 1. RQ 2 and 3 were defined while working on papers B and C. RQ 4 and RQ 5 were identified in conjunction with the work done in Papers F and G, that is, after we changed the research goal toward designing a conceptual architectural model. For reference, the research questions are summarised in Table 5.1.

Table 5.1: Overview of research questions addressed in Papers A to G.

Research question topic		Addressed in paper
RQ 1	Identifying challenges	A – G
RQ 2	Identifying system and adaptation components	A – G
RQ 3	Identifying adaptation mechanisms	B – G
RQ 4	Identifying patterns	G
RQ 5	Identifying best practices	G

5.2.3 Step 3: Design and investigation of artefacts

The third phase in DSRM is the design and investigation of specific artefacts to gather insights and knowledge about the problem in a given context. These are interlinked activities that were done iteratively through the work of Papers A to G. In each paper, we address a specific design problem, which in turn corresponds to a concrete design artefact. The main artefact of each paper is summarised in Table 5.2.

In Paper A we focused on DP 1. Here we designed and studied a solar-powered WSN platform for data collection and analytics. This platform contributed towards RQ 1 since it helped us identify the need for collecting meta-knowledge on device level to help devices stabilise their energy consumption against their energy intake. It also identified some basic architectural components for autonomous collection and analysis of data, which addresses RQ 2.

In Papers B to F, we designed five architectural models which correspond to DP 2 to 6. These artefacts were analysed to identify patterns, components, and mechanisms used to promote autonomous behaviour. This addressed RQ 1, 2 and 3, as follows: In Paper B we addressed DP 2, which corresponds to the testbed for exploring autonomy for constrained sensor nodes. Through the testbed, we explored several challenges regarding energy management, which in turn helped us define our

Table 5.2: Overview of design problems and design artefacts found in Paper A to G.

Design problem		Main design artefact	Mgmt type[1]	Val.[2]	Ana.[3]	Pap.	RQ
DP 1	Data analysis	Data collection and analysis platform	N+A	Dep.	Em.	A	1-2
DP 2	Resource offloading	Device management testbed	N+R	Dep.	Em.	B	1-3
DP 3	Resource optimisation	Learning and prediction algorithms	R	Sim.	St.	C	1-3
DP 4	Self-optimisation	Prediction tuning algorithms	R	Sim.	St.	D	1-3
DP 5	Self-management	Prediction model selection algorithm	R	Sim.	St.	E	1-3
DP 6	Self-management	Training data selection policy	R	Sim.	St.	F	1-3
RG	Cognitive IoT device management	Generalised architectural model	R	Rev.	An.	G	1-5

1) Management type: N = network, A = application, R = resource
2) Validation: Dep = deployment, Sim = simulation, Rev = Review
3) Analysis: Em = empirical, St = statistical, An = Analytical

first objectives. It also identified and validated the five basic components, the three autonomous loops in the architecture and the three types of adaptation mechanisms. In Papers C and D, we explored DP 3 and 4, respectively, through the study of different machine learning models and the mechanisms needed to train and tune them. These studies provided a better understanding of the challenges related to the operation of constrained, energy-harvesting devices located in a non-stationary environment. We also made simulations which demonstrated that machine learning is a viable method for autonomous prediction of future energy intake of such devices. In Paper E we explored DP 5. Here, we investigated a prediction model selection algorithm, which was used in a simulation to study how an autonomous manager can increase the accuracy of predictions in the initial period after deployment. In Paper F we investigated DP 6. Here, we employed a training data selection algorithm to solve the problem of identifying the node that could provide the most relevant data for transfer learning when deploying a new device into an existing IoT deployment in yet another simulation.

In Paper G, we structured and categorised the knowledge gathered through the structured literature review in tables, taxonomies, patterns and models. These artefacts were then analysed to answer RQ 4 and 5. In addition, the paper addresses RQ 1, 2 and 3, both from empirical analysis of the design artefacts provided in the six case studies in Papers A to F and through the analysis of architectural models found in industry and recent literature.

5.2.4 Step 4: Knowledge extraction

The fourth phase of DSRM is the extraction of new knowledge that is acquired through the research.

First, we successively analysed the results from each case study and discussed the implications in Papers A to F. In Paper G we synthesised the knowledge gathered from those six case studies, together with analysed data from 32 reviewed case studies and three architectural models for autonomous behaviour. The synthesis resulted in a generalised architectural model for autonomous, perpetual management of IoT devices. In addition, Paper G contributed the following knowledge: A taxonomy of adaptation mechanisms for autonomous IoT device management, a model of cognitive planning and a list of best practices to guide the design and implementation of cognitive IoT device management platforms.

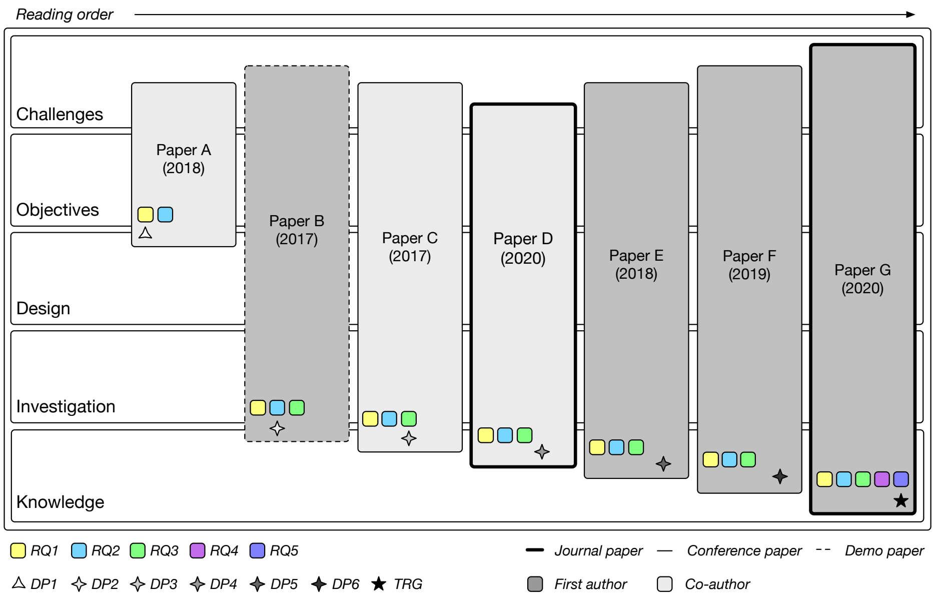

Figure 5.2: Relationship between publications, research questions (RQ), design problems (DP), technical research goal (TRG) and research methodology. Adapted from [142]

Figure 5.2 shows the relationship between the publications, the phases in the design science research methodology that they adhere to, and which research question and design problem they addressed.

Chapter 6

Research results

This chapter provides an overview of the main outcomes and contributions of this PhD thesis. First, in Section 6.1, we revisit the research goal and describe how it was achieved. Next, in Section 6.2, we present answers to each of the five major research questions. Then, we examine each design problem and how the artefacts assisted in resolving them in Section 6.3. Finally, in Section 6.4, we showcase how Veiga et al. [143] utilised the cognitive model we suggested in Paper G to solve a complex issue that matches the set of challenges and aligns with the patterns and best practices mentioned in the same article, thereby validating our synthesised conceptual architecture.

6.1 Research Goal

In Section 1.3 we defined the following research goal for this thesis:

Research goal

Synthesise and describe a generalised architectural model for autonomous and perpetual management of IoT devices deployed in a non-stationary environment.

The main contribution of the thesis is a conceptual model of a generalised service-oriented architecture for cognitive management of constrained IoT devices, which is shown in Figure 6.1. In essence, the model is a service-oriented architectural blueprint that describes the interaction between the stored knowledge in a system and the adaptation processes that are needed to provide cognitive behaviour, i.e., reasoning, learning and planning. We will now explain the basic aspects of the generalised model. A full description can be found in Paper G.

The model is structured around the physical devices, and two distinct types of managers, which serve the device and the system perspectives, respectively. *Devices* are responsible for any interaction with the real world, based on a pre-defined policy or action plan. For each physical device in the model, there is exactly one instance of a *device manager*, following the pattern of digital twins. The device manager contains information about the device and data concerning its past operation. With this knowledge, the device manager can predict the future state of the device it mirrors. In turn, these predictions allow the device manager to evaluate and plan the operation of the device with better precision. The *system manager* contains knowledge about the system, its purpose, the environment in which the devices are placed, and how this particular environment influences the operation of the devices. Its main responsibility is to analyse the past, monitor the present and predict the future states on the system level.

This distinction between system and device supports individual adaptation for each device and ensures perpetual operation for the system as a whole.

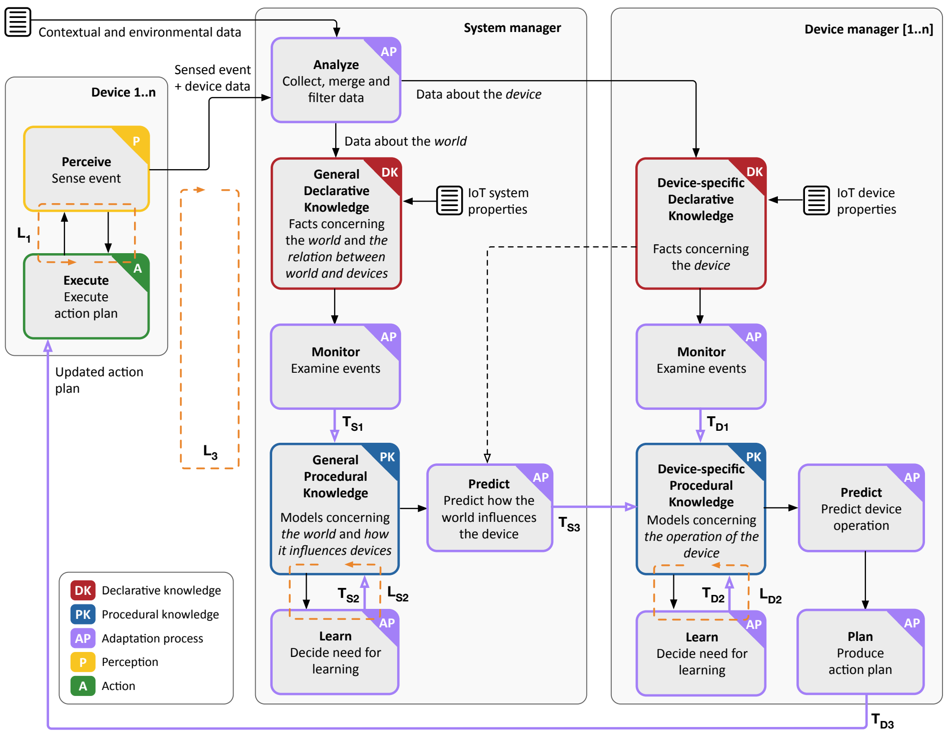
Contextual and environmental data
System manager
Device manager [1..n]
Device 1..n
Sensed event + device data
Analyze Collect, merge and filter data
Data about the device
Data about the world
Perceive Sense event
General Declarative Knowledge Facts concerning the world and the relation between world and devices
IoT system properties
Device-specific Declarative Knowledge Facts concerning the device
IoT device properties
L1
Execute Execute action plan
Monitor Examine events
Updated action plan
L3
TS1
TD1
General Procedural Knowledge Models concerning the world and how it influences devices
Predict Predict how the world influences the device
TS3
Device-specific Procedural Knowledge Models concerning the operation of the device
Predict Predict device operation
DK Declarative knowledge
PK Procedural knowledge
AP Adaptation process
P Perception
A Action
TS2
LS2
Learn Decide need for learning
TD2
LD2
Plan Produce action plan
TD3

Figure 6.1: Component diagram of the proposed architectural model for autonomous, perpetual management of IoT devices. Adapted from Paper G.

On the top level, devices contain components that are responsible for Perception (P) and Action (A), while managers consist of components that handle adaptation processes (AP), declarative knowledge (DK) and procedural knowledge (PK). Adaptation processes are then divided into monitoring, analysing, predicting, learning and planning mechanisms. The separation between declarative and procedural knowledge is beneficial as different types of knowledge often use different implementations. In the context of machine learning, for instance, this means that declarative knowledge can be stored in the form of training data, and procedural knowledge can be represented by trained machine learning models.

Further, the model contains four control loops that handle adaptive behaviour in the system autonomously. L_1 controls each device based on the last instructions received from the device manager. L_{S2} and L_{D2} are learning loops on the system and device level, respectively. They act on incoming events and make decisions whether any new situation calls for a new learning cycle. L_3 is the main control-loop. It monitors the situation and detects any sudden events or changes in the environment that happen on either the system or device level. On detection, it runs an analysis of the situation and decides if any of the devices need to adjust their operation. Note that the model does not prohibit the inclusion of more control loops. For instance, a hypothetical control loop L_4 between the prediction and planning components could enable reinforcement learning.

Adaptive actions are started by triggers that guide the data flow to the right component and regulate the behaviour of the managed system. They are also crucial for controlling parallelism, which can cause errors if not kept in check. In our generalised model we have identified six particular triggers: T_{S1} triggers when a change in the environment happens, e.g., a change in the weather forecast, that might influence the operation of the devices, while T_{D1} activates if the device manager observes an internal change in the device that might affect its operation. T_{S2} and T_{D2} initiate the learning process for the system and device manager, respectively. Examples of triggers that might activate the learning process can be previously unseen events, sudden changes in the environment or the discovery that an executed plan did not have the anticipated effect. T_{S3} triggers if the result of a prediction shows a need for adaptation for one or more devices. T_{D3} informs the device when a new plan is made, i.e., which adaptive action it needs to take.

As stated earlier, the proposed architectural model was derived from two different sources. First, we gathered insights through six case studies. This provided insights regarding the type of components and mechanisms that are needed for cognitive management, and how to build a basic structure and pipeline for the data flow in the proposed architecture. An overview of components and mechanisms identified in each case study is summarised in Table 6.1. Second, we analysed a set of models described in recent literature that solved similar challenges as those described in Section 1.1, and synthesised knowledge about adaptation mechanisms, architectural patterns and best practices from them. This process is fully described in Paper G.

From the 6 case studies and the literature review, we identified and validated the three system components, the five adaptation components, the five adaptation processes, the three types of adaptation mechanisms and the three types of control loops that are needed for the cognitive management of IoT devices. This gave us the knowledge necessary to describe the direction and control of the data flow in the proposed architectural component model. Thus, the final result of this thesis is a new artefact that can enter a new cycle of investigation and knowledge extraction, as described in Chapter 5.

Although other works have described each part of this model separately, to our knowledge we were the first research group to put it all together in one comprehensive, cognitive model, in the context of IoT.

6.2 Research Questions

As detailed in Section 1.4 and Section 5.2.2 this thesis aimed at answering five research questions. In this section, we will describe how each research question contributes towards the research goal.

6.2.1 Challenges

RQ 1

What are the contextual and architectural challenges that must be addressed to realise the research goal?

How to identify the specific challenges in a problem context is fundamental to any research. In Section 1.1 and Section 5.2.1 we gave a general overview of the challenges that we addressed in this PhD thesis.

In Paper A we discovered the initial challenge regarding resource allocation and management for constrained, solar-powered IoT devices. Through Papers B to F, we gradually expanded our understanding of the challenges, both on an operational and architectural level. In Paper G, we concluded that autonomous management of constrained IoT devices is a complex problem that is related to various factors. These factors include the context in which the devices operate, the device topology, available resources, the scale of the deployment, and the problem that the system aims to solve.

Contributions by paper:

- In Paper A we presented a discussion on the need for monitoring energy intake and consumption to estimate battery depletion of solar-powered IoT devices.
- Papers B to F gave insights into the *contextual* and *architectural* challenges that must be addressed to build a generalised architectural model for an AIDM-system.
- In Paper G we synthesised the knowledge gathered from the case studies in Papers B to F and the 32 papers included in the structured literature review, and described each challenge in detail.

6.2.2 System and Adaptation Components

RQ 2

Which system components and adaptation components are required to reach the research goal?

As described in Section 5.2.2, identifying which components to include in the generalised architectural model was an iterative process that were performed during the work on Papers A to F. The whole set of components was fully defined in Paper G.

At the system level, we have identified three components that exhibit adaptive behaviour, namely ‘Device’, ‘Device Manager’, and ‘System Manager’. Moving to the process level, we have found five components that guide adaptation. These components are ‘Perception’, ‘Action’, ‘Adaptation Process’, ‘Declarative Knowledge’, and ‘Procedural Knowledge’. The adaptation process component can be further divided into five different categories that describe the type of adaptation that the component is responsible for. These are ‘Monitor’, ‘Analyse’, ‘Predict’, ‘Learn’ and ‘Plan’. Note that in Paper G we originally classified ‘Predict’ as an ‘Analyse’ component, in line with the MAPE-K model. However, while writing this thesis we concluded that these are two distinct components, which the notation should reflect. This also makes it easier to read the diagrams. An overview of the components that we explored and analysed in the six case studies can be found in

Table 6.1: Overview of included components, processes and mechanisms in the architectural model described in case studies

Paper	SC[1]	AC[2]	APT[3]	Adapt. Mechanism	Purpose
A	D	P	—	Perceive	Sense event
A	D	A	—	Execute	Execute sensing policy
A	S	AP	Mo	Reasoning mechanism	Visualise data
A	S	AP	An	Reasoning mechanism	Merge and filter data
A	S	AP	An	Reasoning mechanism	Analyse data
A	S	DK	—	Data storage	Share data
B	D	P	—	Perceive	Sense event
B	D	A	—	Execute	Execute sensing policy
B	DM	AP	Mo	Reasoning mechanism	Visualise data
B	DM	AP	An	Reasoning mechanism	Collect, merge and filter data
B	DM	AP	Pr	Reasoning mechanism	Predict solar energy input
B	DM	AP	Le	Learning mechanism	Retrain ML models
B	DM	AP	Pl	Planning mechanism	Define sensing cycle policy
B	DM	DK	—	Data storage	Share data
B	DM	PK	—	Prediction model	Share prediction procedure
C	D	P	—	Perceive	Sense event
C	D	A	—	Execute	Execute sensing policy
C	DM	AP	An	Reasoning mechanism	Collect, merge and filter data
C	DM	AP	Pr	Reasoning mechanism	Predict solar energy input
C	DM	AP	Le	Learning mechanism	Retrain ML models
C	DM	AP	Pl	Planning mechanism	Define sensing cycle policy
C	DM	DK	—	Data storage	Share data
C	DM	PK	—	Prediction models	Share prediction procedure
D	D	P	—	Perceive	Sense
D	D	A	—	Execute	Execute sensing policy
D	D	AP	Pl	Planning mechanism	Define sensing cycle policy
D	SM	AP	An	Reasoning mechanism	Collect, merge and filter data
D	SM	AP	An	Reasoning mechanism	Tune ML models
D	SM	AP	Pr	Reasoning mechanism	Predict solar energy input
D	SM	AP	Le	Learning mechanism	Retrain ML models
D	SM	DK	—	Data storage	Share data
D	SM	PK	—	Prediction models	Share prediction procedure
E	D	P	—	Perceive	Sense event
E	D	A	—	Execute	Execute sensing policy
E	DM	AP	An	Reasoning mechanism	Collect, merge and filter data
E	DM	AP	An	Reasoning mechanism	Analyse accuracy of prediction models
E	DM	AP	An	Reasoning mechanism	Select energy prediction model
E	DM	AP	Pr	Reasoning mechanism	Predict solar energy input
E	DM	AP	Le	Learning mechanism	Retrain ML models
E	DM	AP	Pl	Planning mechanism	Define sensing cycle policy
E	DM	DK	—	Data storage	Share data
E	DM	PK	—	Prediction models	Share prediction procedures
F	D	P	—	Perceive	Sense event
F	D	A	—	Execute	Execute sensing policy
F	DM	AP	Mo	Reasoning mechanism	Identify changes in training data
F	DM	AP	Pr	Reasoning mechanism	Predict energy input for device
F	DM	AP	Le	Learning mechanism	Retrain ML models
F	DM	AP	Pl	Planning mechanism	Define new policy for operation
F	DM	DK	—	Data storage	Share device and transfer learning data
F	DM	PK	—	Prediction model	Share prediction procedure
F	SM	AP	An	Reasoning mechanism	Collect, merge and filter data
F	SM	AP	An	Reasoning mechanism	Collect transfer learning data
F	SM	DK	—	Data storage	Share weather forecasts
F	SM	PK	—	Train. data select. policy	Identify correlated device

1) System component: D = Device, S = Server, DM = Device Manager, SM = System Manager
2) Adaptation component: P = Perception, A = Action, AP = Adaptation process, DK = Declarative knowledge, PK = Procedural knowledge
3 Adaptation process type: Mo = Monitor, An = Analyse, Pr = Predict, Le = Learn, Pl = Plan

columns 2 to 4 in Table 6.1.

Contributions by paper:

- In Paper A we show containers for four architectural components for autonomous collection and analysis of data found in the generalised model, namely perception, action, process and declarative knowledge.
- Paper B and C include a deployment model and an architectural model, respectively, of the server backend. In these papers we identify the device

manager as a separate system component that reflects the physical device. The model also defines all five top-level adaptation components.

- Paper D identifies the System Manager as an entity that differs from the Device Manager. While the latter is concerned with the operation of each device, the former is concerned with models concerning the world and how they influence the devices in general.
- Paper E contains a model that describes the basic architecture for energy workflow management. The model shows the need to include components that can reason about the devices in light of the context, i.e., situational awareness.
- In Paper F we describe and discuss how a learning manager and a planning manager in a system can support energy planning for constrained IoT devices. We also show that the adaptation process emerges from the interaction between the system manager and device manager components. In addition, we touch upon the need for controlling data flow between components with triggers.
- Paper G shows the full range of components that are needed to build an architectural model for cognitive management of constrained IoT devices, and the interaction between them, as described above.

6.2.3 Adaptation Mechanisms

RQ 3

Which adaptation mechanisms can be used to achieve the research goal?

In Papers B to F we explored different adaptation mechanisms that can contribute towards the design of an architectural model for the cognitive management of constrained IoT devices. Paper B explored this in an empirical setting, while Papers C to F used simulations to study the behaviour of selected adaptation mechanisms.

The basic mechanisms needed for adaptation, i.e., monitor, analyse, predict, learn and plan, were identified already in Papers B and C. Throughout the case studies these mechanisms were then refined and generalised into three distinct categories, namely ‘Reasoning’, ‘Learning’ and ‘Planning’. However, these terms were only identified through the structured literature review we performed in Paper G, after studying the 32 case studies in light of our research. The mechanisms that are present in the final architectural model are thus a result of both our case studies and the case studies in the literature review.

A taxonomy of the adaptation mechanisms that are used to support cognitive device management in IoT, and how they relate to the proposed architectural model is detailed in Section G.4. Note that on the component level, the reasoning mechanism corresponds to the monitoring, analysis and prediction processes. Through our analysis, we found that reasoning and learning mechanisms can be categorised into three sub-level mechanisms that reflect the underlying principles that are used to infer situational awareness. These are model-driven mechanisms, semantic mechanisms and data-driven mechanisms. Within those categories we found eight different types of implementation in the literature: Linear and nonlinear

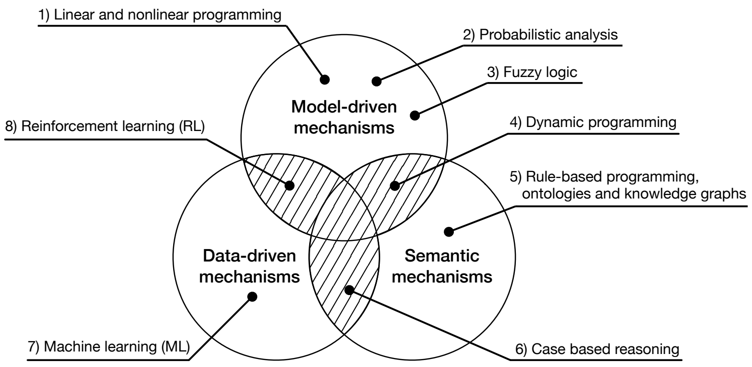
1) Linear and nonlinear programming
2) Probabilistic analysis
3) Fuzzy logic
Model-driven mechanisms
8) Reinforcement learning (RL)
4) Dynamic programming
5) Rule-based programming, ontologies and knowledge graphs
Data-driven mechanisms
Semantic mechanisms
7) Machine learning (ML)
6) Case based reasoning

Figure 6.2: Venn-diagram showing the relationship between reasoning and learning mechanisms that are used in IoT device management. From Paper G

programming; probabilistic analysis, Markov-modeling and Bayesian inference; fuzzy logic; dynamic programming and recursive optimisation; rule-based inference, ontologies and knowledge graphs; case-based reasoning; machine learning; and reinforcement learning. The relationship between the three sub-level mechanisms and the eight types of implementation can be seen in Figure 6.2.

The thesis has focused mostly on the top-level adaptation mechanisms that are contained in the adaptation component since these are the ones that drive the adaptation processes. However, as shown in Table 6.1, all adaptation components include adaptation mechanisms. In our models, we have identified five other examples of adaptation mechanisms contained in an adaptation component, i.e., 'Perceive, 'Execute', 'Data storage', 'Prediction model' and 'Training data selection policy'.

Contributions by paper:

- In Papers B and C we show that autonomous adaptation is driven by control loops, which in turn is controlled by adaptation mechanisms contained in adaptation components.

- In Paper D, E and F we demonstrate and discuss different adaptation mechanisms that can support autonomous IoT device management. We also validate how these adaptation mechanisms can guide the adaptation processes on an architectural level.

- Paper G identifies the taxonomy of reasoning mechanisms, on three different levels. It also demonstrates how these mechanisms guide the adaptation processes in an architectural model for cognitive IoT device management.

6.2.4 Patterns

RQ 4

Which patterns can we identify in existing autonomous architectural models, in industry or recent research, that can guide us toward the research goal?

To answer this question we performed a structured literature review, which is described in detail in Paper G.

In the review, we first analysed 32 models of autonomous IoT device management platforms found in recent research. The most obvious pattern is that the managing process is centralised, either in a star or cluster topology. Both these topologies support constrained devices that typically do not have access to either the resources or the information that is necessary to reason about their operation or to learn from experiences. However, centralised managers seem to be better suited for knowledge distribution.

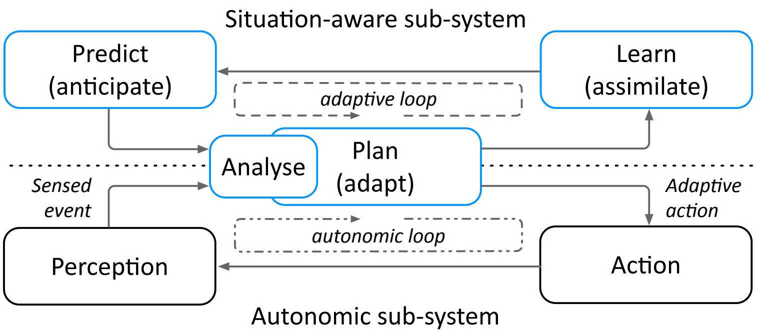

Figure 6.3: A model of cognitive planning. Adapted from Vernon's cognitive cycle [29].

On the component level, we identified several patterns regarding adaptation mechanisms. First, mechanisms that are required for understanding a situation, acquiring new knowledge or making decisions are placed in separate components. Second, reasoning mechanisms are used to guide the internal data flow in the architecture and are typically activated by observed events, internal processes, or predictions. Third, learning mechanisms are usually placed in the situation-aware part of the system. They are most often preceded by a reasoning mechanism and triggered either by a previously unseen event or an internal decision to retrain the machine-learning models. Fourth, it is a common strategy to employ a combination of reasoning and learning mechanisms and disperse them throughout the architecture. Fifth, planning mechanisms are usually placed in a central, coordinating role on the device level to do task allocation or policy management. Together, these strategies ensure modularization and support separation of concerns, which in turn reduces the complexity of a system.

After reviewing the 32 case studies, we made a comparative and complementary analysis of autonomic and cognitive models found in research and industry. This analysis revealed three models that inspired our work. First, we identified the MAPE-K as described by IBM [49] loop. This drives the interaction between the components in the generalised architectural model proposed in this thesis. Second, we saw that the cognitive cycle presented by Vernon [29] could be adapted into the *cognitive planning* model shown in Figure 6.3. From this, we added the two control loops that drive cognitive behaviour, namely the autonomic loop and the adaptive loop. Third, the standard model of human-like minds, presented by Laird et al. [40], showed the benefit of separating declarative knowledge from procedural knowledge. In addition, their model confirmed the need for separate, autonomous control loops for learning. For reference, an adapted version of this model is shown in Figure 6.4.

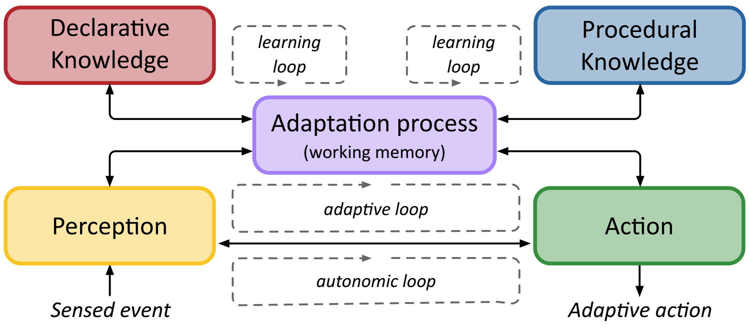

Figure 6.4: A standard model of human-like minds. Adapted from [40].

The knowledge gathered from the six included case studies was then combined with the patterns identified in the review. This provided the basis for the generalised architectural model, described in Section G.7.

For clarification and validation, we have redesigned the models described in Papers A to F, using the notation described in Paper G. These models can be found in Chapter 4. We observe that the identified patterns become more prominent with increasing complexity. Laird et al. claim that adaptation emerges from a combination of the implemented architecture, acquired knowledge and learned skills. In the generalised architectural model, this corresponds to the interaction between the components, guided by autonomous control loops and adaptation mechanisms

Contributions in paper G:

- This paper describes patterns for autonomous behaviour found in 32 recent case studies on IoT device management and three architectural models.
- Further, it presents a distinct cognitive model, based on Vernon's cognitive cycle, which we chose to name 'cognitive planning'.
- Finally, the paper incorporates the identified patterns in a generalised architectural model for cognitive IoT device management.

6.2.5 Best Practices

RQ 5

Which best practices can we identify by analysing existing implementations of autonomous architectures, in industry or recent research, that can help us reach the research goal?

This research question was also answered through the structured literature review, as detailed in Paper G. The review identified five best practices that characterise autonomous device management in IoT. In the following, we will summarise these practices, and show how we incorporated them into the proposed architecture.

BP1: Employ adaptation and reasoning mechanisms in accordance to environmental stationarity

As demonstrated earlier, adaptation requires a mechanism that is capable of situation awareness, that is, analysing the implications an event will have in the context of the situation. However, often the operational context of the devices must also be taken into account when deciding which type of adaptation mechanism to employ. In a stationary environment, a purely autonomic system may suffice, while for systems operating in non-stationary environments, the added complexity might demand the addition of a self-aware subsystem to the architecture.

Through the work of this thesis, we have learned that machine learning is a suitable mechanism when applied to challenges related to adaptation in non-stationary environments. Our case studies confirmed that machine learning can be applied to cognitive tasks like resource-efficient estimation in complex environments, self-management, intelligent decision making and as input to autonomous planning processes.

BP2: Select System topology according to the inherent systemic constraints and requirements

Our review showed that the obvious topology for constrained devices is either star or cluster-based. A star topology is often a better fit for systems that need added elasticity since centralised management makes it easier to distribute a set of resources over many devices. This topology is also preferable if there is a need to reduce the complexity of the system. Thus, a star topology will often be a good choice for systems that require variable access to processing power, memory or storage. In contrast, a cluster topology might be more suitable if the goal is to reduce latency or increase dependability. It is also a good match when it is necessary to handle high variance in networking conditions and when distinct manager nodes are specialised for specific management tasks.

In this thesis, we have consequently employed the star topology in our architectural models since it is a good match for resource management. However, the generalised cognitive model can easily be adapted to support a cluster topology, since each device manager is responsible for exactly one device. On the system level, it is possible to expand such an architecture with sibling managers, where each system manager is responsible for a group of devices, or employ a hierarchical topology where one system manager acts as a parent to a group of system managers.

BP3: Separate concerns and reduce complexity with modularization
Modularisation, also known as microservices, is an architectural style that allows larger applications to be split into smaller, independent components, where each component is responsible for one part of the system.

As described in Section 6.2.4, we observed that a majority of the reviewed architectural models divided reasoning, learning and planning into separate components. This became more prominent with increasing complexity. Furthermore, the reasoning and learning components were typically dispersed throughout the architectural models, while the planning component was placed in a central, coordinated role. This ensures that every device connected to the manager is updated according to the most recent knowledge.

We also observed that architectures aimed at managing devices in non-stationary environments usually showed a higher degree of modularisation than architectural models of systems operating under stable conditions. This allows better control of the data flow and easier control of the different states of the system. In addition, modularisation makes it easier to replace or extend parts of the architecture if the requirements or understanding of the system change.

The generalised model proposed in this thesis employs modularisation in combination with a black-box structure. Thus, the architecture inherently provides the flexibility of choosing the mechanism that is most suited to solve the given problem inside each component The notation also allows any number of reasoning mechanisms to be placed in any order throughout the architecture.

BP4: Control parallelism and data flow with triggers
In complex autonomous management systems, parallel data flows can lead to uncoordinated state changes, especially when different adaptation processes are active in several components at the same time. Hence, architectural models need to include descriptions of how these processes should be controlled, i.e., in which order the components are activated and how the data flows between the components after activation. In addition, the architecture needs to allow for activation both from the producer and consumer of knowledge.

In this thesis, we propose using triggers for controlling parallelism. In our notation, a trigger presents explicitly both the knowledge that is transferred, the direction of the transfer and how this knowledge in turn affects subsequent components.

BP5: Represent devices by digital twins
When a virtual device has a physical counterpart that it mimics, it is known as a digital twin. In addition to acting as a digital copy of the physical device, it can also support simulation, decision-making and control of its twin.

A digital twin contains at minimum the current state of the mirrored device but is often expanded with a record that holds all historical data, a model that describes its possible actions and interactions and even ways to predict future states and events. Thus, digital twins make it possible for device managers to keep track of the past, current and future state of each device individually. This allows to model the behaviour of the devices individually, which in turn makes it easier for the device manager to adjust the operation of each device based on their experience.

In the generalised cognitive model, the device manager is based on this concept. The division between declarative and procedural knowledge makes it easy to construct a digital twin on the level of complexity that is needed for perpetual operation.

6.3 Design Problems

In Section 1.5 we defined 6 design problems targeted at helping us investigate RQ 2 and RQ 3. For each of these design problems, we designed corresponding design artefacts, as described in Section 5.2.3. We will now describe how each of these design problems contributes towards the research questions.

6.3.1 Data analysis

DP 1

Design a platform that supports data collection and analysis.

Our first design artefact was a solar-powered WSN platform for data collection and analytics, which is described in Paper A. The architectural model contained four of the five system components for autonomous collection and analysis of data that is present in the generalised model, namely perception, action, process and knowledge. This was relevant for RQ 2. Although the model includes components for collecting, merging, filtering and analysing data, these are not aimed at resource allocation or device adaptation. Thus, this design problem did not contribute towards RQ 3.

6.3.2 Resource Offloading

DP 2

Design a platform that supports resource offloading.

In Paper B we demonstrated a testbed for exploring autonomy for constrained sensor nodes. In particular, we showed how to conduct resource offloading from constrained IoT devices to a central device manager connected through a wireless sensor network. Through the architectural blueprint of the testbed, we identified and validated the one-to-one connection between a device and a corresponding device manager. It also helped us to identify the three adaptive components found in the device manager in the generalised model, i.e., adaptive processes, declarative knowledge and procedural knowledge. In addition, the model included some basic examples of adaptive processes and autonomous loops. The design artefact was thus helpful in answering RQ 2, and to a lesser degree, RQ 3.

6.3.3 Resource Optimisation

DP 3

Design a mechanism that supports resource optimisation.

In Paper C we focused on DP 3. Here we designed a pipeline used to explore different mechanisms for training machine-learning models, predicting the energy buffer for constrained IoT devices, and planning day-ahead operations.

The pipeline was validated in a simulation, using real data collected via the testbed described in Paper B. This provided a good foundation for understanding the mechanisms a device manager needs to provide resource optimisation services on behalf of a constrained IoT device.

On an architectural level, the design artefact showed that the ability to predict future energy is split between two connected components, namely a component that contains the procedural knowledge that describes how to perform a prediction and a reasoning mechanism responsible for conducting the actual prediction. Thus, this design artefact contributed more towards answering RQ 3 than RQ 2.

6.3.4 Self-tuning

DP 4

Design a mechanism that supports self-tuning of learning processes.

To investigate DP 4 we designed a variety of mechanisms used to explore autonomous tuning of machine-learning models directed at day-ahead predictions for energy intake from constrained, solar-powered IoT devices. In particular, we investigated how to prepare training data, which features are most valuable, which sampling intervals yield the best results, and which machine learning models are most suitable.

The algorithms were run in a series of simulations, each aimed at investigating one aspect of the problem. The statistical analysis performed on the results from the simulation helped us gain a better understanding of both the components and mechanisms needed for self-tuning, which is a form of self-optimisation, on the system level. This addressed RQ 2 and RQ 3, respectively.

6.3.5 Self-management

DP 5

Design a mechanism that supports self-management when deploying a new system.

DP 5 was directed at ‘the bootstrapping problem’, that is, the tendency that machine-learning predictors have low accuracy just after a system is deployed due to a lack of training data. This implies that newly deployed devices risk spending more energy than they can harvest, which in turn is a threat to perpetual operation.

To investigate this problem we designed an autonomous prediction model switching algorithm aimed at increasing the prediction accuracy when available training data is scarce. The algorithm was run in a series of simulations, each covering a different period. This allowed us to study how the precision of the algorithms evolved.

Our results showed that an architecture centred around an autonomous prediction model selection mechanism can increase the general robustness of the system in two ways. First, by increasing prediction accuracy when available training data is limited, and second, by providing some self-healing capabilities when a device experiences a sudden loss of training data after deployment The architectural model of this artefact also provided insights into the internal communication between the components and showed how to use triggers to control the data flow. DP 5 was therefore relevant for answering both RQ 2 and RQ 3.

DP 6

Design a mechanism that supports self-management when deploying new devices into an existing IoT deployment.

Self-management is also the theme of DP 6. In Paper F we implemented an autonomous transfer learning data selection mechanism that increases prediction accuracy for devices that are deployed into an existing WSN by autonomously selecting suitable training data for newly deployed devices. The mechanism was split into two algorithms. The first algorithm looked at correlated data to identify an existing device that resembles the newly deployed device. The second algorithm copied the data from the device manager representing the correlated device to the device manager representing the new device.

To validate the concept, the algorithms were run in a series of simulations. Data from the simulations were then analysed using statistical difference methods. Our results showed that autonomous transfer learning data selection can support self-management by improving its ability to perform self-optimisation, self-healing and self-protection.

The architectural blueprint of the design artefact shows that to solve complex tasks autonomously all the identified system-level components, i.e., the device, the device manager and the system manager, are needed. Furthermore, we see that splitting collected data into *general* and *device-specific* knowledge, allows reasoning mechanisms to handle complexity on different levels simultaneously. In addition, the model includes all five types of adaptive processes, that is, monitor, analyse, learn, predict and plan. The communication between these components is guided by the three autonomous loops: the autonomic loop, the learning loop and the adaptive loop. This data flow is managed by triggers, which in turn control parallelism. Thus, DP 6 contributes both towards RQ 2 and RQ 3.

6.4 Architectural Model Validation

As stated earlier, the objective of this thesis is to design a *conceptual* architectural model of a cognitive manager for constrained IoT devices. To validate this model, we did a theoretical analysis of various case studies that addressed similar challenges. However, in 2023, Veiga et al. [143] worked on a different stream of research where they conducted a study that addressed challenges similar to those mentioned in Paper G. They therefore decided to employ the cognitive model we proposed in that paper as a blueprint for their implementation of an IoT application to count the number of people in a skiing area using a camera sensor. In the following, we will summarise their research, discuss how they implemented their solution based on the proposed general model, and highlight the architectural challenges they faced.

The main focus of the research conducted by Veiga et al. was to examine how to improve the utility of a system by prioritising the data that is most valuable, otherwise known as 'value of information' [144]. The general idea behind this concept is to only spend resources on data that are relevant for the system functionality or needed for perpetual operation. This requires that the system can make decisions about its actions regarding data collection, processing, storage and transmission. However, this type of knowledge is often specific to the goal of the application, the sensed events, the environment, and the construction of the device. In addition, data may vary between individual devices, due to different configurations or environmental placement, and also over time, because sensed phenomena and external conditions may be non-stationary.

In a case study, Veiga et al. counted the number of skiers that pass in front of a camera while minimising the resources needed for operation. To this end, they proposed a solution where a device manager that employs artificial intelligence instructs sensors to adjust their operation according to the situation by employing a technique known as visual attention models [121]. This solution required a self-adaptive application that could learn, maintain and distribute the visual attention models. In addition, to reuse proven AI solutions the IoT platform had to support the creation of pipelines where components are based on containers. Since these prerequisites match the patterns and best practices identified in Paper G, Veiga et al. elected to base the implementation of their solution on the general cognitive model for IoT applications that we proposed. A diagram of their simplified cognitive architecture is shown in Figure 6.5.

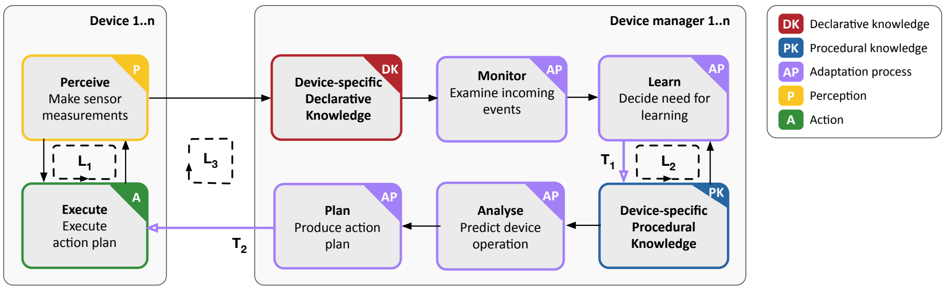
Device 1..n
Perceive Make sensor measurements
P
Execute Execute action plan
A
L1
L3
Device manager 1..n
Device-specific Declarative Knowledge
DK
Monitor Examine incoming events
AP
Learn Decide need for learning
AP
T1
L2
Device-specific Procedural Knowledge
PK
Analyse Predict device operation
AP
Plan Produce action plan
AP
T2
DK Declarative knowledge
PK Procedural knowledge
AP Adaptation process
P Perception
A Action

Figure 6.5: Cognitive architectural model of the proposed camera application, following the reference architecture in Paper G. Adapted from [143]

The architectural model designed by Veiga et al. shows the communication between a device and its respective device manager. Since their solution made use of just one device, there was no need for implementing adaptation mechanisms on the system level. Hence the system manager is not included in the model. The model shows further that the device is composed of two subcomponents, namely Perceive and Execute. We can also see that the device manager contains four components for handling the adaptation processes (AP), one component for specifying declarative knowledge (DK) and one component for describing procedural knowledge (PK). The four adaptation processes are divided into Monitor, Learn, Analyse and Plan, respectively. This setup is consistent with the architectural model we propose.

Further, the modified architectural model shows that data flows between the components in three separate loops. These are an autonomic loop for the device (L1), a learning loop for the device manager (L2) and an adaptive loop that guides the interaction between the device and the device manager (L3). L1 controls the data flow between the Perceive and Execute components inside the device component. This allows the device some degree of autonomy as it can operate even if the device manager fails to send a refreshed visual attention model. L2 is located within the device manager. It controls the learning processes for the visual attention model, such as updated knowledge from reconstructed images and temporal models describing the number of persons detected for each hour of each weekday. L3 is the main adaptive control loop. It is responsible for uploading updated attention models to the device when necessary. Typically, this happens after a learning process caused by a sensed event makes adjustments to the action plan, i.e., an adaptation of the device operation. We observe that the control loops are in alignment with the three loops found in our generalised architectural model.

Finally, we observe that Veiga et al. emphasise the use of triggers to control the data flow and only dispatch resource-intensive tasks like learning when it is necessary. They have identified two such triggers in their architecture: T1 activates the learning process at regular intervals, i.e., once a day, or when new data becomes available. T2 transmits an updated visual attention model to the device if the updated model is significantly different from the previous version. This is in line with the patterns and best practices that we have identified.

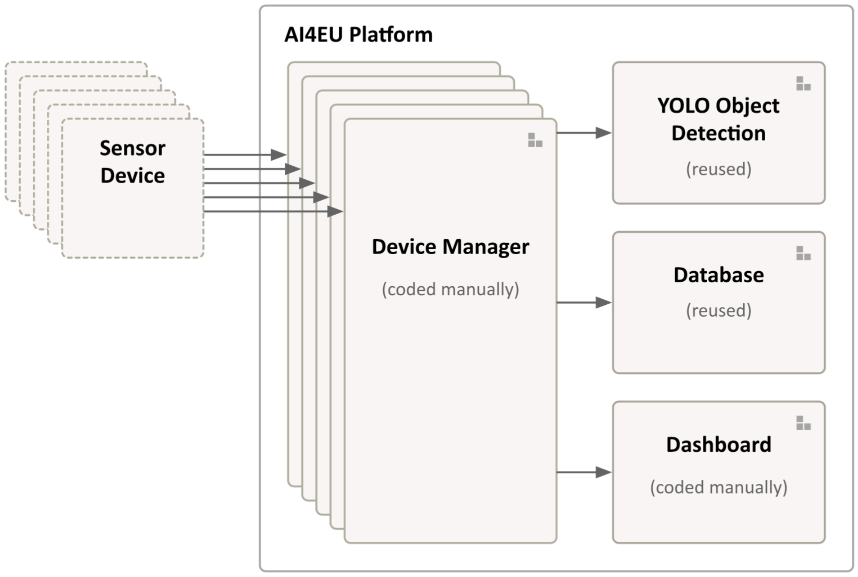

Figure 6.6: Container structure for the deployed solution based on the AI4EU platform [143].

A diagram of the deployed solution can be seen in Figure 6.6. The implementation consists of a container-based solution based on the AI4EU platform. Their pipeline consists of a sensor device, which communicates with an object detection component, a database component and a dashboard component through the device manager. The object detection component follows the ‘You only look once approach’ (YOLO). When an object passes in front of the camera, the data is transmitted through the device manager to an image recognition component in the object detection component. This approach follows the principle of modularisation, which we have identified as a best-practice.

Veiga et al. conclude that since deployment platforms like the AI4EU can integrate core features of cognitive IoT architectures, they have the potential to facilitate the deployment process and simplify the implementation of IoT solutions that contain AI components. However, while this case study demonstrates the benefits of giving IoT devices access to operational meta-knowledge, it also shows that current IoT platforms lack certain features that make it difficult to deploy a modular cognitive architecture. The identified gaps are related to the management of multiple containers, the integration of IoT devices in the deployment process, the lack of built-in support for including explicit triggers in the platforms’ editing tools, and the lack of template architectures in the platforms.

As shown above, the model and implementation demonstrated by Veiga et al. closely follow the conceptual architectural model and best practices described in Paper G. This validates that the proposed architectural model is general enough to also cover other domains than energy budget planning for solar-powered devices.

Chapter 7

Conclusion

In this chapter we first revisit the challenges, research goal and research questions that we addressed in the PhD thesis. Then we briefly describe the research methodology, before we list the contributions. Finally, we indicate possible future directions for research.

7.1 Concluding Remarks

The thesis addresses the problem of managing a myriad of distinct and constrained IoT devices operating in dynamic contexts. This is a highly complex problem, related to the resources that the devices possess, the scale of the deployment, the device topology, the context in which the devices operate, and the type of management that is employed. By addressing each of these challenges separately it was possible to study the problem both in detail and in a broader perspective.

Our research goal was to synthesise and describe a generalised architectural model for cognitive management of IoT devices deployed in a non-stationary environment. To guide our research and gather data, we formulated five research questions. These questions were directed toward identifying overall challenges, system and process components, adaptation mechanisms, patterns, and best practices, respectively.

We used the design science research methodology to gather the necessary data for our synthesis. This methodology is centred around answering knowledge questions and resolving design problems by analysing how concrete design artefacts interact with problems in a particular context. During the thesis, we focused on three different artefacts, namely architectures, components, and mechanisms. Each of these artefacts describes the system's behaviour on a different level in the architectural model. By studying these artefacts, we gained more knowledge and a better understanding of the design problem, which enabled us to come up with a solution to the problem, i.e., the research goal.

The data collection process followed both a bottom-up and a top-down approach. First, we did six case studies where we designed specific parts of the proposed model. These design artefacts were explored to identify which components and mechanisms are needed to perform cognitive device management in specific use cases. Then, we performed a structured literature review, where we studied architectural models found in 32 case studies from recent literature in the domain of IoT device management. The purpose of this analysis was to find general patterns that were used to solve problems related to autonomous management. We also analysed a set of generic autonomic, situation-aware and cognitive models commonly used for automation of tasks, either in the context of IoT or in other domains.

Based on the insights gathered through this thesis we then designed and synthesised a cognitive architectural model that met our research goal. The model describes adaptive behaviour on three levels. On the highest level, the architecture is based on three system components. These are the device, the device manager and the system manager. The device manager acts as a digital twin, that is, a virtual representation of the physical device. On the second level, we can find five distinct adaptation components that are contained within the system components. They are responsible for handling perception, action, adaptation process, declarative knowledge and procedural knowledge, respectively. The adaptation component is further detailed in five adaption processes, namely, monitor, analyse, learn, predict and plan. On the lowest levels, we find adaptation mechanisms and triggers, which describe the data flow between the components. Thus, the cognitive behaviour of IoT devices emerges from the interaction between the adaptation components, driven by adaptive mechanisms.

Apart from the generalised cognitive model, this thesis has made three additional contributions. First, we made a comprehensive taxonomy of adaptation mechanisms for autonomous IoT device management. The research showed that we can use reasoning, learning and planning mechanisms to drive the adaptation process. Reasoning and learning mechanisms can be further categorised into model-driven, semantic and data-driven mechanisms, which reflect the underlying principles that are used to infer situational awareness. Within those categories, these mechanisms can be implemented using linear and nonlinear programming; probabilistic analysis, Markov-modeling and Bayesian inference; fuzzy logic; dynamic programming and recursive optimisation; rule-based inference, ontologies and knowledge graphs; case-based reasoning; machine-learning; and reinforcement learning. This taxonomy can help architects and developers to decide what type of mechanism to include and where to place it in their architecture, based on their particular challenges. Second, we adapted Vernon's model of a cognitive cycle [29] into a model of cognitive planning. The model is centred around the planning component and consists of an autonomic and a situation-aware subsystem, executed through two separate control loops. This model can help architects and developers get a better understanding of the planning process when designing top-level architectures of cognitive systems. Third, we provided a list of best practices to guide the design and implementation of cognitive IoT device management platforms, coupled with recommendations for when and how to apply them. These best practices will be helpful for architects and developers who aim to design and implement cognitive systems for IoT device management.

We also made several smaller contributions through the six case studies: In Paper A, we demonstrated a quick and flexible method to prototype a platform for data management and analytics using wireless sensor nodes. In Paper B, we showed that constrained devices can outsource research-intensive machine learning to a cloud-based device manager. We also proved that it is possible to estimate the energy consumption of applications running on these devices with high precision. In Paper C, we ran a simulation that demonstrated the possibility of predicting solar energy even with limited data access, with the accuracy improving progressively as the system runs. In Paper D, we solved the problem of individual adaptation through individual and autonomous learning models in a simulation. Our results indicated that this approach could improve median prediction scores by more than 20% compared to state-of-the-art predictors for IoT energy prediction. In Paper E, we simulated a mechanism for autonomous prediction model selection as a means of mitigating the bootstrapping problem for constrained devices and helping them stay operational in periods where training data is missing. Finally, in Paper F, we simulated a mechanism that can identify suitable training data for a given node by selecting data from nodes with correlated data. Our results showed that this approach can improve the accuracy of the predictions of a new node by 14%.

7.2 Future Research Directions

The thesis is rooted in resource management for IoT devices and the design of cognitive systems. Both fields are still in an early stage, but maturing fast. This indicates that the research activity is high and that both fields are still relevant. In the following subsections, we will look at possible future research directions, in light of the work we have done and our contributions.

7.2.1 Applied Research on Development of Autonomous IoT Device Management Platforms

As explained in Chapter 1, studies on device management for IoT are characterised by sporadic and chaotic case studies, where each case study often targets specific use cases. Through this thesis, we have also observed that in open-source projects and among commercial vendors, the research and development of generic device management platforms are still mostly focused on remote-based operations. In addition, few actors offer any out-of-the-box functionality above the autonomic level. This means that many easily automated device management tasks are still either performed manually, or clients who use these platforms do a lot of custom-tailored third-party integrations for tasks that could be generalised.

Another weakness with current IoT platforms is the lack of certain features that make it difficult to deploy a modular cognitive architecture, as confirmed by the case study conducted by Veiga et al. [143]. This makes it hard to implement features that extend and improve these platforms. Thus, we encourage the research field to put even more effort into identifying general patterns within the different types of device management and implement generic solutions that can solve them. Further research is also needed on how to apply mechanisms and techniques that support and allow autonomous management, into IoT device management platforms.

7.2.2 Cognitive Resource Management for Constrained Iot Devices

Through the work of this PhD, we have seen that research is advancing slowly toward a higher level of autonomy for systems aimed at IoT device management in general and for IoT resource management in particular. However, as shown in Chapter 3, we have found little research that focuses on using *cognitive* features for the autonomous perpetual operation of constrained IoT devices. This means that there is a need for more research on IoT architectures, platforms and frameworks that use a combination of reasoning, learning and planning mechanisms to achieve a higher level of self-management and adaptation. We also see a need for more research on how to apply artificial intelligence techniques like machine learning, deep learning and reinforcement learning to the different adaptation mechanisms that are needed to solve this problem.

7.3 References

[1] Siddiqui, H., Khendek, F., and Toeroe, M. “Microservices based architectures for IoT systems - State-of-the-art review”. In: *Internet of Things* vol. 23 (2023).

[2] Nikoui, T. S., Rahmani, A. M., Balador, A., and Javadi, H. H. S. “Internet of Things architecture challenges: A systematic review”. In: *International Journal of Communication Systems* vol. 34, no. 4 (2021).

[3] Jantunen, E., Di Orio, G., Hegedus, C., Varga, P., Moldovan, I., Larrinaga, F., Becker, M., Albano, M., and Malo, P. “Maintenance 4.0 world of integrated information”. In: *Enterprise interoperability VIII*. 2019.

[4] Brock, D. M. “Autonomy of individuals and organizations: Towards a strategy research agenda”. In: *International Journal of Business and Economics* vol. 2, no. 1 (2003).

[5] Lewandowski, T., Henze, D., Sauer, M., Nickles, J., and Bruegge, B. “A software architecture to enable self-organizing, collaborative IoT resource networks”. In: *2020 Fifth international conference on fog and mobile edge computing (FMEC)*. IEEE. 2020, pp. 70–77.

[6] Sifakis, J. “System design in the era of IoT — Meeting the autonomy challenge”. In: *1st International workshop on methods and tools for rigorous system design (MeTRiD 2018)*. 2018, pp. 1–22.

[7] Foteinos, V., Kelaidonis, D., Poulios, G., Vlacheas, P., Stavroulaki, V., and Demestichas, P. “Cognitive management for the Internet of Things: A framework for enabling autonomous applications”. In: *IEEE Vehicular Technology Magazine* vol. 8, no. 4 (2013), pp. 90–99.

[8] Sheng, Z., Mahapatra, C., Zhu, C., and Leung, V. C. M. “Recent advances in industrial wireless sensor networks toward efficient management in IoT”. In: *IEEE Access* vol. 3 (2015), pp. 622–637.

[9] Imteaj, A., Thakker, U., Wang, S., Li, J., and Amini, M. H. “A survey on federated learning for resource-constrained IoT devices”. In: *IEEE Internet of Things Journal* vol. 9 (2021), pp. 1–24.

[10] Pereira, F. S., Correia, R., Pinho, P., Lopes, S. I., and Carvalho, N. B. de. “Challenges in resource-constrained IoT devices: energy and communication as critical success factors for future IoT deployment”. In: *Sensors* vol. 20 (2020).

[11] Ray, P. P. “A review on TinyML: State-of-the-art and prospects”. In: *Journal of King Saud University-Computer and Information Sciences* vol. 34 (2021), pp. 1595–1623.

[12] Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. “Machine learning in wireless sensor networks: Algorithms, strategies, and applications”. In: *IEEE Communications Surveys & Tutorials* vol. 16, no. 4 (2014), pp. 1996–2018.

[13] Imteaj, A., Mamun Ahmed, K., Thakker, U., Wang, S., Li, J., and Amini, M. H. “Federated learning for resource-constrained IoT devices: Panoramas and state-of-the-art”. In: *Federated and Transfer Learning* (2022), pp. 7–27.

[14] Ahlers, D., Kraemer, F. A., Braten, A. E., Liu, X., Anthonisen, F., Driscoll, P. A., and Krogstie, J. “Analysis and visualization of urban emission measurements in smart cities”. In: *Proceedings of the 21st international conference on extending database technology (EDBT)*. 2018.

[15] Colakovic, A. and Hadzialic, M. “Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues”. In: *Computer Networks* vol. 144 (2018), pp. 17–39.

[16] Albreem, M. A. M., Sheikh, A. M., Alsharif, M. H., Jusoh, M., and Yasin, M. N. M. “Green Internet of Things (GIoT): Applications, practices, awareness, and challenges”. In: *IEEE Access* vol. 9 (2021), pp. 38833–38858.

[17] Jamshed, M. A., Ali, K., Abbasi, Q. H., Imran, M. A., and Ur-Rehman, M. “Challenges, applications, and future of wireless sensors in Internet of Things: A review”. In: *IEEE Sensors Journal* vol. 22 (2022), pp. 5482–5494.

[18] Qiu, T., Chen, N., and Zhang, S. *Robustness Optimization for IoT Topology*. 2022.

[19] Li, G., Dong, M., Yang, L. T., Ota, K., Wu, J., and Li, J. “Preserving edge knowledge sharing among IoT services: a blockchain-based approach”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* vol. 4 (2020), pp. 653–665.

[20] Zheng, J., Cai, Y., Shen, X., and Zheng, Z. “Green energy optimization in energy harvesting wireless sensor networks”. In: *IEEE Communications Magazine* vol. 53, no. 11 (2015).

[21] Tien, J. M. “Internet of Things, real-time decision making, and artificial intelligence”. In: *Annals of Data Science* vol. 4, no. 2 (2017).

[22] Braten, A. E. and Kraemer, F. A. “Towards cognitive IoT: Autonomous prediction model selection for solar-powered nodes”. In: *2018 IEEE international congress on internet of things (ICIOT)*. 2018, pp. 118–125.

[23] Braten, A. E., Kraemer, F. A., and Palma, D. “Adaptive, correlation-based training data selection for IoT device management”. In: *Sixth international conference on Internet of Things: Systems, management and security (IOTSMS)*. 2019.

[24] Li, C., Jiang, K., and Luo, Y. “Dynamic placement of multiple controllers based on SDN and allocation of computational resources based on heuristic ant colony algorithm”. In: *Knowledge-Based Systems* vol. 241 (2022), p. 108330.

[25] Rosendo, D., Costan, A., Valduriez, P., and Antoniu, G. “Distributed intelligence on the edge-to-cloud continuum: A systematic literature review”. In: *Journal of Parallel and Distributed Computing* vol. 166 (2022), pp. 71–94.

[26] Garrido-Hidalgo, C., Olivares, T., Ramirez, F. J., and Roda-Sanchez, L. “An end-to-end Internet of Things solution for reverse supply chain management in Industry 4.0”. In: *Computers in Industry* vol. 112 (2019), p. 103127.

[27] Kiddee, P., Naidu, R., and Wong, M. H. “Electronic waste management approaches: An overview”. In: *Waste management* vol. 33, no. 5 (2013), pp. 1237–1250.

[28] Nunes, P., Santos, J., and Rocha, E. “Challenges in predictive maintenance – A review”. In: *CIRP Journal of Manufacturing Science and Technology* (2023).

[29] Vernon, D. *Artificial Cognitive Systems: A Primer*. 2014.

[30] Fahmideh, M. and Zowghi, D. “An exploration of IoT platform development”. In: *Information Systems* vol. 87 (2020), p. 101409.

[31] Sinche, S., Raposo, D., Armando, N., Rodrigues, A., Boavida, F., Pereira, V., and Silva, J. S. “A survey of IoT management protocols and frameworks”. In: *IEEE Communications Surveys & Tutorials* (2019).

[32] Pramanik, P. K. D., Pal, S., and Choudhury, P. “Beyond automation: the cognitive IoT. Artificial intelligence brings sense to the Internet of Things”. In: *Cognitive computing for big data systems over IoT: Frameworks, tools and applications*. Ed. by Sangaiah, A. K., Thangavelu, A., and Meenakshi Sundaram, V. 2018, pp. 1–37.

[33] Gill, S. S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A., Golec, M., Stankovski, V., Wu, H., Abraham, A., et al. “AI for next generation computing: Emerging trends and future directions”. In: *Internet of Things* vol. 19 (2022), p. 100514.

[34] Kakousis, K., Paspallis, N., and Papadopoulos, G. A. “A survey of software adaptation in mobile and ubiquitous computing”. In: *Enterprise Information Systems* vol. 4, no. 4 (Nov. 2010), pp. 355–389.

[35] Tahir, M., Ashraf, Q. M., and Dabbagh, M. “Towards enabling autonomic computing in IoT ecosystem”. In: *2019 IEEE intl conf on dependable, autonomic and secure computing, intl conf on pervasive intelligence and computing, intl conf on cloud and big data computing, intl conf on cyber science and technology congress (DASC/PiCom/CBDCom/CyberSciTech)*. 2019, pp. 646–651.

[36] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. “Context aware computing for the Internet of Things: A survey”. In: *IEEE Communications surveys & Tutorials* vol. 16, no. 1 (2014), pp. 414–454.

[37] Chen, M., Herrera, F., and Hwang, K. “Cognitive computing: Architecture, technologies and intelligent applications”. In: *IEEE Access* vol. 6 (2018), pp. 19774–19783.

[38] Modha, D. S., Ananthanarayanan, R., Esser, S. K., Ndirango, A., Sherbondy, A. J., and Singh, R. “Cognitive computing”. In: *Communications of the ACM* vol. 54 (2011), pp. 62–71.

[39] Funge, J., Tu, X., and Terzopoulos, D. “Cognitive modelling: Knowledge, reasoning and planning for intelligent characters”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999, pp. 29–38.

[40] Laird, J., Lebiere, C., and Rosenbloom, P. “A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics”. In: *AI Magazine* vol. 38, no. 4 (2017).

[41] Lieto, A., Bhatt, M., Oltramari, A., and Vernon, D. “The role of cognitive architectures in general artificial intelligence”. In: *Cognitive Systems Research* vol. 48 (2018), pp. 1–3.

[42] Vernon, D. “Cognitive crchitectures”. In: *Cognitive Robotics*. Ed. by Cangelosi, A. and Asada, M. 2022, pp. 1–37.

[43] Sifakis, J. “Autonomous systems – An architectural characterization”. In: *Models, languages, and tools for concurrent and distributed programming. lecture notes in computer science*. 2019.

[44] Shields, R. W. “Functional anatomy of the autonomic nervous system”. In: *Journal of Clinical Neurophysiology* vol. 10 (1993), pp. 2–13.

[45] Mølgaard, P. “Competitive effect of grass on establishment and performance of taraxacum officinale”. In: *Oikos* vol. 29, no. 2 (1977), pp. 376–382.

[46] Axelby, G. “Practical methods of determining feedback control loop performance”. In: *IFAC Proceedings Volumes* vol. 1, no. 1 (1960), pp. 78–85.

[47] Plate, H., Basile, C., and Paraboschi, S. “Policy-driven system management”. In: *Computer and information security handbook*. Ed. by Vacca, J. R. Third Edition. 2013, pp. 427–460.

[48] Rossi, D., Poggi, F., and Ciancarini, P. “Dynamic high-level requirements in self-adaptive systems”. In: *Proceedings of the 33rd annual ACM symposium on applied computing*. ACM. 2018, pp. 128–137.

[49] Corporation, I. *An architectural blueprint for autonomic computing*. Vol. 31. 2006. 2006, pp. 1–6.

[50] Arcaini, P., Riccobene, E., and Scandurra, P. “Modeling and analyzing mape-k feedback loops for self-adaptation”. In: *IEEE/ACM 10th international symposium on software engineering for adaptive and self-managing systems (SEAMS)*. 2015.

[51] Romero-Garces, A., Hidalgo-Paniagua, A., Gonzalez-Garcia, M., and Bandera, A. “On managing knowledge for MAPE-K loops in self-adaptive robotics using a graph-based runtime model”. In: *Applied Sciences* vol. 12, no. 17 (2022).

[52] Kephart, J. O. and Chess, D. M. “The vision of autonomic computing”. In: *Computer* vol. 36, no. 1 (2003), pp. 41–50.

[53] Beinhocker, E. D. “Strategy at the edge of chaos”. In: *Strategy - Critical perspectives on business and management*. Ed. by Faulkner, D. 1997, pp. 24–29.

[54] Van Bunningen, A. H., Feng, L., and Apers, P. M. G. “Context for ubiquitous data management”. In: *Proceedings of the 2005 International Workshop on Ubiquitous Data Management (UDM’05)*. IEEE. 2005.

[55] Sheth, A. “Internet of Things to smart IoT through semantic, cognitive, and perceptual computing”. In: *IEEE Intelligent Systems* vol. 31, no. 2 (2016), pp. 108–112.

[56] Sterritt, R. “Autonomic computing”. In: *Innovations in Systems and Software Engineering* vol. 1, no. 1 (2005), pp. 79–88.

[57] Aamodt, A. and Plaza, E. “Case-based reasoning: Foundational issues, methodological variations, and system approaches”. In: *AI Communications* vol. 7 (1994), pp. 39–59.

[58] Gonzalez-Briones, A., Prieto, J., Prieta, F. de la, Herrera-Viedma, E. E., and Corchado, J. M. “Energy optimization using a case-based reasoning strategy”. In: *Sensors* vol. 18 (2018).

[59] Demnitz-King, H., Gonneaud, J., Klimecki, O. M., Chocat, A., Collette, F., Dautricourt, S., Jessen, F., Krolak-Salmon, P., Lutz, A., Morse, R. M., Molinuevo, J. L., Poisnel, G., Touron, E., Wirth, M., Walker, Z., Chetelat, G., and Marchant, N. L. “Association of self-reflection with cognition and brain health in cognitively unimpaired older adults”. In: *Neurology* vol. 99 (2022).

[60] Hirt, R. and Kuhl, N. “Cognition in the era of smart service systems: inter-organizational analytics through meta and transfer learning”. In: *Thirty-ninth international conference on information systems.* Association for Information Systems (AIS). 2018.

[61] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. “Cognitive Internet of Things: A new paradigm beyond connection”. In: *IEEE Internet of Things Journal* vol. 1, no. 2 (2014), pp. 129–143.

[62] Mallory, C. D. and Boyce, M. S. “Observed and predicted effects of climate change on Arctic caribou and reindeer”. In: *Environmental Reviews* vol. 26, no. 1 (2018), pp. 13–25.

[63] Murphree, J. “Machine learning anomaly detection in large systems”. In: *2016 IEEE AutoTestCon.* 2016, pp. 1–9.

[64] Faraji-Mehmandar, M., Jabbehdari, S., and Javadi, H. H. S. “Fuzzy Q-learning approach for autonomic resource provisioning of IoT applications in fog computing environments”. In: *Journal of Ambient Intelligence and Humanized Computing* vol. 14 (2023), pp. 4237–4255.

[65] Butzin, B., Golatowski, F., and Timmermann, D. “Microservices approach for the Internet of Things”. In: (2016), pp. 1–6.

[66] Hernandez, A. B. “Autonomous management of Internet of Things technologies”. PhD thesis. Universidad Politécnica de Madrid, 2019.

[67] Giuppi, F., Niotaki, K., Collado, A., and Georgiadis, A. “Challenges in energy harvesting techniques for autonomous self-powered wireless sensors”. In: *2013 European microwave conference.* IEEE. 2013, pp. 854–857.

[68] Delicato, F. C., Pires, P. F., Batista, T., et al. *Resource Management for Internet of Things.* Vol. 16. Springer briefs in computer science. 2017.

[69] Horn, P. J. *Autonomic computing: IBM’s perspective on the state of information technology.* 2001.

[70] Milenkovic, M., Robinson, S. H., Knauerhase, R. C., Barkai, D., Garg, S., Tewari, V., Anderson, T. A., and Bowman, M. “Toward Internet distributed computing”. In: *Computer* vol. 36 (2003), pp. 38–46.

[71] Marsh, D. W., Tynan, R., O'Kane, D., and O'Hare, G. M. P. "Autonomic wireless sensor networks". In: *Engineering Applications of Artificial Intelligence* vol. 17 (2004), pp. 741–748.

[72] Kang, H., Li, X., and Moran, P. J. "Autonomic sensor networks: A new paradigm for collaborative information processing". In: *2006 2nd IEEE international symposium on dependable, autonomic and secure computing.* IEEE. 2006, pp. 258–268.

[73] Dyo, V. and Mascolo, C. "A node discovery service for partially mobile sensor networks". In: *Proceedings of the 2nd international workshop on middleware for sensor networks.* ACM. 2007.

[74] Marin-Perianu, R., Scholten, H., and Havinga, P. "Prototyping service discovery and usage in wireless sensor networks". In: *32nd IEEE conference on local computer networks (LCN 2007).* IEEE. 2007, pp. 841–850.

[75] Paradis, L. and Han, Q. "A survey of fault management in wireless sensor networks". In: *Journal of Network and Systems Management* vol. 15 (2007), pp. 171–190.

[76] Abbasi, A. A., Akkaya, K., and Younis, M. "A distributed connectivity restoration algorithm in wireless sensor and actor networks". In: *32nd IEEE conference on local computer networks (LCN 2007).* IEEE. 2007, pp. 496–503.

[77] Vieira, M. A. M., Coelho, C. J. N., Silva, D. C. da, and Mata, J. M. da. "Survey on wireless sensor network devices". In: *2003 IEEE conference on emerging technologies and factory automation.* Vol. 1. IEEE. 2003, pp. 537–544.

[78] Duan, S. and Yuan, X. "Exploring hierarchy architecture for wireless sensor networks management". In: *2006 IFIP international conference on wireless and optical communications networks.* IEEE. 2006, 6–pp.

[79] Dimitriou, T. and Krontiris, I. "Autonomic communication security in sensor networks". In: *Workshop on autonomic communication.* Springer. 2005, pp. 141–152.

[80] Krontiris, I., Dimitriou, T., Giannetsos, T., and Mpasoukos, M. "Intrusion detection of sinkhole attacks in wireless sensor networks". In: *International symposium on algorithms and experiments for sensor systems, wireless networks and distributed robotics.* Springer. 2007, pp. 150–161.

[81] Brown, S. and Sreenan, C. *Updating software in wireless sensor networks: A survey.* Tech. rep. 2006.

[82] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. "Power management in energy harvesting sensor networks". In: *ACM transactions on embedded computing systems (TECS)* vol. 6, no. 4 (2007).

[83] Shah, K. and Kumar, M. "Distributed independent reinforcement learning (DIRL) approach to resource management in wireless sensor networks". In: *2007 IEEE international conference on mobile adhoc and sensor systems.* IEEE. 2007, pp. 1–9.

[84] Tan, L. and Wang, N. “Future Internet: The Internet of Things”. In: *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*. Vol. 5. IEEE. 2010.

[85] Bandyopadhyay, D. and Sen, J. “Internet of things: Applications and challenges in technology and standardization”. In: *Wireless personal communications* vol. 58 (2011), pp. 49–69.

[86] Lopez, T. S., Brintrup, A., Isenberg, M.-A., and Mansfeld, J. “Resource management in the Internet of Things: Clustering, synchronisation and software agents”. In: *Architecting the internet of things* (2011), pp. 159–193.

[87] Hayes-Roth, B. “An architecture for adaptive intelligent systems”. In: *Artificial intelligence* vol. 72, no. 1-2 (1995), pp. 329–365.

[88] Maes, P. “Artificial life meets entertainment: lifelike autonomous agents”. In: *Communications of the ACM* vol. 38, no. 11 (1995), pp. 108–114.

[89] Atzori, L., Iera, A., and Morabito, G. “The Internet of Things: A survey”. In: *Computer networks* vol. 54, no. 15 (2010), pp. 2787–2805.

[90] Horre, W. “Management solutions for distributed software applications in multi-purpose sensor networks”. Available at https://lirias.kuleuven.be/1655716?limo=0. PhD thesis. Leuven, Netherlands: KU Leuven, Oct. 2011.

[91] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, F., Poulios, G., and Demestichas, G. “Enabling smart cities through a cognitive management framework for the Internet of Things”. In: *IEEE Communications Magazine* vol. 51, no. 6 (2013), pp. 102–111.

[92] Tsagkaris, K., Vlacheas, P. T., Athanasiou, G., Stavroulaki, V., Filin, S. A., Harada, H., Gebert, J., and Muck, M. “Autonomics in wireless network management: Advances in standards and further challenges”. In: *IEEE Network* vol. 25 (2011).

[93] Movahedi, Z., Ayari, M., Langar, R., and Pujolle, G. “A survey of autonomic network architectures and evaluation criteria”. In: *IEEE Communications Surveys & Tutorials* vol. 14 (2012), pp. 464–490.

[94] Rajan, M., Balamuralidhar, P., Chethan, K., and Swarnahpriyaah, M. “A self-reconfigurable sensor network management system for internet of things paradigm”. In: *2011 international conference on devices and communications (ICDeCom)*. IEEE. 2011, pp. 1–5.

[95] Liu, Y., Seet, B.-C., and Al-Anbuky, A. “An ontology-based context model for wireless sensor network (WSN) management in the Internet of Things”. In: *Journal of Sensor and Actuator Networks* vol. 2 (2013), pp. 653–674.

[96] Brown, S. and Sreenan, C. J. “Software updating in wireless sensor networks: A survey and lacunae”. In: *Journal of Sensor and Actuator Networks* vol. 2 (2013), pp. 717–760.

[97] Ferry, N., Ducloyer, S., Julien, N., and Jutel, D. “Power/energy estimator for designing WSN nodes with ambient energy harvesting feature”. In: *EURASIP Journal on Embedded Systems* vol. 2011 (2011), pp. 1–17.

[98] Hormann, L. B., Glatz, P. M., Steger, C., and Weiss, R. “Designing of efficient energy harvesting systems for autonomous WSNs using a tier model”. In: *2011 18th international conference on telecommunications*. IEEE. 2011, pp. 174–179.

[99] Dondi, D., Scorcioni, S., Bertacchini, A., Larcher, L., and Pavan, P. “An autonomous wireless sensor network device powered by a RF energy harvesting system”. In: *IECON 2012-38th annual conference on IEEE industrial electronics society*. IEEE. 2012, pp. 2557–2562.

[100] Kroener, M. “Energy harvesting technologies: Energy sources, generators and management for wireless autonomous applications”. In: *International multi-conference on systems, signals & devices*. IEEE. 2012, pp. 1–4.

[101] Wan, Z., Tan, Y., and Yuen, C. “Review on energy harvesting and energy management for sustainable wireless sensor networks”. In: *2011 IEEE 13th international conference on communication technology*. IEEE. 2011, pp. 362–367.

[102] Kibria, M. G. and Chong, I. “Context-awareness provisioning to support user-centric intelligence in Web of Object platform”. In: *2015 international conference on information and communication technology convergence (ICTC*. IEEE. 2015, pp. 388–392.

[103] Alam, K. M. and El Saddik, A. “C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems”. In: *IEEE Access* vol. 5 (2017), pp. 2050–2062.

[104] Boschert, S., Heinrich, C., and Rosen, R. “Next generation digital twin”. In: *Proceedings of TMCE*. Vol. 2018. 2018, pp. 7–11.

[105] Derhamy, H., Eliasson, J., Delsing, J., and Priller, P. “A survey of commercial frameworks for the Internet of Things”. In: *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)*. IEEE. 2015, pp. 1–8.

[106] Meriem, T. B., Chaparadza, R., Radier, B., Soulhi, S., Lozano-Lopez, J.-A., and Prakash, A. *GANA—Generic autonomic networking architecture*. 2016.

[107] Tsagkaris, K., Logothetis, M., Foteinos, V., Poulios, G., Michaloliakos, M., and Demestichas, P. “Customizable autonomic network management: integrating autonomic network management and software-defined networking”. In: *IEEE Vehicular Technology Magazine* vol. 10 (2015), pp. 61–68.

[108] Neves, P., Cale, R., Costa, M. R., Parada, C., Parreira, B., Calero, J. M. A., Wang, Q., Nightingale, J., Perez, E. C., Jiang, W., Schotten, H. D., Koutsopoulos, K., Gavras, A., and Barros, M. J. “The SELFNET approach for autonomic management in an NFV/SDN networking paradigm”. In: *International Journal of Distributed Sensor Networks* vol. 12, no. 2 (2016).

[109] Zhao, Z., Schiller, E., Kalogeiton, E., Braun, T., Stiller, B., Garip, M. T., Joy, J., Gerla, M., Akhtar, N., and Matta, I. “Autonomic communications in software-driven networks”. In: *IEEE Journal on Selected Areas in Communications* vol. 35 (2017), pp. 2431–2445.

[110] Ndiaye, M., Hancke, G. P., and Abu-Mahfouz, A. M. “Software defined networking for improved wireless sensor network management: A survey”. In: *Sensors* vol. 17 (2017).

[111] Afzal, A., Zaidi, S. A. R., Shakir, M. Z., Imran, M. A., Ghogho, M., Vasilakos, A. V., McLernon, D. C., and Qaraqe, K. “The cognitive Internet of Things: A unified perspective”. In: *Mobile Networks and Applications* vol. 20, no. 1 (Feb. 2015), pp. 72–85.

[112] Fortino, G., Guerrieri, A., Russo, W., and Savaglio, C. “Middlewares for smart objects and smart environments: overview and comparison”. In: *Internet of Things based on smart objects: Technology, middleware and applications.* Ed. by Fortino, G. and Trunfio, P. 2014.

[113] Venkatesh, J., Chan, C., Akyurek, A. S., and Rosing, T. S. “A context-driven IoT middleware architecture”. In: *Proceedings from TechCon.* 2015, pp. 122–127.

[114] Nascimento, N. M. do and Lucena, C. J. P. de. “FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things”. In: *Information Sciences* vol. 378 (Feb. 2017), pp. 161–176.

[115] Khazaei, H., Bannazadeh, H., and Leon-Garcia, A. “Savi-IoT: A self-managing containerized IoT platform”. In: *2017 IEEE 5th international conference on future Internet of Things and Cloud (FiCloud).* IEEE. 2017, pp. 227–234.

[116] Bacciu, D., Chessa, S., Gallicchio, C., and Micheli, A. “On the need of machine learning as a service for the Internet of Things”. In: *Proceedings of the 1st international conference on Internet of Things and machine learning.* 2017, pp. 1–8.

[117] Khan, M. A., Peters, S., Sahinel, D., Pozo-Pardo, F. D., and Dang, X.-T. “Understanding autonomic network management: A look into the past, a solution for the future”. In: *Computer Communications* vol. 122 (2018), pp. 93–117.

[118] Arzo, S. T., Bassoli, R., Granelli, F., and Fitzek, F. H. P. “Multi-agent based autonomic network management architecture”. In: *IEEE Transactions on Network and Service Management* vol. 18 (2021), pp. 3595–3618.

[119] Abbas, K., Khan, T. A., Afaq, M., and Song, W.-C. “Ensemble learning-based network data analytics for network slice orchestration and management: An intent-based networking mechanism”. In: *NOMS 2022-2022 IEEE/IFIP network operations and management symposium.* IEEE. 2022, pp. 1–5.

[120] Suh, Y.-H., Woo, S.-P., and Park, D.-H. “SLICE: Self-learnable IoT common software engine”. In: *Proceedings of the 8th international conference on the Internet of Things.* IOT ’18. Santa Barbara, California, 2018, 19:1–19:8.

[121] Dzeparoska, K., Beigi-Mohammadi, N., Tizghadam, A., and Leon-Garcia, A. “Towards a self-driving management system for the automated realization of intents”. In: *IEEE Access* vol. PP (2021), pp. 1–1.

[122] Agyemang, J. O., Yu, D., and Kponyo, J. J. “Autonomic IoT: Towards smart system components with cognitive IoT”. In: *Pan-African artificial intelligence and smart systems.* Ed. by Ngatched, T. M. N. and Woungang, I. 2022, pp. 248–265.

[123] Alfonso, I., Garcés, K., Castro, H. E., and Cabot, J. “A model-based infrastructure for the specification and runtime execution of self-adaptive IoT architectures”. In: *Computing* vol. 105 (2023), pp. 1883–1906.

[124] Chatterjee, B., Sen, S., Cao, N., and Raychowdhury, A. “Context-aware intelligence in resource-constrained IoT nodes: Opportunities and challenges”. In: *IEEE Design & Test* vol. 36 (2019), pp. 7–40.

[125] Wang, D., Zhong, D., and Souri, A. “Energy management solutions in the Internet of Things applications: Technical analysis and new research directions”. In: *Cognitive Systems Research* vol. 67 (2021), pp. 33–49.

[126] Li, J., Dai, J., Issakhov, A., Almojil, S. F., and Souri, A. “Towards decision support systems for energy management in the smart industry and Internet of Things”. In: *Computers & Industrial Engineering* vol. 161 (2021), p. 107671.

[127] Bharti, M., Kumar, R., Saxena, S., and Jindal, H. “Optimal resource selection framework for Internet-of-Things”. In: *Computers & Electrical Engineering* vol. 86 (2020), p. 106693.

[128] Liao, S., Wu, J., Mumtaz, S., Li, J., Morello, R., and Guizani, M. “Cognitive balance for fog computing resource in Internet of Things: An edge learning approach”. In: *IEEE Transactions on Mobile Computing* vol. 21 (2022), pp. 1596–1608.

[129] Kumar, M., Kishor, A., Samariya, J. K., and Zomaya, A. Y. “An autonomic workload prediction and resource allocation framework for fog-enabled industrial IoT”. In: *IEEE Internet of Things Journal* vol. 10 (2023), pp. 9513–9522.

[130] Asemani, M., Abdollahei, F., and Jabbari, F. “Understanding IoT platforms: towards a comprehensive definition and main characteristic description”. In: *2019 5th international conference on web research (ICWR).* IEEE. 2019, pp. 172–177.

[131] Mijuskovic, A., Ullah, I., Bemthuis, R. H., Meratnia, N., and Havinga, P. J. M. “Comparing apples and oranges in IoT context: A deep dive into methods for comparing IoT platforms”. In: *IEEE Internet of Things Journal* vol. 8 (2020), pp. 1797–1816.

[132] Babun, L., Denney, K., Celik, Z. B., McDaniel, P., and Uluagac, A. S. “A survey on IoT platforms: Communication, security, and privacy perspectives”. In: *Computer Networks* vol. 192 (2021), p. 108040.

[133] Microsoft. *IoT central device management guide.* URL: https://learn.microsoft.com/en-gb/azure/iot-central/core/overview-iot-central-operator#automate (visited on 12/19/2023).

[134] AWS. *What is AWS IoT?* URL: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html (visited on 12/19/2023).

[135] SAS. *IoT Analytics Solutions*. URL: https://www.sas.com/no_no/solutions/iot.html (visited on 12/19/2023).

[136] IBM. *IBM Watson IoT Platform Lite*. 2019. URL: https://www.ibm.com/docs/en/watson-iot-platform?topic=product-overview-features (visited on 12/19/2023).

[137] Particle. *IoT device management*. URL: https://www.particle.io/iot-guides-and-resources/iot-device-management-challenges-and-platforms/ (visited on 12/19/2023).

[138] Kaa. *Start building your solution with Kaa Enterprise IoT Platform*. URL: https://www.kaaiot.com/#dev-man (visited on 12/19/2023).

[139] Technologies, F. *One-IoT device management*. URL: https://friendly-tech.com/products/iot-device-management/ (visited on 12/19/2023).

[140] Wieringa, R. J. *Design science methodology for information systems and software engineering*. Nov. 2014.

[141] Hevner, A. R., March, S. T., Park, J., and Ram, S. "Design science in information systems research". In: *MIS quarterly* vol. 28, no. 1 (2004).

[142] Ditawit, K. "Smart grid demand response with mutual utility-consumer benefits". Available at https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2434714. PhD thesis. Trondheim, Norway: Norwegian University of Science and Technology, Nov. 2016.

[143] Veiga, T., Asad, H. A., Kraemer, F. A., and Bach, K. "Towards containerized, reuse-oriented AI deployment platforms for cognitive IoT applications". In: *Future Generation Computer Systems* vol. 142 (2023), pp. 4–13.

[144] Alawad, F. and Kraemer, F. A. "Value of information in wireless sensor network applications and the IoT: A review". In: *IEEE Sensors Journal* vol. 22 (2022), pp. 9228–9245.

Papers

Analysis and Visualization of Urban Emission Measurements in Smart Cities

Dirk Ahlers, Frank Alexander Kraemer, Anders Eivind Braten, Xiufeng Liu, Fredrik Anthonisen, Patrick Driscoll, John Krogstie

Abstract

Cities worldwide aim to reduce their greenhouse gas emissions and improve air quality for their citizens. Therefore, there is a need to implement smart city approaches to monitor, model, and understand local emissions to better guide these actions. We present our approach that deploys a number of low-cost sensors through a wireless Internet of Things (IoT) backbone and is thus capable of collecting high-granular data. Based on a flexible architecture, we built an ecosystem of data management and data analytics including processing, integration, analysis, and visualization as well as decision-support systems for cities to better understand their emissions. Our prototype system has so far been tested in two Scandinavian cities. We present this system and demonstrate how to collect, integrate, analyze, and visualize real-time air quality data.

A.1 Introduction

Urban emissions contribute over 60 % to global greenhouse gas emissions. Cities aim at reducing their emissions through tailored policy and integration to Smart City approaches. Smart City approaches facilitate easier integration of emission sensing into city systems and fulfill city requirements through novel and low-cost approaches [145, 146, 147, 148, 149]. The overall aim of our project1 is to fulfill the information needs of cities that need specific data for emission reduction actions by providing complementary on-the-ground emission data for improved understanding and decision making [150]. In short, the need based on future challenges faced by cities will be better and more high-granularity measurements to complement existing official measurement stations. Some Nordic cities have specific challenges in that they have already implemented a range of climate actions, which means that future impact on a certain class of emissions can only be achieved by a more detailed and granular understanding and analysis of emissions, since many broad measures are already in place. The next step then is to get better insight into more difficult to measure components, also to be able to adapt policy in fast feedback loops and at varying scales. This includes impact assessment of measures ranging from small-scale such as closing down certain streets (and being able to observe spillover and evasion effects in surrounding parts of the city) to large-scale such as changes in public transport or denser urban development. A high spatial granularity of sensor deployments is obviously not possible with the existing expensive high-quality measurement stations that are often provided nationally. Our approach, in contrast, is to use low-cost sensors to cover a city's spatial footprint with a much higher sensor density. This enables a trade-off of high number and high granularity of low-cost sensors that can compensate for their relatively lower accuracy. Existing official measurement stations are equipped with high-quality sensors that cost up to $500,000. Our low-cost approach could provide a very dense coverage of a city with 250 additional sensors for the price of one additional station by using sensor units of around $2,000 each. For ease of installation, this requires standalone sensor units that do not need cabling for electricity or connectivity. We achieve that by deploying solar-powered sensor nodes with a wireless data link over the LoRaWAN standard for Smart City IoT applications, which also enables us to quickly scale up the sensor

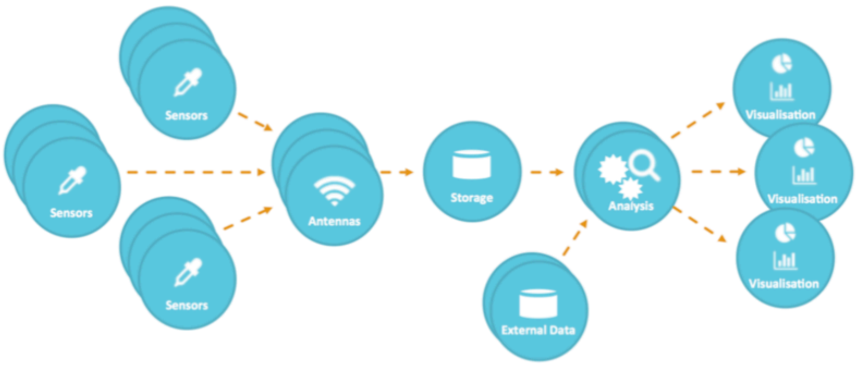

Figure A.1: Overall system architecture

deployment. The approach allows to quickly prototype system components on the hardware and software side for the overall goal of linking the measurement data to the information needs of the cities for emission reduction both for baseline and continuous data collection. After having built and deployed the general IoT sensor net-work before [150], we focus here on the integration of data sources and the data analysis infrastructure for Smart City applications.

A.2 Approach

Our approach is to build an ecosystem of relevant tools and methods to better understand city emissions and work with data, such as analytics [151, 152], visualizations [153], and decision support systems [145, 146, 154] around local emission measurements and the integration of external data sources. This is an important aspect of Smart Cities [151], and can also be used as a case study to understand and build similar systems. Our system is piloted in the two cities of Trondheim, Norway, and Vejle, Denmark. In this paper, we describe key aspects of this ecosystem of data analysis and visualization that strongly relates to challenges and requirements of the cities. We further demonstrate the integration and aggregation of data sources for a smart city.

A.2.1 Architecture

The system architecture and data flow is sketched in Fig. 1, which consists of four components: a city-wide IoT sensor network, cloud-based systems for data collection and storage, integration of external data, and analysis and visualization platforms for stakeholders. The architecture is flexible through an ecosystem approach and accommodates different components for a range of related tasks. Our technology stack follows common concepts for IoT and Smart City systems [148] with project-specific adaptations. The sensor network is composed of sensor nodes deployed within the city, which measure emissions and air parameters: CO2, NO2, PMx (particulate matter); temperature, pressure, and humidity. The data is transmitted

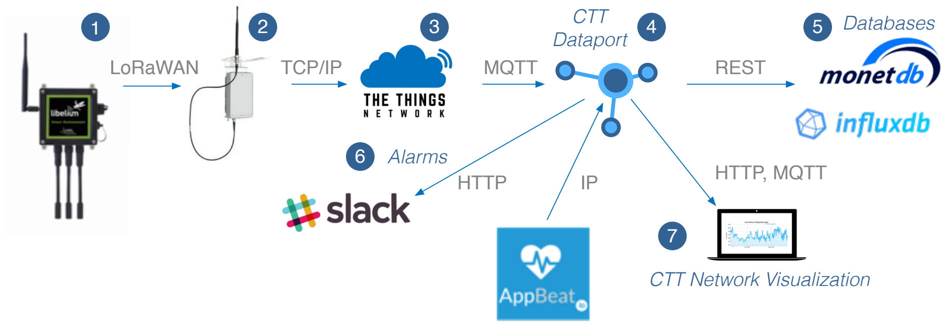

Figure A.2: Dataport protocol diagram

to the IoT backbone, which forwards collected data to the cloud storage, from where it is available for analysis and visualization, using relevant external data sources. The backbone uses LoRaWAN as a radio-based urban sensor networks through a number of gateways covering the pilot regions [150]. Data forwarding and cloud sensor management was built through the event-driven MQTT communication protocol. Visualizations and analyses are connected to all stages of the data processing. Examples are network monitoring and early data validation close to the sensors, stream processing on measurement data, up to C&C centers, satellite measurement grounding, integration into GML-based 3D city models, and other forms of mapping and integration that we describe in the following.

Table A.1: Examples of external data integration.

Type	**Example**	**Description**
Official air quality measurements	NILU data (Norwegian Air Quality Institute)	Ground truth for certain pollution types, grounding and calibrating measurements to high-quality reference stations
Remote sensing	NASA OCO-2 satellite CO2 measurements	Ground truth top-down measurements for certain emission types, large-scale coverage, low spatial resolution, coupling to large-scale modeling and validation
Traffic data	Traffic density from here.com	Estimate traffic emissions by correlating continuous external traffic density to emission measurements
	Municipal traffic counts	Validate traffic estimations, but only available for short periods
3D city models	Municipal 3D model of Vejle	Integration into existing visualization tools. Use of city geometry in future emission modeling
National statistics	GHG emission estimates from national statistics office	Down-scaled national GHG emission data, often with high uncertainties
Other municipal data and tools	GIS, statistics, decision support, etc.	Understanding emissions in the context of the city

A.2.2 Data Integration

Apart from the direct sensor data, there is a range of municipal and national data sets available as well as other external data sources that need to be included in the data analytics and visualization to support analyses and improve data quality. Table 1 gives an overview of these sources and how they can be utilized. They range from direct measurements of air quality that can be used to validate and calibrate the sensor network to other data sources that help to understand emissions in the context of a city, for example through traffic patterns [152] or integration into city tools and systems. The sensor network has the usual issues of missing data that is dealt with on a technical monitoring level and being handled by standard methods in the analyses, as well as the aggregation of data from multiple sensor units. More interesting are the challenges posed in the data integration. The sources contain highly heterogeneous data, with different timescales, measurement frequencies, spatial distributions and granularities, measurement technologies, and a complex set of related uncertainties and inaccuracies in the data.

A.2.3 Network Metadata Analysis and System Status Monitoring

The network, server components, gateways and sensors are subject to transient and permanent failures, which can ultimately result in missing data. Although the later analysis tasks can detect such losses of data, they do not analyze the cause for the error, or prevent further losses. Instead, failures in the system should be detected as quickly as possible, so that data loss is kept at a minimum. We therefore built a monitoring application (the dataport) to monitor the status of all sensors, gateways and the network [155]. It is built with the Akka framework, which facilitates the creation of fault-tolerant applications based on the actor model [156]. Actors are independent, supervised processes that encapsulate data and control logic and communicate via messages. Each device in the real world corresponds to a dedicated actor that acts as its digital twin, which is a virtual model of the sensor or gateway. It keeps track of its state in real-time, monitors all communication and triggers alarms if data is not received as expected. Incoming data contains meta-data that identifies the originating sensor and the gateway from which it was received. In this way, the digital twin for a gateway can detect if a gateway operates as expected.

Faults of a more complex nature, such as decaying sensors, erroneous behavior of sensor nodes, or missing data patterns need specific analysis. For example, a single missing measurement is expected occasionally. Based on the measurement frequency of individual sensors, it takes some cycles to determine a failure with certainty. As sensors nodes can adapt their frequency based on battery levels, a complex model of the sensor node and its status is needed for detection. Actors are organized hierarchically. On higher levels, failures can be grouped so that for example a distinction can be drawn between sensor failures versus a gateway outage that would make a set of sensors invisible. The dataport also monitors the larger system, such as the The Things Network (TTN) cloud backend and the MQTT connection. If any of the components on the data path from the sensors to the data storage fails, the dataport generates a notification. If the dataport itself fails, it is detected by an external watchdog service, in this case AppBeat. The

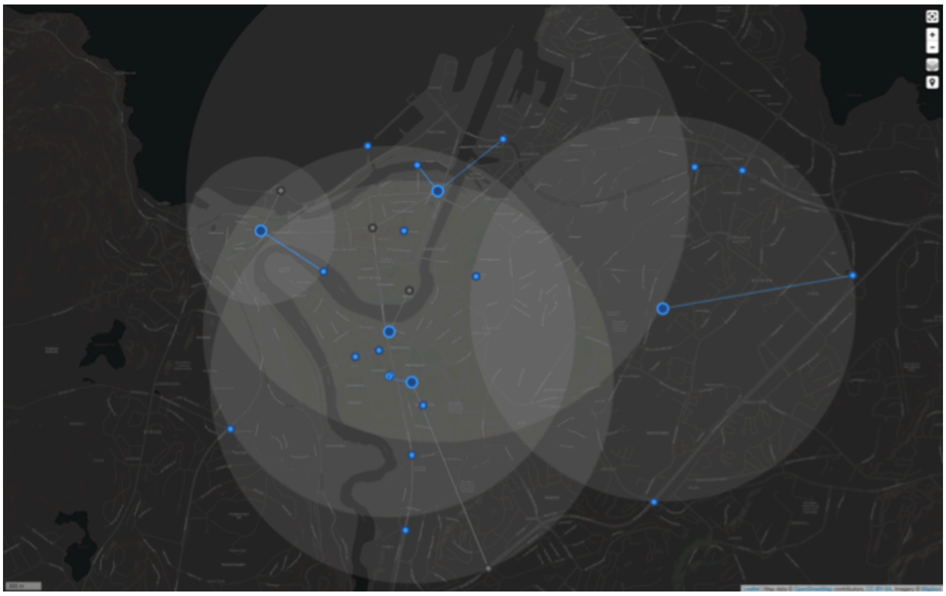

Figure A.3: Visualization of sensors, gateways and links

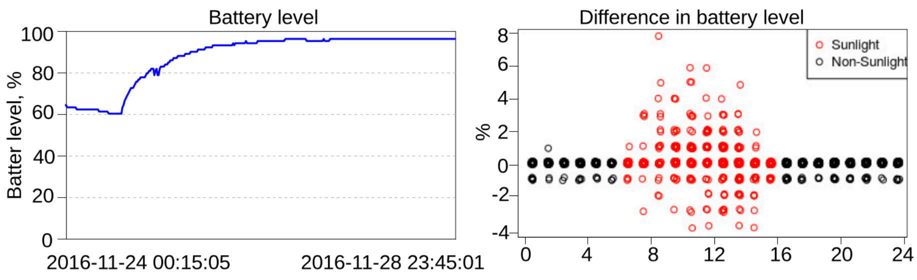

Figure A.4: Battery level analysis

dataport further drives a visualization of the network itself, shown in Fig. 3, of the structure of digital twins for sensors and gateways, their location, the connections and live data transmission between sensors and gateways. Apart from the practical value of monitoring the network, it is also a useful illustration of the spatial and measurement characteristics.

A.2.4 Data Analyses and Visualizations

A range of analyses work on the collected data streams as illustrated in Fig. 1 apart from the more operational network analysis. Examples are ongoing data collection and analysis, understanding of patterns, as well as comparison of sensor measurements to air quality measurement stations to ground the network and calibrate the sensors. There are very few official stations; to support the grounding and calibration, we have co-located one of our sensor units to the only station in

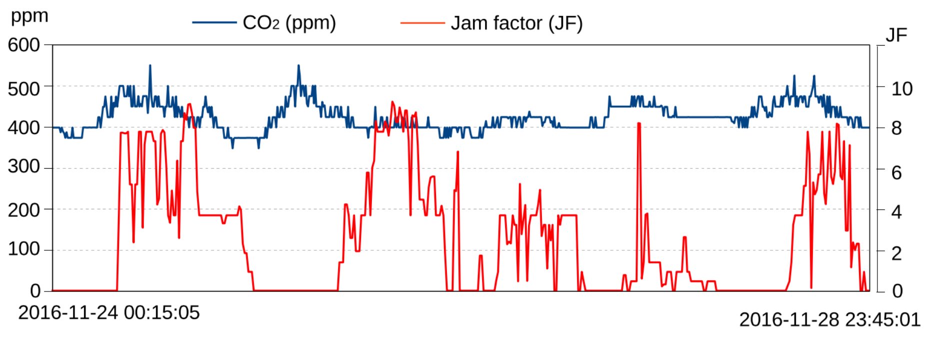

Figure A.5: A study of co2 dynamics

the pilot area. This allows to compare both absolute and relative accuracy and calibrate the local sensor and, through larger-scale correlated trends, the network, but with lower certainty. In connection with the network monitoring, it also allows the identification of outliers and malfunctioning sensors. Main ongoing work is modeling dependencies of NO2, PMx , and CO2, especially from transport emissions, which therefore also looks at linking to traffic patterns [152]. We discuss some analytics around this data in the following levels depend on the charging of the autonomous sensor units through their solar panels. Charged occurs during daytime, and is affected by weather conditions. It is important to monitor the battery level to keep the nodes running. Fig. 4 shows the battery level as a function of time (left), and the difference in battery-level from previous sent package versus time of day, and where red indicates whether the nodes could have been charged by sunlight since the previous package (right). This allows to estimate battery depletion. Dynamics of CO2 emissions and possible links to traffic in the form of a traffic jam factor (from here.com data) is shown in Fig. 5. According to the plots, we can conclude for this sensor location that traffic is not the only factor that accounts for the dynamics of the CO2 emission as they exhibit different patterns, and have no apparent correlation. In fact, CO2 emission dynamic is a more complex issue that may be affected by many factors, including traffic, wind speed, temperature, humidity and other weather conditions, as well as daily and seasonal patterns, which we will further investigate in our future work.

Visualizations and Dashboards for real-time monitoring. Fig. 6 shows the air quality and traffic flow dashboard, respectively. The dashboard is implemented using Apache Zeppelin as the visualization platform and accesses the data from the OpenTSDB time series database. The mapped sensors show the real-time data and analytic results for each location. Examples are the the air quality and traffic indicators in Fig. 6. This was further integrated into a 3D CityGML model as seen in Fig. 7 and also into a full network and data overview wall display shown in Fig. 8.

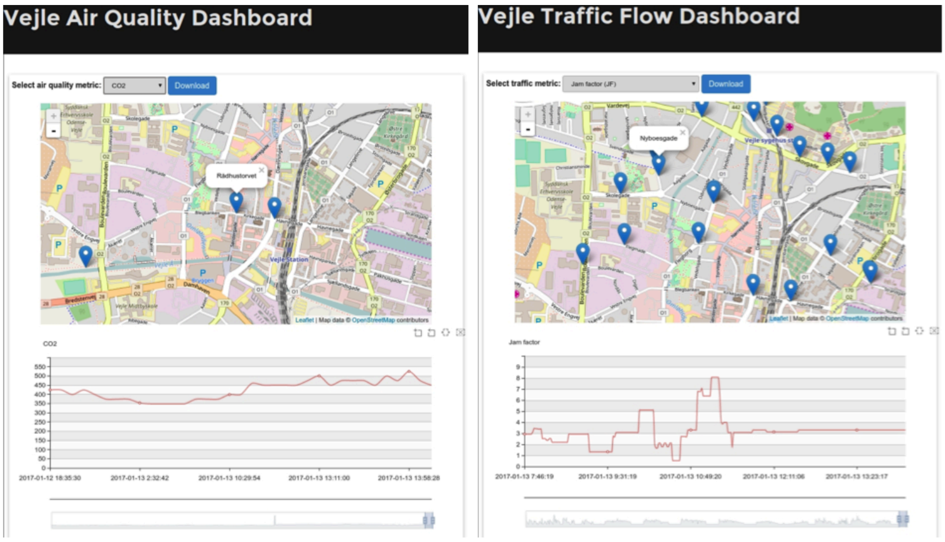

Figure A.6: Example of dashboards for air quality and traffic

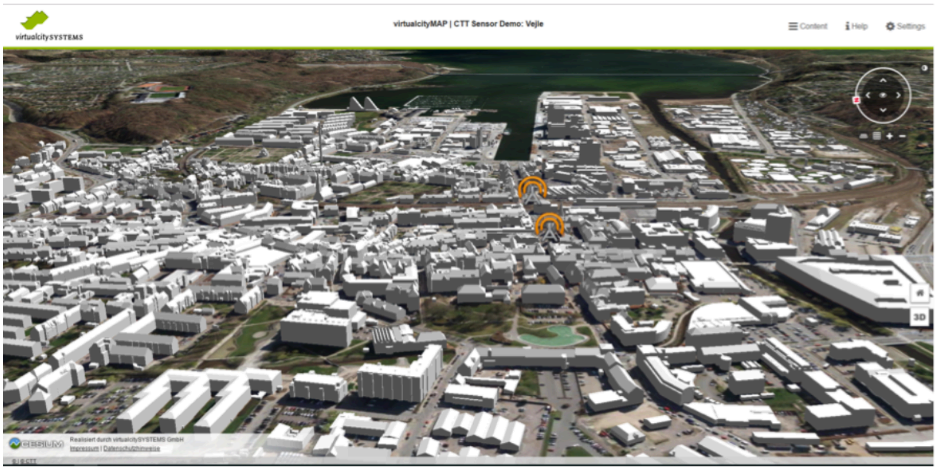

Figure A.7: Integration of sensor data into 3D city model

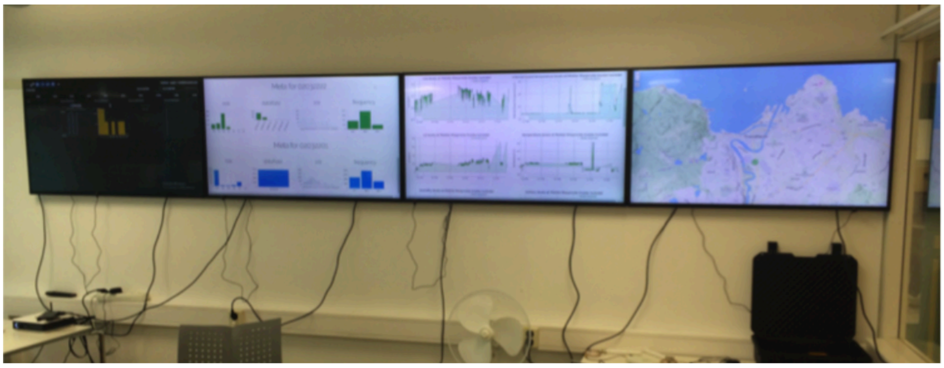

Figure A.8: Network monitoring and data visualization dashboards

A.3 Demonstration

In this first full demonstration of the CTT air quality system, we show the architecture and implementations of IoT and analytic technologies in air quality monitoring and explore insights. We use two use cases of deploying our systems in Vejle, Denmark and Trondheim, Norway, where two and twelve sensors were deployed respectively to collect air quality data. We demonstrate our system from the perspective of developers, city policymakers, and citizens. For developers, we explore the system in detail, demonstrate the building blocks of the system, and show how to build similar IoT systems; for policymakers, we aim to assist them in decision making for smart cities with the proposed IoT technologies, e.g., urban planning; for citizens, we aim to raise the awareness of environment protection and greenhouse gas reduction for better city life. We use real-time data collected from the deployed sensors from both cities, as well as traffic data sets streamed from the third- party traffic flow monitoring operator, here.com. The sensor data consist of the CO , NO , and PM , and weather data including 22x humidity and temperature. The sensor data is collected at a five-minute interval. The demo also uses historic data saved in our time-series database, collected since January 2017. Developers' point of view: We show the architecture and components used by our air quality monitoring system, including sensors, IoT sensor network, cloud storage for sensor data and external traffic data, analytics, and dashboards. We demonstrate how to collect, process and visualize high-frequent sensor data in our system developed on the Zeppelin platform; and how to streamline the whole data flow, including segmentation, chaining, and automation. Finally, we demonstrate how to generate dashboards and integrate analysis algorithms in the web interface. Attendees can vary system and analysis properties, and observe the reflection on the dashboard; and change the dependency of the data flow to evaluate the flexibility of the data stream analysis. City officials' point of view: We show an interactive dashboard to analyze CO2 dynamics using real-time and historic measurement data, and demonstrate the pattern and its correlation to the traffic flow (see Fig. 5–6). In addition, we demonstrate the 3D CityGML model integrating different measuring points of air quality (see Fig. 7). In this demo scenario, we can inject synthetic data showing different pollution levels. We interact with attendees by discussing urban planning issues such as construction sites of roads, buildings or factories, and see how different pollution levels will affect their decision makings. Also, we consult with attendees about choosing the sites of air quality monitoring, e.g., according to the road network and building density. Citizens' point of view: We demonstrate air quality and traffic flow on the dashboard using the real-time data. Similarly, we use synthetic data with different pollution levels, and discuss the influence on routing planning, and citizens' approaches for emission reduction. Attendees can browse historic data in the system to investigate anomalous emission levels.

A.4 Conclusion

We have described the possibilities for urban emission monitoring and our approach and the prototype system we have developed together with the approaches to data flows and analysis. The flexible and scalable solution allows to quickly prototype different analysis approaches on top of the sensor streams to link measured data to cities' information needs for emission reduction. In future work, we plan to improve the measurement network and the real-time and aggregate dashboards. Further, with more data collected, we will be able to tune models for emission distribution and dispersion to overcome some of the issues and provide improved analysis with better models. Integration into decision support systems is a far goal. Urban emission monitoring needs a range of heterogeneous data and we are continuing to build useful urban systems around it.

A.5 Acknowledgments

This paper is part of the Carbon Track & Trace (CTT) project. Funding for this project was in part provided by the LoCaL2 Climate-KIC flagship programme. We thank all our project partners and specifically VCS3 for the 3D city model demo. Parts of this article are based on project deliverables [154].

References

[145] Ingelrest, F., Barrenetxea, G., Schaefer, G., Vetterli, M., Couach, O., and Parlange, M. B. "SensorScope: Application-specific sensor network for environmental monitoring". In: *ACM Trans. Sens. Networks* vol. 6 (2010), 17:1–17:32.

[146] Kumar, P., Morawska, L., Martani, C., Biskos, G., Neophytou, M. K.-A., Sabatino, S. D., Bell, M. C., Norford, L. K., and Britter, R. "The rise of low-cost sensing for managing air pollution in cities." In: *Environment international* vol. 75 (2015), pp. 199–205.

[147] Martinez, K., Hart, J. K., and Ong, R. "Environmental sensor networks". In: *Computer* vol. 37 (2004), pp. 50–56.

[148] Zanella, A., Bui, N., Castellani, A. P., Vangelista, L., and Zorzi, M. "Internet of Things for Smart Cities". In: vol. 1 (2014), pp. 22–32.

[149] Zheng, Y., Liu, F., and Hsieh, H.-P. “U-Air: when urban air quality inference meets big data”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013).

[150] Ahlers, D., Driscoll, P. A., Kraemer, F. A., Anthonisen, F. V., and Krogstie, J. “A Measurement-Driven Approach to Understand Urban Greenhouse Gas Emissions in Nordic Cities”. In: *NIK Norsk Informatikkonferanse* (Nov. 2016), pp. 1–12.

[151] Schleicher, J. M., Vogler, M., Inzinger, C., and Dustdar, S. “Towards the Internet of Cities: A Research Roadmap for Next-Generation Smart Cities”. In: *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics* (2015).

[152] Zheng, Z., Wang, D., Pei, J., Yuan, Y., Fan, C., and Xiao, L. F. “Urban Traffic Prediction through the Second Use of Inexpensive Big Data from Buildings”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016).

[153] Liono, J., Salim, F. D., and Subastian, I. F. “Visualization Oriented Spatiotemporal Urban Data Management and Retrieval”. In: *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics* (2015).

[154] Ahlers, D. and Driscoll, P. *Technical Architecture. CTT2.0 Deliverable. Technical Report D2.4.* Tech. rep. Norwegian University of Science and Technology, 2016.

[155] Grønsleth, H. H. “Urban Greenhouse Gas Level Monitoring Using LoRaWAN”. MA thesis. Norwegian University of Science and Technology, 2016.

[156] Hewitt, C. E., Bishop, P. B., and Steiger, R. “A Universal Modular ACTOR Formalism for Artificial Intelligence”. In: *International Joint Conference on Artificial Intelligence.* 1973.

Towards Cognitive Device Management: A Testbed to Explore Autonomy for Constrained IoT Devices

Anders Eivind Braten, Nattachart Tamkittikhun, Frank Alexander Kraemer, Doreid Ammar

Abstract

Providing constrained IoT devices with more intelligence is important to make them work optimally with regard to energy consumption and quality of data. To overcome the constraints of the sensors, we place cognition, i.e., learning and planning, in the cloud. In this demonstration paper, we present a testbed for exploring autonomy for constrained sensor nodes.

B.1 Motivation: Autonomous constrained nodes

Miorandi et al. [157] emphazise the importance of including self-management and autonomic capabilities in devices connected to the Internet of Things (IoT). Often, the purpose of such devices is to collect as much data as possible, with the best possible quality. Since data collection requires energy, which is a scarce resource in most IoT systems, many devices harvest energy from their environment. However, they run the risk of depleting their battery if they lack knowledge of their energy budget, i.e., energy intake vs. energy consumption. This is a challenge since IoT devices often are too constrained in regard to both computation and communication to learn and plan ahead. Constrained devices usually also lack access to contextual data, like weather forecasts. Such data can contribute to better estimations. Outsourcing the estimation process to the cloud is a countermeasure to these constraints. We have built a lab to explore how to make constrained sensor nodes able to operate more autonomously. The lab has two goals: 1) to explore autonomous resource management for IoT devices using machine learning; and 2) to make the learning process itself autonomous.

B.2 System overview and architecture

In our testbed, we use Libelium Waspmotes to collect data. They are connected to The Things Network (TTN) via LoRaWAN antennas. We have three distinguished sets of nodes, deployed in different ways. Some are placed in various locations in the city of Trondheim. These are used for long-term data collection. We also have equipment to do high-precision measurements of energy consumption in the lab. Our most prominent sensor nodes are the 8 solar powered sensor nodes we have deployed on the roof, as shown in Figure B.9. These are used for experimentation and testing under real weather conditions. The sensor nodes provide sensor data such as CO_2 level, sound level and temperature, as well as energy-related meta data, e.g., battery level and solar charging current. The data is uploaded to TTN via an antenna located approximately 400 meters away. An overview of the architecture of the testbed is shown in Figure B.10. We also collect the latest local weather forecast from the Norwegian Meteorological Institute. The collected data is stored as .csv files and then pre-processed in the cloud using Python and Pandas. Our ultimate goal is to build a system (using Scikit-Learn) that continuously learns, predicts and plans the energy budget for each device. To meet such requirements, we train a machine learning model every day. The model uses the most recent data (i.e., weather forecast and the position of the sun) as input to predict the energy intake per sensor node for the next 24 hours. Later, we use the predictions to identify

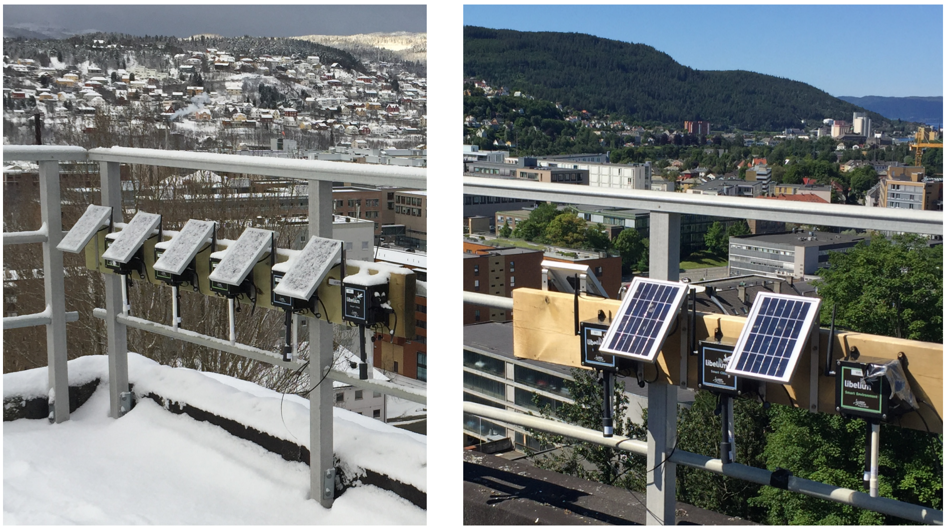
Figure B.9: Sensor nodes used for testing in real weather conditions.

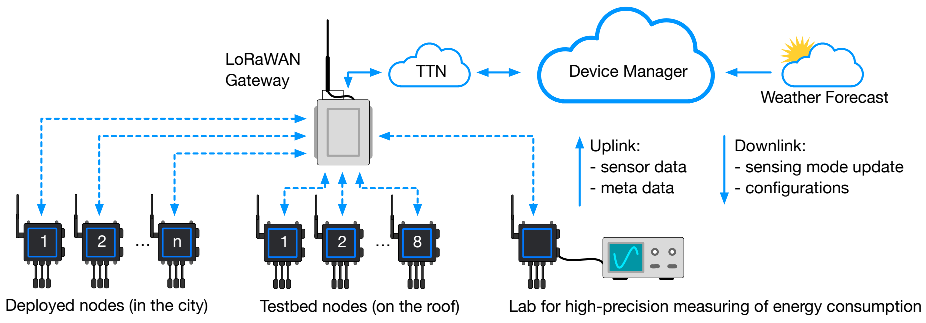

Figure B.10: Deployment of the autonomous sensor testbed lab.

the optimal sensing mode for each sensor device, i.e., instruct the sensor to send as much data as it can while keeping the battery level as stable as possible. Through the TTN downlink, we instruct the sensor node to apply the identified sensing mode. The downlink is also used to update general configuration parameters, when necessary.

B.3 Energy and Sensing Cycle

The sensor node can adapt its own behavior by adjusting activities in different phases in its sensing cycle. We define the phases as sensing, data processing, transmitting data, and going to sleep for a period. The energy consumption of each phase can be adjusted by varying the execution time. This way, we can instruct the nodes to use different sensing modes to adjust their energy consumption behaviors.

For instance, one sensing mode performs frequent sensing and calculates a mean to reduce the risk of errors, while another mode acquires the data sparingly to conserve energy. However, this comes with the tradeoff of lower data quality.

B.4 Findings

By means of our testbed, we have made these discoveries:

- The energy consumption of applications running on a constrained device can be estimated with high precision using a generalized model of different sensing modes [158].
- Constrained devices can outsource research-intensive machine learning to a device manager located in the cloud. This is possible even when the communication channels themselves are constrained [159].
- To predict energy intake properly, it is important that the electronics of the sensor nodes reveal more data regarding the system operations, such as the current that is produced by a solar panel and the state of the charger [159].

To assist a system that also does the learning autonomously, we additionally need the following:

- Device management will have to manage, among other things, the process of knowledge gathering explicitly. This means that it also needs to take knowledge and learning into account, hence evolving into *cognitive device management*.
- To collect better training data, sensor nodes have to accept commands that override their normal operation. For instance, to gather more knowledge, they need to be able to switch on or off the charging, measuring short circuit currents, and switch to a different sensing mode on command. This way, the central device manager can choose to prioritize learning over normal operation in a period after a device has been deployed.

B.5 Demonstration setup

In the demonstration, we will: a) present an overview of our lab; b) give a real-time demonstration of how we are using the Waspmotes for noise detection; c) show interactive charts that visualize the collected data and the predictions that are made using machine learning; d) play time-lapse videos that shows the sensor nodes under real weather conditions.

B.6 Acknowledgements

We thank David Palma and Pål Sæther for their help in setting up the lab, and Dirk Ahlers and Fredrik Anthonisen for their support with the Waspmotes through the Carbon Track and Trace project.

References

[157] Miorandi, D., Sicari, S., Pellegrini, F. D., and Chlamtac, I. “Internet of things: Vision, applications and research challenges”. In: *Ad Hoc Networks* vol. 10 (2012), pp. 1497–1516.

[158] Tamkittikhun, N., Hussain, A., and Kraemer, F. A. “Energy Consumption Estimation for Energy-Aware, Adaptive Sensing Applications”. In: *International Conference on Mobile, Secure, and Programmable Networking*. 2017.

[159] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N., and Palma, D. “Solar energy prediction for constrained IoT nodes based on public weather forecasts”. In: *Proceedings of the Seventh International Conference on the Internet of Things*. New York, New York, USA, 2017, pp. 1–8.

Solar Energy Prediction for Constrained IoT Nodes Based on Public Weather Forecasts

Frank Alexander Kraemer, Doreid Ammar, Anders Eivind Braten, Nattachart Tamkttikhun, David Palma

C

Abstract

Solar power is important for many scenarios of the Internet of Things (IoT). Resource-constrained devices depend on limited energy budgets to operate without degrading performance. Predicting solar energy is necessary for an efficient management and utilization of resources. While machine learning is already used to predict solar power for larger power plants, we examine how different machine learning methods can be used in a constrained sensor setting, based on easily available public weather data. The conducted evaluation resorts to commercial IoT hardware, demonstrating the feasibility of the proposed solution in a real deployment. Our results show that predicting solar energy is possible even with limited access to data, progressively improving as the system runs.

C.1 Introduction

Energy harvesting provides a sustainable source of power for Internet of Things (IoT) nodes and also simplifies their deployment. The downside is that their performance often changes considerably over time. In case of solar energy, variations occur throughout the seasons and depend on climatic parameters like weather, temperature, solar irradiance, hourly solar angle, season and geographical location [160, 161]. Tilt angle and orientation of the solar panel [162], and other effects like shadows [163, 164], also influence the output that can be produced by a stationary photovoltaic (PV) panel. Rechargeable batteries dampen fluctuations through energy buffering, but only to some degree. For instance, within a project involving the deployment of solar-driven nodes to measure greenhouse gas emissions [150], sensor devices worked properly during summer, but failed during winter when sun exposure was lacking.

Data quality and energy consumption are closely related [165]. Instead of abruptly running out of power, nodes should adapt their operation [166]. Sensor devices can for instance average over several measurements to reduce the number of transmitted messages and save power [167]. On the other hand, to increase accuracy and adapt to sudden and unexpected changes, sensors may increase their sampling rate at the cost of increased energy consumption [168]. Additionally, systems consisting of several nodes may also balance the load between them, improving energy consumption while providing a good overall sensing coverage [169].

Node failure due to lack of energy is one problem; another is to not use enough of the available energy to gain better data. When a battery is fully charged, all solar energy that exceeds the consumption of a node is wasted [82]. Instead, the sensor node could have consumed more energy and thereby acquired more data or data with higher accuracy. For that, sensors need to be aware of their current and future energy budget and plan ahead so that they can operate optimally. Due to the scale of IoT, such optimizations must happen autonomously. And since sensors are placed into heterogeneous and changing environments, optimizations must happen continuously and for each sensor individually. One method of choice is machine learning to predict the energy budget of a sensor node. An important component in such a prediction is the solar energy, which depends on features such as the position of the sun relative to the solar panel, the atmosphere and other effects.

Forecasting the available solar power with different time horizons is already part of the operation of solar power plants, to predict fluctuations in energy production and quickly compensate for sudden changes. These deployments are usually large in scale and can look back on a long history of available data of dedicated sensor information, for instance by observing the clouds in the sky by cameras, and even allow for manual supervision [170]. In an IoT setting, such a high degree of instrumentation and supervision is unrealistic. Sensor nodes are placed at many different locations, and requiring specific orientation towards the sun would complicate the deployment process. Adding more instrumentation to nodes (like irradiation measurement or sky cameras observing the cloud coverage) would make them more expensive and complex. Instead, systems should be based on off-the-shelf sensor nodes and require only minimal instrumentation and setup.

In this paper, we present the results of an experiment on a prototype designed to investigate how and to which degree machine learning algorithms can be used to predict the solar energy budget for sensor nodes, based on easily available input data. In detail, our contributions are the following:

- An explanation of effects in embedded systems that need to be taken into account with a set of data preparation steps to gain good training data.
- A study of which features are most effective as input for solar energy prediction.
- A study of different machine learning algorithms and how they score as predictors.

One of the most crucial factors for success in machine learning is the availability of data. Due to the heterogeneity of environments into which sensor devices are deployed, this may be a problem. We therefore also study how the approach develops over time, starting with no initial data.

The focus of this paper is the prediction of solar energy. Nevertheless, we also outline the planning algorithm that selects the proper sensor operation mode, since it gives context to the solar energy prediction. Our approach also takes the constraints typical in IoT into consideration: Since machine learning is executed on a server as part of the device management, sensor nodes only require minimal computational effort. As communication we use the constrained LoRaWAN protocol.

C.2 Related Work

Machine learning is already widely used in big scale forecasting for large solar-power farms. Weather data from numerous sources are blended with several models and methods via post processing, in order to produce the most accurate solar-energy forecast possible [170]. This requires high computational power and access to large amounts of data. However, these systems use highly specialized models tuned to the specific location of the solar farm. For more distributed energy resources, where small scale photovoltaic panels are connected to a smart grid, making an accurate prediction of the expected output is also important. These are used to forecast and plan the total distribution of the produced energy over the entire energy grid [171, 161]. In contrast to the solar farms, this needs to be done with more generalized models and less fine-grained weather data. The power produced by a PV-panel highly depends on the irradiation reaching it. Shi et. al [172] propose a forecasting model to predict the output of PV-panels based on a classification of the weather. They show that weather conditions, clouds, solar angle, and season are factors that must be taken into consideration when predicting the energy budget for a solar-energy harvesting device.

Kansal et al. [173] show how power management is inherently different for a node using energy-harvesting compared to one powered by battery only. This is due to the variability of available energy and because conventional energy optimization methods are not always optimal in an energy-harvesting scenario. As such, it is important to adapt the workload to the amount of energy that can be harvested. They demonstrate how a closed-loop electronic circuit can be used to predict and plan the energy budget to achieve energy-neutrality. However, they do not consider the usage of weather forecasts in their energy planning algorithm, since they assume that the expected energy production is typically the same on a given time for consecutive days. This, however, does not apply to many parts of the world, where the weather condition can shift from day to day, and even from hour to hour. Hsu et al. [174] propose a modified power management method that is taking unstable and uncontrollable conditions into consideration, using reinforcement learning. They claim their method gives a performance increase of 2.3 % in summer time.

Constrained devices that scavenge for energy in a location where weather conditions are shifting, need to be able to predict their energy budget to keep a steady battery level. Szydlo et al. [175] propose a concept of a two-stage predictive power-adaptation method that uses weather forecast services to plan how much energy it is possible to harvest from the wind in the near future. Their aim is to create a power management system that address the problem of optimal control of battery neutrality under shifting weather conditions, and that at the same time can guarantee a satisfactory level of functionality. Based on these plans, they propose to change the energy consumption of the devices by switching between four operation modes, which use different amounts of energy. This is a similar approach to ours. However, their results are based on values gathered from a simplified test unit and then run in simulations, while we are using an off-the-shelf sensor station and have built a prototype for running the test and measure the actual values.

C.3 System Overview

Figure C.11 shows an overview of the system. It consists of sensor nodes that communicate via gateways with a backend server.

C.3.1 Sensor Nodes

We deployed eight Waspmotes [176] from Libelium, shown in Figure C.12. They are based on the 8-bit Atmega1281 microcontroller. Each sensor node is coupled with sensors to measure CO_2, temperature, pressure and humidity. Some sensors also measure particle matters (PM) and NO_x. However, since, we focus on the energy

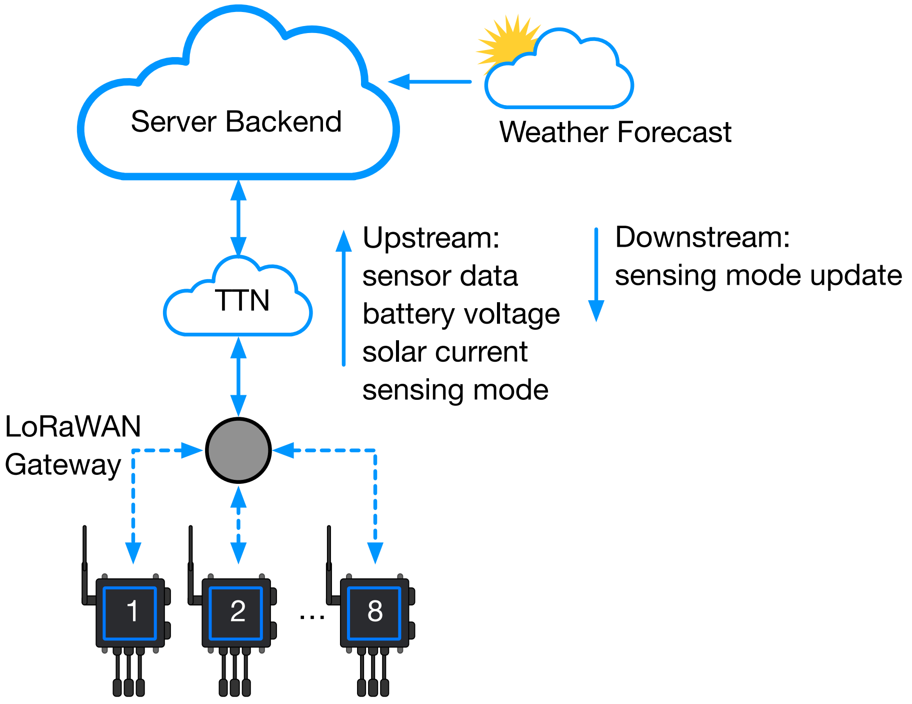

Figure C.11: Overview of the system

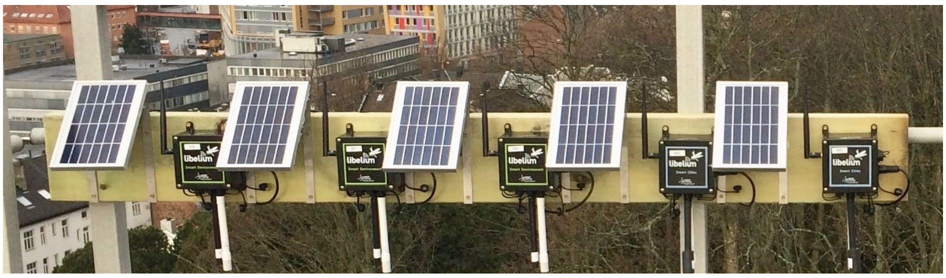

Figure C.12: Testbed with five of the eight sensor nodes

management, only the energy consumption of the sensors is relevant in the following. The CO_2 sensors, for instance, include a heater to correctly capture the gas density at a specific temperature. Therefore, they require a considerable amount of energy when sensing.

Each sensor node is powered by a lithium-ion polymer (LiPo) battery with the capacity of 6600 mAh and a maximum voltage of 4.2 V. The battery is connected via a charging controller to a solar panel, which can provide a current of up to 330 mA. For this experiment, all solar panels face the same direction. The controller protects the battery from overcharging, which has some implications for our data as explained later.

The sensor nodes periodically execute sensing cycles, i.e., they wake up, make some measurements, send the results to the LoRaWAN gateway and then go into sleep mode again. To adjust their behavior, sensor nodes can be configured using discrete *sensing modes* [158]. Each sensing mode assigns specific values to the length of the sleep cycle and the number of measurements within each sensing cycle. The sensing modes are designed so that a lower sensing mode yields less frequent measurements and less samples per sensing cycle, and accordingly uses less energy. Higher sensing modes provide more and better data and use more energy.

C.3.2 Network

For the wireless connection, we use LoRaWAN (868 MHz and 433 MHz ISM band) [177]. The corresponding gateway is deployed 500 meters away and connected to The Things Network (TTN, [178]). Before put into deep sleep, sensor nodes transmit the necessary data required by the backend server to TTN's gateways via the LoRaWAN uplink channel. This is the actual sensor data together with the energy-related data i.e., the battery voltage, the incoming solar current and the sensing mode. From TTN, our server fetches the data of all sensor nodes at regular intervals, as explained later. LoRaWAN also provides a downlink channel from the server towards the sensor nodes. We use this channel to update the sensing mode in the sensor nodes as a result of the energy planning. The fair access policy of LoRaWAN and TTN limits the uplink transmission to a 30 seconds airtime per day per node, which corresponds to roughly 647 bytes per day, given the spreading factor of 7. The downlink is even more restricted, with 10 messages per day per node.

C.3.3 Server Backend

The server backend collects all data and has the task to determine the optimal sensing mode for each sensor node. Figure C.13 provides an overview. The server operations are coarsely structured into learning, predicting and planning.

The server repeatedly collects the data sent via LoRaWAN to The Things Network (TTN) from their servers **(1)**. This raw byte data is decoded so that the individual data fields can be stored as files in CSV format. The server also collects weather forecast data **(2)**. Both the weather and the sensor device data are combined into training and testing data for the various machine learning methods **(3)**, explained later in detail.

For the prediction part, there are three machine learning modules. The first one **(5)** predicts the solar energy output based on the weather data and time, and is the main focus of this paper. The other two modules are used to predict the expected battery level based on the incoming solar current **(6)**, and to predict the energy consumption for a sensor node given its sensing mode **(7)**.

The prediction modules contain different machine learning algorithms, detailed later. They are trained by corresponding modules. This means, for instance, if **(5)** is a neural network, **(4)** encapsulates the execution of the backpropagation algorithm to train it.

The planning module **(8)** uses the prediction modules to simulate several potential energy budgets for a day, given different sensing modes. These potential budgets are then evaluated, which leads to the selection of the best sensing mode

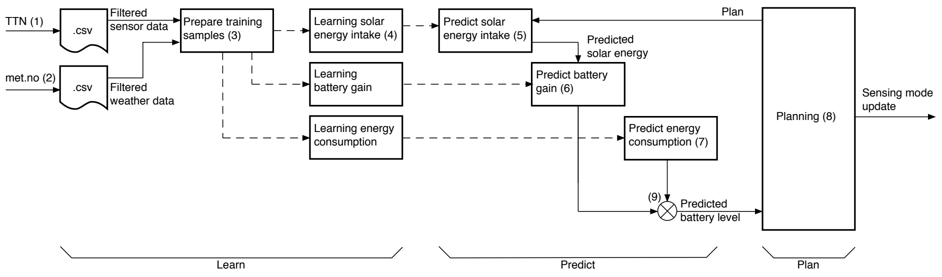

Figure C.13: Overview of the operations in the server backend

for each sensor node, which is then sent back to the them as a sensing mode update, via LoRaWAN.

C.4 Data Acquisition

Table C.2 lists the data required for learning.

Table C.2: Overview of input data

Energy data	battery voltage, solar charge current, sensing mode
Weather data	forecast time, production time, location, temperature, wind speed, wind direction, pressure, humidity, cloudiness (high, medium, low, total), fog, dewpoint temperature, precipitation, weather symbol
Sun position	zenith, azimuth

C.4.1 Energy Data from Sensor Nodes

The firmware of the Waspmotes provides access to the voltage at the battery and the current arriving from the solar panel. The latter is a good indicator for the available solar energy, and will be the value to be predicted by our algorithms in the next section. There is one caveat with the current from a solar panel: It only flows if the Waspmote also consumes energy. We will later see which consequences this has for the data preparation. However, measuring the current from the solar panel is still the best indication for the available solar energy. In contrast, only measuring the open-circuit voltage at the solar panel is less useful. Solar panels are usually non-linear in terms of sunlight intensity versus output open-circuit voltage [179]. This means that the open-circuit voltage is an indication if the sun is shining, but not very precise regarding the available energy.

C.4.2 Weather Forecast and Sun Position

The weather forecast data is collected from the Norwegian Meteorological Institute via their publicly accessible API [180]. Forecasts are published at irregular intervals, about three times a day. The different data fields are listed in Table C.2. Each weather forecast has a production timestamp, i.e., the time when it was created. The weather forecast includes predictions for several points in time in the future, each labelled by the forecast timestamp. For each forecast timestamp, the report lists several numerical weather values. Cloudiness is provided at three different levels (low, medium, high) as well as an aggregated value (total). The weather symbol is a discrete value between 1 and 50 that encodes a weather scenario. For instance, sun is encoded as 1, rain as 10 and snow as 13.

The position of the sun is expressed in terms of two angles, azimuth and zenith. These can be calculated on the server based on the location of the deployment and the time, i.e., this data does not need to be collected.

C.5 Data Preparation

The effectiveness of the machine learning algorithms depends to large degrees on the preparation of the input data. This preparation requires knowledge about the domain of the system, i.e., the electrical properties of batteries, solar panels, the charging controller and some knowledge about the sun and weather forecasts.

C.5.1 Preparing Battery Level Data

For Li-ion/LiPo batteries, the battery level (also called *state-of-charge*, SoC), and the voltage at the battery have a relationship that can be approximated by a linear model as shown in Figure C.14. It is therefore possible to estimate the battery level as percentage from 0 to 100 based on the measured voltage at the battery. For our experiment, we use the approximation provided by Libelium as part of their firmware [181]. The conversion from battery voltage to battery level in percent is only an issue of understanding the data, since it is more convenient to look at a percentage ($0 \dots 100$ %) than at a voltage ($\approx 3.3 \dots 4.2$ V) when talking about the state of a battery. For the actual prediction this conversion has little impact as long as it is done consistently. In our case, we derive the battery level from two approximated linear functions of the battery voltage illustrated by the two connected solid lines, as opposed to the unknown real model shown as dotted line in Figure C.14, which represents the general curve of Li-ion/LiPo batteries [182].

C.5.2 Preparing Current Data from Solar Energy

Figure C.15 shows the solar current throughout three days. April 7th is a day with extremely varying weather, which makes the solar current vary over the day. April 9th is a consistently cloudy day, and April 12th is a very sunny day. The curve of the sunny day shows a steep slope because of the sunrise at around 6:00. However, at 8:45 the solar current drops to zero. This is due to the charging controller. When the battery level is rising above 98 % or 99 % (depending on the node), the charging controller switches off charging and the Waspmote is powered from the battery only,

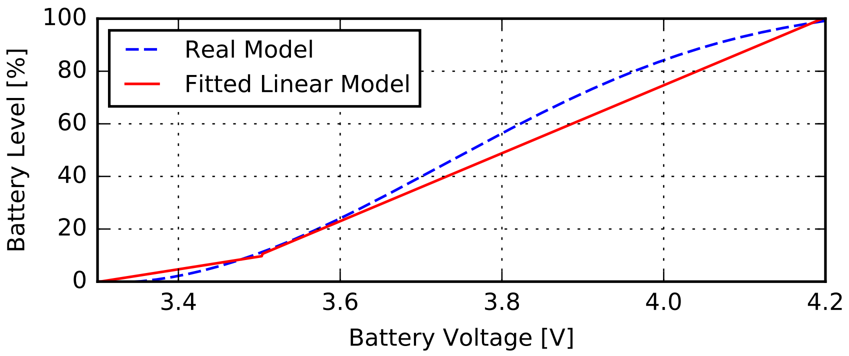

Figure C.14: The relationship between the voltage and battery level (state-of-charge, SOC)

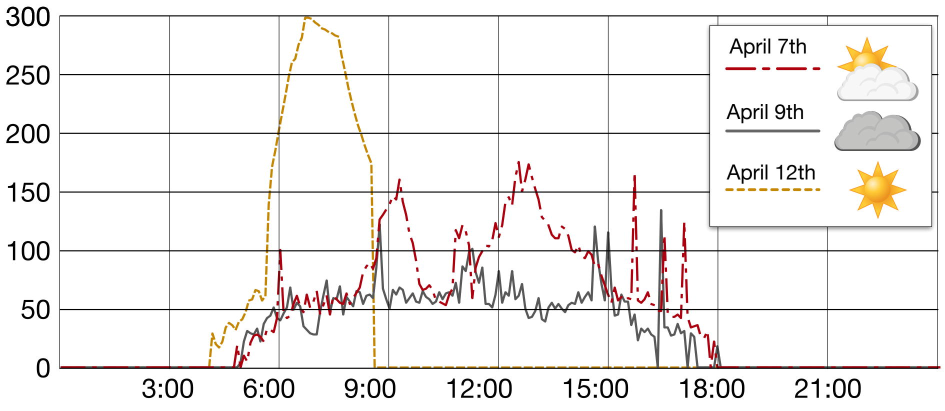

Figure C.15: The solar current on three days with very different weather conditions.

even if solar energy is available. The charger switches back into the charging state once the battery voltage falls below ≈ 4.07 V, which corresponds to ≈ 83.7 % of battery level. This explains why there are periods in which the solar current is zero despite the shining sun.

The machine learning algorithms need to know about this charging behavior; otherwise, they would be confused by sunny conditions that seemingly do not lead to the expected solar current. Adding data about the charging state is part of the data preparation of the solar current. Since Waspmotes do not provide any data regarding the state of the charger, we have to guess it from the charging current and the battery level. For this, we look for the point when charging stops, and observe the battery level at the same point for all days of a node. Then we select the highest battery level as the full battery threshold for that node. The thresholds for all the nodes are used in the data filtering program to assign battery full state to data points. After every data point is marked with states, those with the state *battery full* are removed.

C.5.3 Aggregation of Weather Forecast

Each forecast is provided as a table, where each row describes the expected weather at specific points in time. For the close future, this interval is one hour. For predictions further in the future, these intervals are increasing to up to six hours. Each row has therefore two timestamps: the time for which the weather is predicted (forecast time), and the time when the forecast was produced (production time). Since the forecasts cover overlapping times, the data preparation selects for each time the forecast where forecast time and production time have the shortest distance. This means that the most up-to-date forecast is selected.

C.6 Energy Budget Planning

The planning algorithm is only sketched in the following since it is not the focus of this article, and rather provides context for the machine learning and prediction of the solar power current. This algorithm simulates the development of the battery level, given that the node executes a specific sensing mode. As input it uses the current battery level, the sensing mode and the weather prediction. The planning algorithm simulates the time ahead in intervals, for which we selected 30 minutes. It first uses the machine learning module for the solar power prediction (see Figure C.13 **(5)**) to predict the expected solar current, detailed in the next section.

From the predicted solar current, the planning algorithm needs to estimate how the battery will develop within the simulated 30 minute interval. This depends on two components: (1) how much energy the sensor node consumes in the given sensing mode, and (2) the energy added from the solar panel. These two components can be estimated by two different modules, and then added together, as done in Figure C.13 **(9)**).

- To analyze the energy consumption of a given sensing mode (Figure C.13 **(6)**), we select periods during the night in which the solar energy is zero. During these periods, the battery level decreases. We analyze the slope, averaging over several nights, which gives a good indication of the energy consumption of the currently executed sensing mode.

- To come from the solar intake to the increase in battery level, we need to determine a simple factor that calculates the increase in battery level from the average solar current for a given time interval. We currently estimate this factor manually by comparing simulated data with real ones, but we foresee also here automated statistical methods.

The plans for the different sensing modes are compared with each other, based on a utility function that penalizes an empty battery or wasting solar energy. The planning algorithm then selects the sensing mode that leads to a plan with the best utility, and updates the sensing mode of a sensor node accordingly, using the downstream LoRaWAN channel.

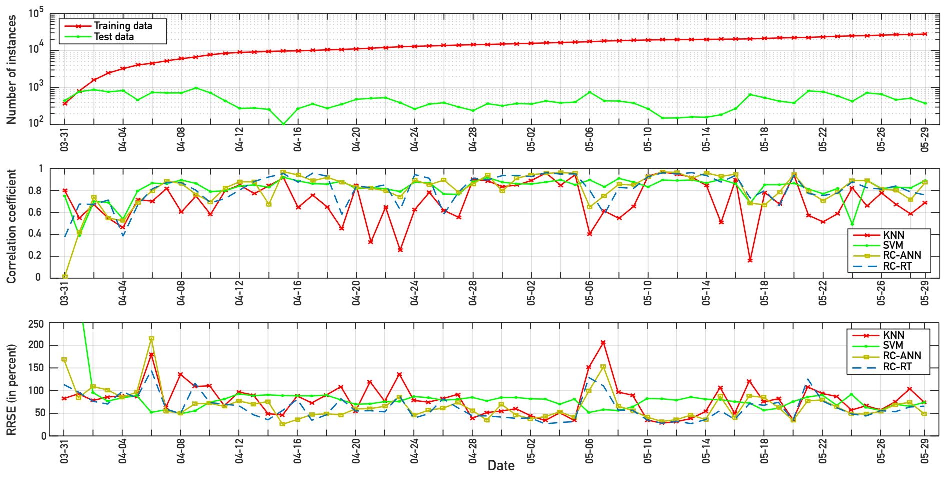

Figure C.16: Top: Number of instances of training and testing data. Middle and bottom: Correlation coefficient and RRSE (root relative squared error) for the models built for each day.

C.7 Solar Power Prediction

The biggest unknown influence on the energy budget is the availability of solar energy, which varies considerably between days based on the weather. The following section shows the prediction accuracy of the solar current of different machine learning methods, taking the weather forecast and solar angles as input features.

C.7.1 Training and Test Data

In our experiment, we are also interested in the bootstrapping problem, that means, how the accuracy of the prediction will develop over time when we start with no data at all. This corresponds to a realistic scenario of a fresh deployment of sensors that has not yet observed any data at all. We simulate therefore how the training data as well as the algorithms will develop over time. For each day d_n with $n \in \{0, \ldots, 60\}$ for which we have collected data, we use all data from the previous days $d_0, \ldots, d_{n-1}$ as training data to create a new model R_n. Data of day d_n serves as test data. In our context, it is important to take data from entire days as test data, and remove those entire days from the training data, to avoid data leakage. Simply taking out 20 % of random values over a longer period is not good enough. The relatively slowly changing conditions would lead that the training data to contain very similar samples as there are in the test data. Instead, our method of using an entire day as test data and only using data from previous days as training data corresponds to a causal way that gives realistic results.

Figure C.16 shows at its top the number of data samples in the training and test sets (on a logarithmic scale). The lower curve shows the samples from each day, which are used as test data for that day. Its minimum is on the 15th of April, where there was very little training data because the batteries of all nodes were fully charged. The upper curve shows the training data used for building the model on that day. Since the training data for day d_n is the collective test data of all days $d_0, \ldots, d_{n-1}$ before it, the number of training samples monotonously increases.

C.7.2 Attribute Selection

To improve the accuracy of solar power prediction, we used the *Ranker* method in [183] for attribute selection to rank attributes and to perform attribute selection by the removal of redundant and irrelevant attributes. As a result, the selected list of features, sorted by ascending attribute rank-order, is the following: zenith, azimuth, low clouds, high clouds, temperature forecast, medium clouds, symbol.

C.7.3 Machine Learning Algorithms

In our work, we have selected a subset of widely-used ML algorithms, namely *k-nearest-neighbor* (k-NN), *Support Vector Machines* (SVM), *Artificial Neural Networks* (ANN), *Random Tree* (RT) decision tree learner and *Random Committee* (RC), which are briefly described hereafter. For more details, please refer to [183].

- k-nearest-neighbor (k-NN) is an instance-based lazy learner that compares the value to predict with existing values gathered from the k closest training instances (neighbors) using a distance metric. This algorithm requires all the training instances to be kept in memory and does not produce any model.
- Support Vector Machines (SVM) is a technique that builds cutting hyperplanes, which separate the data in an optimal manner.
- Artificial Neural Networks (ANNs) consist of an interconnected network of nodes (artificial neurons). Each node maps complex relationships between inputs and outputs using weights learned iteratively over the training data by the backpropagation learning algorithm.
- Decision Trees are prediction models in the form of a tree graph that are built by binary splitting the set of data using a recursive greedy search algorithm. We selected the *RandomTree* decision tree algorithm that considers a given number of random features at each node.
- Random Committee is a meta technique that can be applied on other algorithms in order to turn them into more powerful learners. It builds an ensemble of base algorithms and averages their predictions in a way to avoid overfitting and reduce the variance of the algorithm output. This technique makes sense if the base algorithm is randomized. In our work, *Random Committee* works only for ANN and *Random Tree* since the base of these algorithms is randomized, which is not the case for k-NN and SVM.

C.7.4 Building Machine Learning Models

For each of the tested ML algorithms, we built models using the entire training set by means of the Waikato Environment for Knowledge Analysis (Weka) [183]. In this

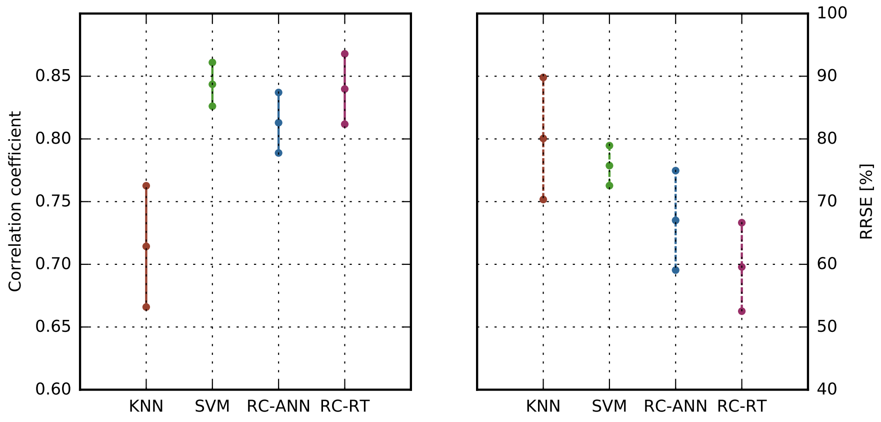

Figure C.17: Average correlation coefficients and RRSE for the models

work, we have conducted several tests to select a good tuning for each algorithm. However, tuning these models further is beyond the scope of the paper.

C.8 Evaluation and Discussion

The middle of Figure C.16 shows the correlation coefficient for the various models on each day. This coefficient measures the statistical correlation between the actual values of solar power and the predicted values. It ranges from -1 for perfect negative correlation, through 0 when there is no correlation, to 1 when the results are perfectly correlated. The bottom of Figure C.16 shows the root relative squared error (RRSE), which is one of the error evaluation measures described in [183]. Note that the best numerical prediction model is still the best no matter which error measure is used. Figure C.17 shows the average correlation coefficients and the average RRSE for all models, with a 95 % confidence interval.

The accuracy of the prediction depends on several factors:

- The accuracy of the weather forecast, this means to which degree the weather forecast corresponds to the actual weather at the sensor node.
- New weather situations that have not been observed before.
- Missing data, which can happen either due to node failures or transmissions problems.
- Long periods with fully charged batteries in many nodes at the same time. In these cases, the system does not learn anything about the solar energy available.

The RRSE of most models spike on the 6th and 7th of May On these days, the pattern of reported solar energy was significantly different from the two weeks before. Revisiting Figure C.15, we see that the most common pattern was similar to that on

April 12th with a spike in charging and a fully charged battery. However, in some other days where we also registered an increased error, the intake is distributed over time and without any identifiable pattern. This could be due to a number of factors (i.e., weather forecast and battery level) and we believe with more data such variations will be less frequent.

Concerning the lack of learning data when batteries are fully charged, we can see this effect in the period following May 25th, when most batteries were full due to very good weather. This resulted in few samples for the solar intake, that were in addition taken in short periods between fully charged batteries, which is why they may not accurately represent the actual solar energy.

C.8.1 Quality of the Predictors

An overall analysis of the results for each predictor reveals their different properties. k-NN is not a good predictor. It is unstable, with the lowest correlation coefficient and highest RRSE. The other predictors are better. SVM is quite stable and its average RRSE is 76 % with a low standard deviation. RC-RT has the lowest RRSE and a high value of the correlation coefficient. In summary, SVM, RC-ANN and RC-RT are good predictors. They may even improve with further tuning of these algorithms.

C.8.2 Suitability of the Prediction

Since the learning process starts at the second day of the systems operation with only the data from day one, the prediction accuracy is understandably low during the first days. However, we see that it is improving and, for the methods other than k-NN, remains relatively stable. Based on our 60 days of observations, we argue that the predictions are already useful for a planning algorithm. The predictions rely on the accuracy of the weather forecast, which itself is prone to errors. In addition, the weather at the test site in Trondheim, Norway, is very volatile. Of course, one remedy would be to maintain a dedicated local weather station. However, with that we would make the deployment of IoT nodes more complicated; our intention is to only rely on easily available public weather forecast data.

C.9 Conclusion

We have presented an approach to predict the solar energy input for constrained IoT nodes based on numerical weather forecasts that are typically easily available. This allows for effective energy-budget planning, which is much needed for resource-constrained nodes. We have also observed that the choice of machine learning method matters. k-NN shows the biggest drop in accuracy on some days, while the other algorithms stay more stable.

Given that the predictions are based on weather forecast data, which itself contains uncertainties, a planning algorithm needs to take the accuracy of the prediction into account. It can for instance analyze confidence intervals, study best and worst cases and eventually select the most adequate strategy. In addition, we have observed that the behavior of the charging controllers used in off-the-shelf

IoT nodes leads to less training data, since they cannot measure the available solar energy once the battery is full. This should have influence on the design of solar-driven embedded IoT nodes; they should be able to gain some insights on the available solar energy even when fully charged.

Acknowledgment

David Palma is funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie Grant Agreement No. 699924, SINet, Software-defined Intermittent Networking project.

References

[82] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. "Power management in energy harvesting sensor networks". In: *ACM transactions on embedded computing systems (TECS)* vol. 6, no. 4 (2007).

[150] Ahlers, D., Driscoll, P. A., Kraemer, F. A., Anthonisen, F. V., and Krogstie, J. "A Measurement-Driven Approach to Understand Urban Greenhouse Gas Emissions in Nordic Cities". In: *NIK Norsk Informatikkonferanse* (Nov. 2016), pp. 1–12.

[158] Tamkittikhun, N., Hussain, A., and Kraemer, F. A. "Energy Consumption Estimation for Energy-Aware, Adaptive Sensing Applications". In: *International Conference on Mobile, Secure, and Programmable Networking.* 2017.

[160] Yadav, A. K. and Chandel, S. "Solar radiation prediction using Artificial Neural Network techniques: A review". In: *Renewable and Sustainable Energy Reviews* vol. 33 (2014), pp. 772–781.

[161] Yang, H.-T., Huang, C.-M., Huang, Y.-C., and Pai, Y.-S. "A weather-based hybrid method for 1-day ahead hourly forecasting of PV power output". In: *IEEE transactions on sustainable energy* vol. 5, no. 3 (2014), pp. 917–926.

[162] Hartner, M., Ortner, A., Hiesl, A., and Haas, R. "East to west–The optimal tilt angle and orientation of photovoltaic panels from an electricity system perspective". In: *Applied Energy* vol. 160 (2015), pp. 94–107.

[163] Lefevre, B., Peeters, S., Poortmans, J., and Driesen, J. "Predetermined static configurations of a partially shaded photovoltaic module". In: *Progress in Photovoltaics: Research and Applications* vol. 25, no. 2 (2017), pp. 149–160.

[164] Shaiek, Y., Smida, M. B., Sakly, A., and Mimouni, M. F. "Comparison between conventional methods and GA approach for maximum power point tracking of shaded solar PV generators". In: *Solar energy* vol. 90 (2013), pp. 107–122.

[165] Lawson, V. and Ramaswamy, L. "Data Quality and Energy Management Tradeoffs in Sensor Service Clouds". In: *Big Data (BigData Congress), 2015 IEEE International Congress on.* IEEE. 2015, pp. 749–752.

[166] Jayakumar, H., Lee, K., Lee, W. S., Raha, A., Kim, Y., and Raghunathan, V. “Powering the internet of things”. In: *Proceedings of the 2014 international symposium on Low power electronics and design*. New York, New York, USA, 2014, pp. 375–380.

[167] Moser, C., Thiele, L., Brunelli, D., and Benini, L. “Adaptive power management for environmentally powered systems”. In: *IEEE Transactions on Computers* vol. 59, no. 4 (2010), pp. 478–491.

[168] Dias, G. M., Nurchis, M., and Bellalta, B. “Adapting Sampling Interval of Sensor Networks Using On-Line Reinforcement Learning”. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)* (2016), pp. 460–465. ieeexplore: 7845391.

[169] Yoo, H., Shim, M., and Kim, D. “Dynamic duty-cycle scheduling schemes for energy-harvesting wireless sensor networks”. In: *IEEE communications letters* vol. 16, no. 2 (2012), pp. 202–204.

[170] Haupt, S. E. and Kosovic, B. “Big Data and Machine Learning for Applied Weather Forecasts: Forecasting Solar Power for Utility Operations”. In: *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE. 2015, pp. 496–501.

[171] Tushar, W., Chai, B., Yuen, C., Smith, D. B., Wood, K. L., Yang, Z., and Poor, H. V. “Three-party energy management with distributed energy resources in smart grid”. In: *IEEE Transactions on Industrial Electronics* vol. 62, no. 4 (2015), pp. 2487–2498.

[172] Shi, J., Lee, W.-J., Liu, Y., Yang, Y., and Wang, P. “Forecasting power output of photovoltaic systems based on weather classification and support vector machines”. In: *IEEE Transactions on Industry Applications* vol. 48, no. 3 (2012), pp. 1064–1069.

[173] Kansal, A., Hsu, J., Srivastava, M., and Raghunathan, V. “Harvesting aware power management for sensor networks”. In: *Proceedings of the 43rd annual Design Automation Conference*. 2006, pp. 651–656.

[174] Hsu, R. C., Liu, C.-T., Wang, K.-C., and Lee, W.-M. “QoS-aware power management for energy harvesting wireless sensor network utilizing reinforcement learning”. In: *Computational Science and Engineering, 2009. CSE’09. International Conference on*. Vol. 2. IEEE. 2009, pp. 537–542.

[175] Szydlo, T. and Brzoza-Woch, R. “Predictive power consumption adaptation for future generation embedded devices powered by energy harvesting sources”. In: *Microprocessors and Microsystems* vol. 39, no. 4 (June 2015), pp. 250–258.

[176] *Waspmote Datasheet*. v7.0. Libelium. Oct. 2016.

[177] Sornin, N., Luis, M., Eirich, T., Kramp, T., and Hersent, O. *LoRaWAN Specification*. 1st ed. LoRa Alliance. Jan. 2015.

[178] *The Things Network*. https://www.thethingsnetwork.org. Accessed: 2017-06-06. 2017.

[179] Ashry, M. and Fares, S. "Electrical characteristic measurement of the fabricated CdSe/p-Si heterojunction solar cell under radiation effect". In: *Microelectronics and Solid State Electronics* vol. 1, no. 2 (2012), pp. 41–46.

[180] *Meteorologisk institutt.* https://api.met.no. Accessed: 2017-06-06. 2017.

[181] Gascon, D., Bielsa, A., Cuartielles, D., and Carmona, Y. *Waspmote API v026 - WaspPWR.cpp.* http://www.libelium.com/api/waspmotev26/da/d2b/WaspPWR_8cpp_source.html. [Source Code]. Accessed: 2017-06-06. Dec. 2016.

[182] Gandolfo, D., Brandao, A., Patino, D., and Molina, M. "Dynamic model of lithium polymer battery–Load resistor method for electric parameters identification". In: *Journal of the Energy Institute* vol. 88, no. 4 (2015), pp. 470–479.

[183] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. *Data Mining: Practical machine learning tools and techniques.* 2011.

Operationalizing Solar Energy Predictions for Sustainable, Autonomous IoT Device Management

Frank Alexander Kraemer, David Palma, Anders Eivind Braten, Doreid Ammar

Abstract

For sustainable IoT systems, solar-power prediction is an essential element to optimize performance, allowing devices to schedule energy-intensive tasks in periods with excess energy. In regions with volatile weather, this requires taking the weather forecast into account. The problem is how to provide such solar energy predictions with high accuracy for large-scale IoT systems with various devices in an autonomous way, without manual adaptation effort. We present a detailed study on machine-learning approaches for the prediction of solar power intake for large scale IoT systems. We examine which machine learning models, feature sets and sampling rates gain the best results for a medium-term forecasting horizon. We also explore an operational setting in which devices are deployed without prior data and machine learning models are re-trained for each sensor continuously as a form of online learning. Our results show that prediction errors can be reduced by 20 % compared to the state of the art, despite strong weather volatility.

D.1 Introduction

Energy harvesting via solar panels allows wireless devices to replenish their energy buffers and is thus one element towards a sustainable, maintenance-free Internet of Things (IoT) with perpetual operation, as it removes or reduces the need to switch batteries [184]. Use cases where solar power has great potential cover a wide range of domains, like smart cities [150], [185], harbors [186] and agriculture [187]. Rainforest Connection [188], for example, creates acoustic monitoring systems to detect illegal deforestation, using recycled phones powered by solar panels. Operating sustainably, with minimal or no maintenance, is crucial for the feasibility and economic aspects of such use cases. The better the predictions, the more strategically, and hence optimally, IoT devices can act: Apart from adjusting their sensing intervals, they can schedule energy-intensive tasks in periods of energy surplus. Such tasks can include software updates, transmission of aggregated sensing data, or re-training of machine learning models. Improved energy management helps to minimize the required energy buffer and solar panel size of IoT devices, making them simpler, easier to deploy and less obtrusive. This makes systems cheaper, or possible at all, and further facilitates approaches that even integrate solar energy supplies directly onto chips [189].

To plan energy budgets effectively also under volatile weather conditions, Sharma et al. [190] and Renner et al. [191] argue that IoT devices require access to solar energy predictions that also take the weather forecast into account. With the availability of new machine learning methods and computational power in general, this leads to the question of how these methods can improve the prediction of solar energy intake. While machine learning has been applied for that purpose in the domain of renewable power, less attention has been paid to solar prediction for IoT nodes, which require medium-term predictions (up to several days ahead) for energy budget planning. There is also the challenge of heterogeneity in large-scale deployments, where devices operate in different settings, for example regarding their position towards the sun or local obstacles such as trees or buildings. To avoid manually modeling these differences, which is prohibitive for large IoT installations, prediction models should hence be individual, that is, per IoT device and work autonomously. We envision that IoT devices are supplied with solar energy predictions as part of the device management, offered by edge or cloud services. For such an approach to work efficiently at scale, it is required that the operation of the prediction models is feasible. Instead of manual adjustment and tuning, predictions must refer directly to individual sensors' expected energy, based on the previous energy intake and on features that are easy to acquire, like a public weather forecast.

In this work, we explore the use of various machine learning techniques combined with public weather forecasts for the prediction of solar energy. This is the first work that discusses the issue of machine learning for medium-term solar energy prediction for IoT devices, paying the necessary attention to operational aspects. We have previously examined how different machine learning methods can be used in a constrained sensor setting [159]. We now go further and (i) present the performance of various machine learning techniques, (ii) conduct an ablation study to identify the most useful features, (iii) introduce scaled forecast metrics that allow us to compare prediction performance independent of seasonal changes, (iv) study the influence of sampling frequency (i.e., how often solar energy should be sampled as training data), and (v) investigate how the accuracy of the predictions develops after a deployment in an operational setting. Our results show that the machine learning models based on weather forecasts outperform other methods by more than 20 %.

We start with an introduction to solar harvesting in IoT in Sect. D.2 that also provides the system context, and a study of related work around energy planning and solar energy prediction in Sect. D.3. This is followed by the discussion of our method in Sect. D.4, which highlights the techniques for model and feature selection and introduces the metrics for our evaluation. We then systematically analyze the type of machine learning models, the suitability of features for the predictions, and the sensitivity to the sampling frequency in Sect. D.5. In Sect. D.6, where we combine all insights, we explore and discuss the performance of the prediction models in an operational setting, and compare them to the state of the art.

D.2 Solar Energy Harvesting Prediction in IoT

We first provide an overview of the significance of solar energy prediction in IoT and explain then how energy predictions can be integrated into device management that constitutes our system context.

D.2.1 The Need for Solar Energy Predictions

The challenge with energy harvesting is its stochasticity, and that energy is not always available when needed by an application [192]. Some of the stochasticity is compensated by energy buffers like batteries or super capacitors. But this only helps to a certain degree, as the required capacity of the buffers must be limited to reduce device cost and physical dimensions [82]. Therefore, energy planning is required [193, 194, 195], aiming at aligning the application energy demand closer

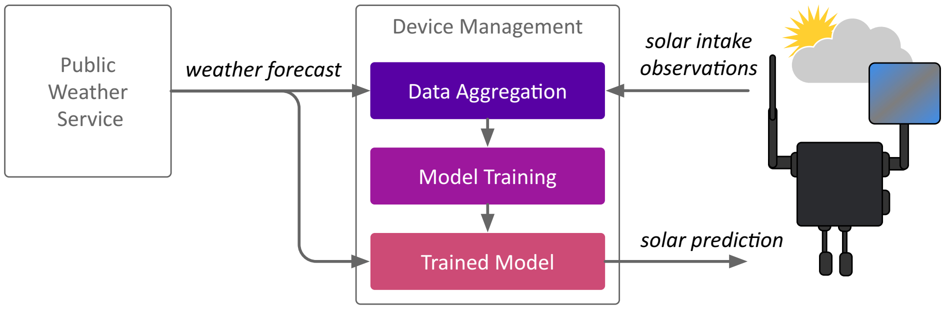

Figure D.18: Operational setting, including IoT device and device management platform.

with the availability of harvestable energy. This can be achieved for instance by adjusting the duty cycle of the application, allocating tasks to nodes with better energy budgets [193], or allowing a tradeoff between sensing accuracy and energy consumption [196]. Some tasks, such as training of machine learning models, are also tolerant to delays and can hence be scheduled in time slots where more energy is available or when the demand from other tasks is lower.

A basis for effective energy budget planning is the availability of highly accurate predictions for the incoming energy [185, 82, 193, 194, 195, 197]. Available solar energy often follows quasi-cyclic diurnal patterns [192], which motivates approaches that estimate the incoming solar energy based on historic data, which we will review in Sect. D.3. While such approaches may be suitable for short time horizons (up to 3 hours) or long-term horizons (beyond several days), they are not sufficient for medium-term horizons (3 hours to 3 days), as Sharma et al. [190] and Renner [191] concluded. This is because the arriving energy is not only dependent on the position of the sun relative to the solar panel but also the coverage of the sky with clouds at various levels, which can vary considerably with the local weather conditions. Sharma et al. [190] and Renner [191] therefore highlighted the importance of also taking the weather forecast into account, and report significant improvements in accuracy compared with approaches that only rely on historic data.

Another aspect of sustainable and cost-efficient IoT solutions is operational: due to the system scale, devices must operate autonomously. Prediction models must not require manual fitting or oversight for the individual devices. We therefore turn our attention to off-the-shelf machine learning methods and want to explore how they can improve the prediction accuracy of solar prediction when taking the weather forecast into account. We focus in our study on a medium-term time horizon, as this time horizon is significant for energy budget planning in devices with typical energy buffer sizes, and approaches based solely on historic data do not perform well for this horizon. The novelty of our work is the thorough exploration of machine learning options, selection of features and sampling frequencies to achieve better prediction performances than the current approaches as the basis for IoT device energy budget management.

D.2.2 System Context and IoT Device Management

We suggest to include the weather forecasts into prediction models as part of the device management [23]. This allows the training of machine learning models for solar energy prediction and the actual prediction to be executed off-device, in cloud or edge hardware. Figure D.18 provides an overview of the system, explained in the following. To allow the device management to be specific to the individual IoT devices' settings and micro-environments, the device management distinguishes different device instances as follows:

- IoT devices record their individual solar intake observations and send them to the device management module. In Sect. D.5.3, we will discuss the significance of the reporting frequency.
- The data aggregation step combines the solar intake observation with the weather forecast data of a region covering a device, further explained in Sect. D.5.1.
- These data is the input for training prediction models, which is the main focus of our work. Which input features to use is discussed in Sect. D.5.2, and the amount of training data to store is discussed in Sect. D.6.1. In principle, models can be retrained with the arrival of every new observation, but for most use cases a daily training is sufficient, as shown in our final evaluation in Sect. D.6.2. We note that the computational effort for retraining a model is manageable compared with the typical tasks of device management, further discussed in Sect. D.6.3.
- The trained model is then used to provide solar energy predictions to the device, taking weather predictions as input. The computational effort for this prediction is negligible. Depending on the planning mode of the device, predictions can be provided every hour, or for instance at midnight for the entire next day.

In this article, we focus on the performance of the prediction models and which features and frequency of data they require as input.

D.3 Related Work

There is a wide range of approaches for solar power prediction, which vary in terms of input data, forecast horizon and temporal resolution. Wireless sensor nodes are typically constrained in computation, which motivates approaches that use the exponentially weighted moving average (EWMA), like Kansal et al. [82]. They divide a day d into N time slots (for instance $N = 48$) and observe the energy intake x in each time slot n. For each time slot n they iteratively compute the EWMA $\bar{x}$ using

$$\bar{x}_n^{(d)} = \alpha \bar{x}_n^{(d-1)} + (1-\alpha) x_n^{(d)},$$

where $\bar{x}_n^{(d-1)}$ is the averaged value of time slot n from the previous day. The prediction of a slot in the future is the EWMA of the observations for that slot on previous days, making use of the diurnal pattern of outdoor solar energy. This type of model is attractive for embedded systems as it only requires previous observations that can be accumulated locally, and the EWMA only requires to store one value for each of the N time slots. However, the performance of such approaches depends on the stability of weather conditions. As an improvement, Piorno et al. [198] propose a weather-conditioned moving average (WCMA), which corrects EWMA-based average values with a factor that indirectly depends on the weather. This factor is calculated based on the solar intake of the day so far, compared to that of the previous days, and hence limits the prediction to a short-term horizon, only a few time slots ahead. UD-WCMA [199] poses another improvement by choosing weighting parameters autonomously. However, the short forecasting horizon remains. Saidi et al. [200] use a Kalman fiter with an autoregressive model to predict solar energy intake, but also this approach only considers a short forecasting horizon, until the next time slot.

Persistence models are another type of forecasting model only considering past observations, which are used as baselines in solar forecasting for the power grid [201, 202]. Instead of averaging over past observations, they take the value from the previous day $d-1$ as forecast for day d. The *smart* persistence model corrects the historic observations with the diurnal variance of the solar irradiance [203]. The global horizontal irradiance (GHI) represents the potential amount of energy that can be harvested by a solar panel. It depends on the angle between the sun and the plane of the solar panel and the travel length through the atmosphere. The GHI can be calculated by using a model that estimates the clear sky global irradiance directly, such as the simplified Solis model described by Ineichen et al. [204], which is coherent for the solar elevation angles at most latitudes and calculated as

$$GHI_S = I'_o \cdot e^{\left(-\frac{\tau}{\sin^g(h)}\right)} \cdot \sin(h), \tag{D.1}$$

where I'_o is the extraterrestial irradiance modified by the atmospheric radiation component, h is the solar elevation angle, τ is the global total optical depth, and g is the corresponding fitting parameter for the GHI. Therefore, future energy intake can be predicted using

$$\hat{E}_{in}(t+24h) = \begin{cases} E_{in}(t)\frac{GHI_S(t+24h)}{GHI_S(t)}, & GHI_S(t) > 0.1 \\ E_{in}(t), & GHI_S(t) \leq 0.1 \end{cases} \tag{D.2}$$

where E_{in} is the observed energy intake at a given time t, and GHI_S is the irradiance given by the simplified Solis clear sky model. The used threshold of 0.1 can be adjusted to avoid unrealistic high levels of irradiance at sunrise and sunset.

The above-mentioned techniques, which only rely on past observations, do not perform well in locations with volatile weather and for medium-term prediction horizons, as Sharma et al. [190] point out. They instead propose the inclusion of weather forecasts in the form of cloud coverage C for the prediction of solar power P, and formulate a model $P_{sun} = P_{max} \cdot (1 - C)$, where P_{max} is approximated by a quadratic model with coefficients for each month, that are derived by manually selecting sunny days. While this work indicates the benefits of including weather forecasts, it has the drawback of manually fitting models, which is not realistic in a large-scale IoT setting with heterogeneous devices and environments.

Renner [205] combines cloud-cover information of the weather forecast with an EWMA-based model. This combination is similar to that of WCMA, but uses actual weather forecasts. IoT devices are provided with access to the cloud coverage forecast (CCF) for each time slot, and use this value to determine what corresponds to the clear-sky value for each time slot. The EWMA of these values is then used as basis that is again combined with the CCFs to compute the actual predictions. Together with the previously described methods EWMA and smart persistence (SP), we use this approach – CCF – as another baseline in Sect. D.6.

Another domain for which solar energy prediction is relevant is the power grid and renewable energy. In this domain, machine learning techniques are much more common, and used at different forecasting horizons, from short-term prediction in terms of minutes to react to fluxes of solar power, to long-term predictions to reason about the feasibility of solar installations. Voyant et al. [206] provide a comprehensive review of machine learning methods for solar radiation forecasting. For instance, Bacher et al. [207] use autoregressive models and find that for medium-term horizons, numerical weather predictions increase the accuracy considerably. Many approaches employ various machine learning techniques with the use of distinct prediction variables as input. Yadav et al. [160] provide an overview of neural networks as prediction models, while Sharma et al. [208] study machine learning based on weather forecasts, including the sky coverage, using support vector machines. Alternatively, Dahl and Bonilla [209] use Gaussian Processes as a forecasting model, which can also quantify the confidence level in the prediction estimate. Benali et al. [210] use separate models for the different components of radiation. Similarly, the use of blended learning with a mixture of models has been addressed by several authors [22, 211, 212].

Tang et al. [213] used an approach based on the least absolute shrinkage and selection operator (LASSO) for the short-term solar prediction. In [214] this approach is extended by long short-term memory units (LSTM) of neural networks into a mixture model based on different weather types. They use weather observations instead of the public weather forecasts, and identify temperature and humidity as valuable features. This is probably due to their short forecasting horizon, and the absence of cloudiness in the weather observations, as these turn out to be the most important prediction features for medium-term horizons in our analysis.

Altogether, while there is considerable attention on solar forecasting, there is a lack of discussion on the operational aspects of the prediction models relevant for IoT, which are crucial for making them work in a scalable and autonomous way for constrained devices.

D.4 Methodology

The main goal of our work is to identify machine learning models and corresponding features to increase the performance of solar energy prediction models. To that end, we start with a set of standard machine learning models which we test on an exhaustive set of feature combinations, resulting in a selection of models and features based on their performance. In the following, we discuss the significant aspects of our research method. This includes data collection, the chosen metrics, and the approaches for feature selection, prevent data leakage and ensure results under realistic conditions.

D.4.1 Solar Energy Data Collection

We use real data that we collected over more than two years, so that we were able to cover all seasons throughout a year more than once. The source of our training data is a solar panel with horizontal orientation on the top of a university building in Trondheim, Norway. We measured the voltage on a resistor and logged the data every minute, which resulted in a data set covering two years, starting in October 2017.[1] We denote each measurement with $a_i^{(d)}$, where d is the day and $i \in I_d$ the index of the value within the day. Figure D.19 shows the solar energy intake over three days, which also shows their extreme volatility from day to day.

Since measurement values are taken in regular intervals Δt (one minute in our raw data), the total energy collected during a day can be approximated by summing over the individual values $a_i^{(d)}$,

$$E(d) = \gamma \cdot \Delta t \sum_{i \in I_d} a_i^{(d)},$$

where γ is a factor including the solar panel's size, its efficiency, and the efficiency of the power converter. The specific value of this factor is not relevant here, as it depends on each individual IoT device and is an internal variable in the prediction process. Figure D.20 shows $E(d)$ over two years and reveals large seasonal difference between winter and summer. We also calculate the exponentially weighted moving average (EWMA) of the solar intake $\bar{E}(d)$ for each day

$$\bar{E}(d) = \begin{cases} E(d), & d = 1 \\ \alpha \cdot E(d) + (1 - \alpha) \cdot \bar{E}(d-1), & d > 1. \end{cases} \tag{D.3}$$

Figure D.20 shows $\bar{E}(d)$ with $\alpha = 0.095$, which also illustrates the differences within the same month from year to year. The average energy harvested in June 2018, for example, deviates significantly from the same period in the year 2019.

D.4.2 Prediction Metrics

Although various metrics to evaluate solar energy forecasting exist [215], their scaling is significant to ensure comparability, as we will show next. Two standard metrics for the prediction performance on a day d are the mean absolute error (MAE) of the individual prediction values, as well as the total absolute error for the entire day (TAE):

$$MAE(d) = \frac{1}{|I_d|} \sum_{i \in I} |a_i^{(d)} - p_i^{(d)}| \tag{D.4}$$

$$TAE(d) = \left| \sum_{i \in I_d} a_i^{(d)} - \sum_{i \in I_d} p_i^{(d)} \right|. \tag{D.5}$$

[1]The data set as well as all the code used in this article will be available after publication.

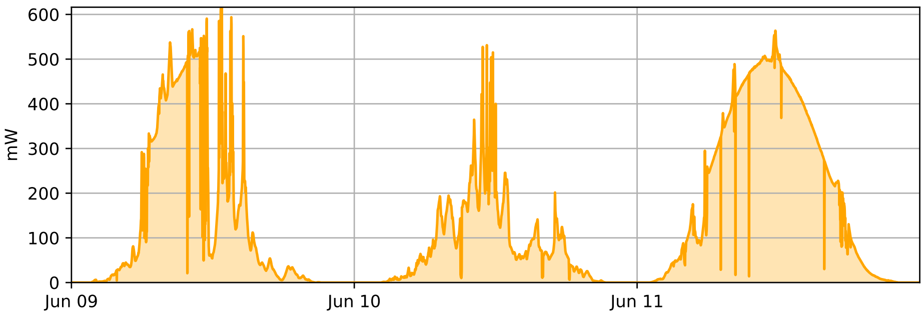

Figure D.19: Three days in June 2018 illustrating the volatility of solar energy intake from day to day.

As the MAE also considers intra-day accuracy, a good score with the MAE also implies a good TAE. Yet, considering the TAE could reveal good predictors for the overall day that are just imprecise with their timing. For IoT energy management, we will in the end address how we can combine these two aspects, but we first need to address the problem of seasonality.

Figure D.21 shows the MAE for the SP prediction model from in Sect. D.3. (The TAE shows similar behavior.) Both metrics are scale-dependent (see [216] for a discussion), and vary with the seasons. This is problematic for our purposes, as we do not know whether to attribute changes in the score of a prediction to changes of a predictor's quality or just seasonality. To eliminate this scale-dependency and seasonality, we consult the corresponding relative *percentage* errors MAPE and TAPE, which scale the error to the actual value $a_i^{(d)}$:

$$MAPE(d) = \frac{100}{|I_d^*|} \sum_{i \in I_d^*} \frac{\left|a_i^{(d)} - p_i^{(d)}\right|}{a_i^{(d)}} \tag{D.6}$$

$$TAPE(d) = \frac{100}{\sum_{i \in I_d^*} a_i^{(d)}} \left| \sum_{i \in I_d^*} a_i^{(d)} - \sum_{i \in I_d^*} p_i^{(d)} \right| \tag{D.7}$$

The modified set I_d^* includes only the indices of observations that are non-null, to prevent division by zero in (D.6). These percentage errors prevent seasonal variations, but have the drawback that errors on days with very little energy get very large, as the high variation in Figure D.21 for the MAPE shows. In line with the normed errors RMSE and ME in [205] we introduce the scaled MAPE (SMAPE) and the scaled total absolute error (STAPE), which are scaled to the moving average introduced in (D.3)

$$SMAPE(d) = \frac{100}{\bar{E}(d)} \sum_{i \in I_d} \left|a_i^{(d)} - p_i^{(d)}\right| \tag{D.8}$$

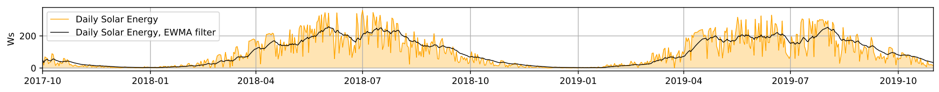

Figure D.20: Measured daily solar power intake over two years. The solid black line shows the exponentially weighted moving average (EWMA) of the daily intake with $\alpha = 0.95$, corresponding to a span of ca. 20 days. Daily values and average show the high daily and seasonal variations.

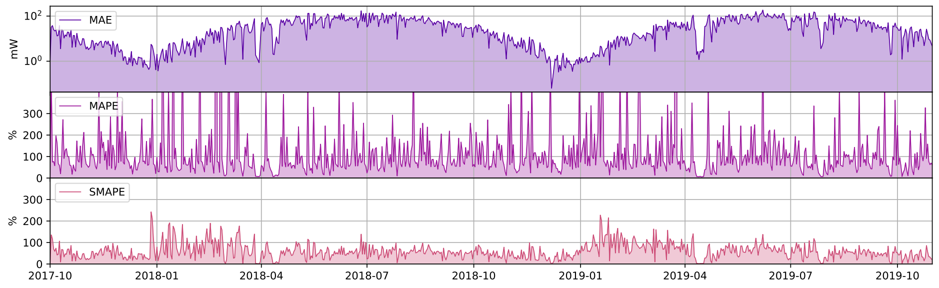

Figure D.21: Mean absolute error (MAE), mean absolute percentage error (MAPE), scaled mean absolute percentage error (SMAPE) of the smart persistence predictor. The corresponding TAE, TAPE and STAPE metrics are not shown here for brevity, but they exhibit the same characteristics as their counterparts.

$$STAPE(d) = \frac{100}{\bar{E}(d)} \left| \sum_{i \in I_d} a_i^{(d)} - \sum_{i \in I_d} p_i^{(d)} \right| . \tag{D.9}$$

Neither of them exhibit the challenges of the previous metrics. STAPE is hence a measure of how many percent, relative to the average intake during that time, the total energy for a day is off, while SMAPE is also taking into account how accurate the prediction is within a day. Taking percentages instead of absolute values, as in [205], has the benefit that scores can be also compared across different IoT devices. As a single-number metric we use the arithmetic mean between SMAPE and STAPE, which we for simplicity call the scaled absolute percentage error (SCAPE)

$$SCAPE(d) = \frac{1}{2}(SMAPE(d) + STAPE(d)). \tag{D.10}$$

This metric balances between the total prediction and the intraday accuracy.

D.4.3 Prevention of Data Leakage

A proper split between training and test data is important to prevent data leakage and ensure applicability and generalization of the results. Since the observed solar energy intake tends to be similar from one minute to the next, a standard randomized training/test split would effectively result in data leakage. For example, a measurement from 12:00 could serve as training data and an almost identical entry from 12:01 could end up as test data. We therefore only assign only entire days to the test set. For that, days of 2018 (which we use for the first parts of the experiment) are numbered consecutively, and every fourth day is taken into the holdout test set. Data from these days will not be used to train models or tune parameters, they are only used to validate the results after models have been developed. The days for validation are evenly assigned into two of ten crossfolds.

D.4.4 Feature Selection Method

Feature selection is one of the core concepts in machine learning and involves selecting the most relevant features that yield the best model performance. In an IoT setting, omitting irrelevant features is especially interesting since it may reduce training and inference time and requires to store and transmit fewer data.

Many different feature selection methods exist in the literature and they are being widely used [217]. An optimal feature selection method is the exhaustive feature search [218] The main strength of the exhaustive feature selection algorithm is that it is guaranteed to find the best set of features. However, the main drawback of this algorithm is the complexity cost.

In our context, the size of the dataset, the number of required features, and the computing resources allow the exhaustive feature selection to be computationally feasible. Therefore, to identify the best set of features, we employ an exhaustive feature selection algorithm to evaluate all possible feature combinations. More specifically, we train several thousand prediction models and evaluate them independently. Following that, we conduct a study of different features, referred to as an ablation study, to evaluate the performance impact of removing a given feature from the machine learning model.

D.4.5 Evaluation in Operational Settings

To ensure the relevance of our approach, we carried out the main evaluation of our work as a case study under quasirealistic conditions, in an operational setting further detailed in Sect. D.4.2. This means that we train the selected machine learning models with the same data they would receive if deployed in reality, and evaluate them with the metrics from Sect. D.4.2.

For a comparison of our work with current state-of- the-art solutions, we selected the baselines introduced in Sect. D.3. SP is the standard reference for solar energy forecasting [201, 202] and EWMA is a fundamental prediction technique in wireless sensor networks [82]. The CCF itself outperforms other techniques as shown in [205]. Together, these different forecasting techniques constitute a relevant baseline for our approach.

D.5 Weather-Based Machine Learning Models

In the following, we will discuss the preparation of training data in more detail and then proceed with the identification of suitable machine learning models and selection of the most valuable features, and close with the consideration of the significance of the sampling intervals for the solar intake observations.

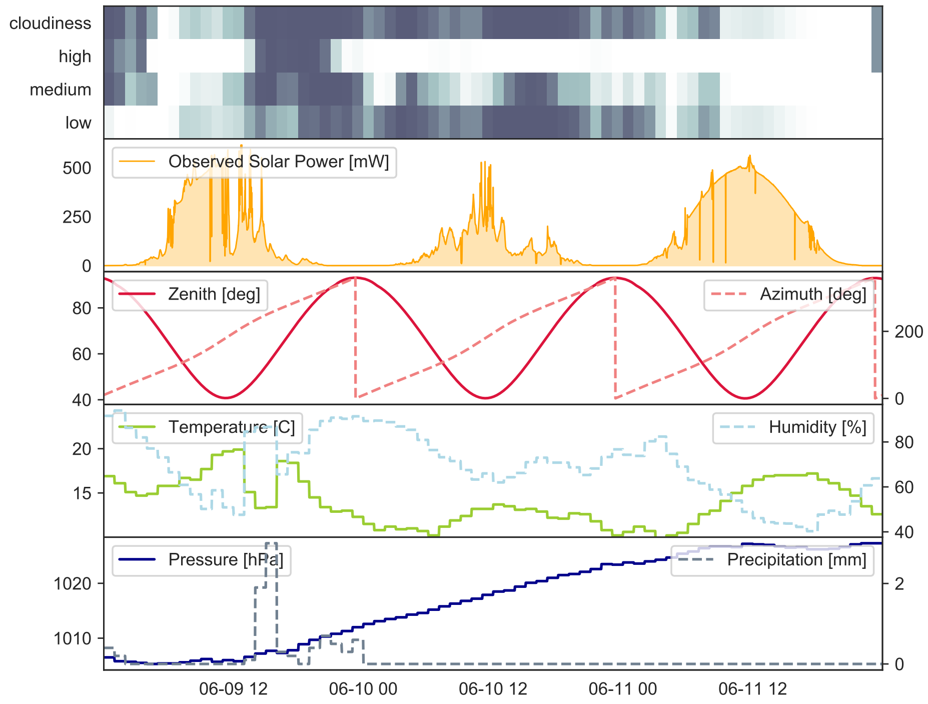

Figure D.22: Example weather forecast values for three days, together with the observed solar energy. The heatmap at the top shows the cloudiness overall and at the three levels high, medium and low. Darker shades imply more clouds.

D.5.1 Weather Forecast Data

We use the weather forecast provided by the public application programming interface (API) of the Norwegian Meteorological Institute [219]. Weather forecasts are usually issued three times a day, around every 5 to 8 hours. Each issue contains a forecast for the upcoming 60 hours with an hourly resolution. We extract, for each hour, the publication and forecast timestamps, temperature, humidity, pressure, precipitation and the amount of clouds covering the sky. Figure D.22 illustrates the weather data for three sample days. Cloudiness is provided at several levels. Internally, the weather model calculates the amount of clouds at 65 vertical levels in the atmosphere. In the forecast, we use the cloudiness at four different aggregation levels: low clouds (below 2.5 km), medium clouds (2.5–5 km), high clouds (above 5 km) as well as a total cloudiness percentage, calculated from the entire stack of cloud levels.

We merge the solar data collected every minute with the hourly slots of the weather forecast. We clean the data by dropping a negligible number of days where the weather forecast could not be collected or the solar panel was out of order. Based on the timestamp, we add the solar angles zenith and azimuth for our location. This results in a data set with the features $f_1 \dots f_{10} \in F$ that represent zenith, azimuth, temperature, precipitation, pressure, humidity, cloudiness, lowclouds, mediumclouds and highclouds. The truth value for the data set is the observed solar energy intake.

D.5.2 Feature Selection

For the exhaustive search, we use machine learning models from Scikit-Learn [220], and specifically a random forest regressor (RFR) with 30 estimators, an artificial neural network (ANN) with a single hidden layer of 100 perceptrons, and a deep neural network (DNN) with three hidden layers of 30 perceptrons each. For both neural networks, we used ReLU activation functions.

We check the performance of all the three base models on different feature sets, i.e., combinations of the features f_1 to f_{10} from F. As the zenith (feature f_1) is the dominating variable describing the position of the sun, we include it in all feature sets. The set FS of all feature sets that include f_1 is then described by

$$FS = \{x \,|\, x \in \mathcal{P}(F) \wedge f_1 \in x\}, \tag{D.11}$$

where $\mathcal{P}(F)$ is the power set of F. This results in a total of $2^9 = 512$ feature sets.

For each model and feature combination, we calculate the performance with at least 3 crossfolds and take the average of them. Figure D.23 shows the results for all models and all feature sets. Each point shows the performance in terms of SMAPE and STAPE for a specific combination of machine learning model and feature set. The different colors distinguish the different models (RFR, ANN, DNN). We observe that there is in general a strong correlation between the SMAPE and STAPE metric. The magnification to the right shows that RFR produces the best results. This is also confirmed by the histograms in Figure D.24, which shows the distribution of errors of the different models trained with different feature sets. RFR manages to achieve the best mean and median results over all feature sets.

We have tried to improve the scores of the neural networks by tuning their hyper-parameters and applying different activation functions, optimizers, and architectures regarding the hidden layers. However, we have not been able to achieve the same robust and consistent performance as with the relatively simple RFR model, which is why we continue with the RFR models in the following.

For the ablation study, we use the RFR model due to its general good performance as shown above. Based on each of the optional features $f_2 ... f_{10}$ we define pairs of feature sets, $(F^+, F^-) \in PFS_{f_i}$, where the first set F^+ includes feature f_i and the second feature set F^- does not, that means

$$\begin{aligned} PFS_{f_i} = \{(F^+, F^-) \,|\, & F^+, F^- \in FS \wedge f_i \in F^+ \\ & \wedge f_i \notin F^- \\ & \wedge F^+ \setminus \{f_i\} \equiv F^-\}. \end{aligned} \tag{D.12}$$

With the total of 512 feature sets in FS, there are 256 pairs for each of the optional features f_i. We then compute the mean performance of the prediction models for all pairs, using 10 crossfolds, and consider scatter plots as shown in Figure D.25. Each scatter plot includes 256 pairs. The x-coordinate is given by the mean SCAPE of the models trained using feature sets F^+ *with* f_i included. Correspondingly, the y-coordinate shows the mean SCAPE of the models trained

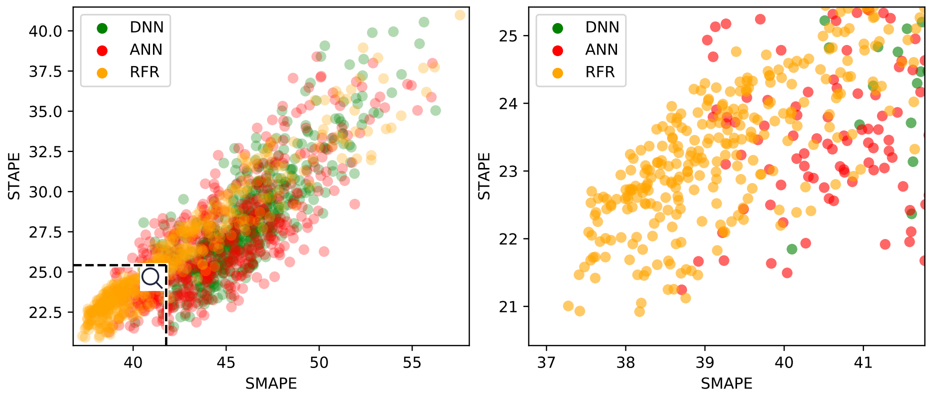

Figure D.23: SMAPE and STAPE of the various basic machine learning models for different feature sets. The right graph is a magnification as indicated by the dashed lines to the left.

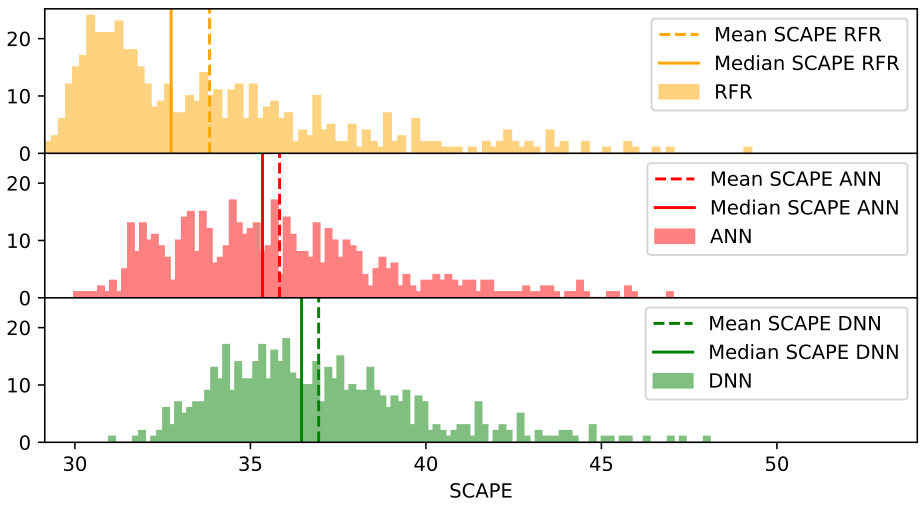

Figure D.24: Distribution of errors (SCAPE) of the three types of base models (RFR, ANN, DNN) for all 512 combinations of features. Mean SCAPE (dashed) and median (solid) errors are also shown.

on the feature sets F^{-}, i.e., *without* f_i. If a feature is useful, its inclusion should reduce the SCAPE value. Hence, points above the identity line indicate pairs where including f_i improves performance, while points below correspond to pairs where removing f_i is detrimental. Points close to the identity line show that the given feature f_i has little influence on performance. The mean SCAPE for each plot is depicted by a black cross in the intersection between the two dashed lines.

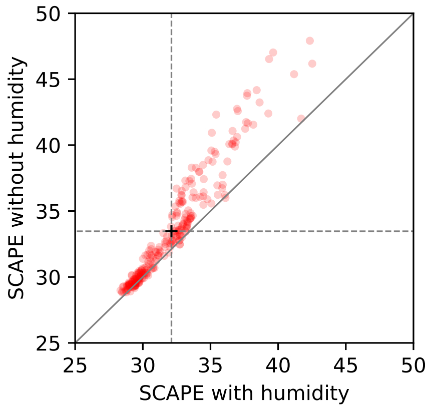

Figure D.25: Ablation study for the humidity feature. Each point denotes a feature set pair, the x-coordinate showing the SCAPE *with* humidity, the y-axis *without*. The dashed lines and cross show the mean values.

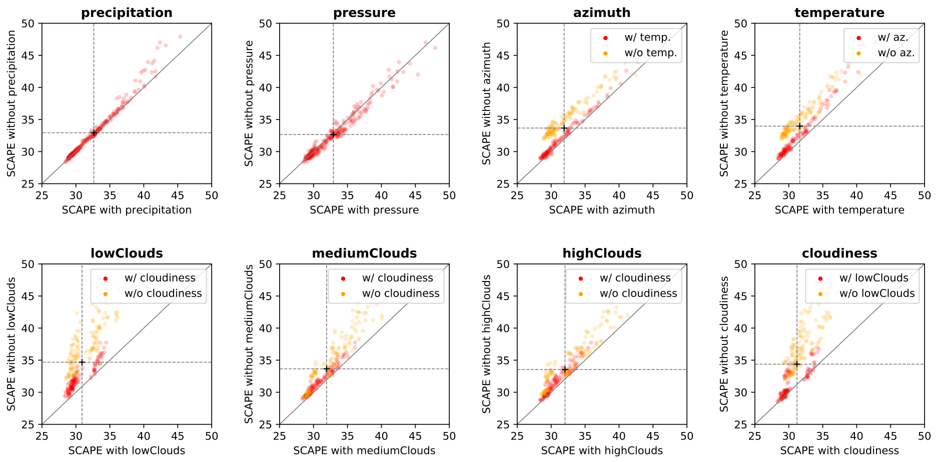

Figure D.26: Ablation study for the remaining features. Different colors are used to distinguish further certain subsets to show feature dependencies.

1) Humidity For the humidity feature in Figure D.25, feature pairs are close to the identity line, especially for good models. There are some improvements for models that score worse in general, however they yield significantly poorer results than the best feature sets not including this feature. This reveals that humidity is not a useful feature.

The scatter plots for the other features are shown in Figure D.26.

2) Precipitation Similarly to humidity, the results for the precipitation feature ablation lie mostly on the identity line, with the exception for very few of the worse performing models. This is also confirmed by the mean SCAPE result which coincides with the identity line, proving that precipitation should not be considered as a feature.

3) Pressure The pressure feature has most pairs along or even below the identity line, indicating that this feature has little usefulness, being even responsible for decreasing performance in some feature combinations. In fact, the obtained mean SCAPE is located below the identity line, confirming that using pressure as a feature is overall detrimental to the models' quality.

4)Azimuth and Temperature Azimuth and temperature are both valuable features, showing loss of performance when removed from the feature set. In particular, we found a dependency between these two features. In the azimuth plot, the orange markers are of model pairs that do not include temperature as a feature. Similarly, in the temperature feature plot, the orange markers indicate pairs where no azimuth was present. This means that the azimuth feature is especially valuable if temperature is not a feature, and vice versa. We attribute this to the often observed pattern of both the azimuth and the temperature raising in the morning (see Figure D.22). Since the temperature feature is obtained from an uncertain weather forecast while the azimuth can be calculated precisely, we consider the azimuth to be the better feature of the two and discard temperature.

5) Low-Clouds When the low-clouds feature is removed, the mean SCAPE of the RFR models increases from approximately 31 to almost 35. In addition, the variability of results increases, which explains the appearance of two vertical columns, with the SCAPE being as high as 45. The coloring of the plot distinguishes the presence of the general cloudiness feature, but interactions between the two features are not obvious here.

6) Medium-Clouds The impact of removing the medium-clouds feature is not as significant as with low-clouds but the mean SCAPE still increases when this feature is not used. The resulting variation also improves slightly, suggesting that the medium-clouds should be considered as a feature.

7) High-Clouds The obtained performance by removing the high-clouds feature is similar to the medium-clouds feature, except that a smaller variation occurs. By analyzing this third cloud-related feature we can conclude that they complement each other, even though low-clouds have a stronger correlation to the overall performance of the model.

8) Cloudiness As shown in the ablation plot, having the overall cloudiness feature improves the performance of our model similar to the low-clouds feature. We explain this with the fact that the densest clouds are found in the lower levels. If a large proportion of the sky is covered by clouds in this layer, clouds in the medium or high layers have a significant influence on how much light reaches the ground.

The dependencies between the different cloudiness features are not obvious in the pairwise comparison above, which is why we also computed the performance of all combinations of cloudiness-features, shown in Table D.3. Each row shows the mean performances and variation of the models matched with the combination of cloudiness features given to the left. The table reveals that models score similarly if they have at least three cloud-related features included. When the three cloud-levels (low, medium high) are present, the overall cloudiness feature does not contribute to

Table D.3: Average scores of all models given the availability of the various cloudiness features. Using the RFR model with 30 estimators, 10-fold cross-validation.

cloudiness	lowClouds	mediumClouds	highClouds	feature count	SCAPE	SMAPE	STAPE
	●	●	●	**3**	30.0 (0.3)	37.7 (0.3)	22.3 (0.3)
●	●	●	●	**4**	30.0 (0.3)	37.7 (0.3)	22.3 (0.3)
●	●	●		3	30.3 (0.3)	38.0 (0.3)	22.7 (0.3)
●	●		●	3	30.4 (0.3)	38.2 (0.3)	22.6 (0.3)
	●		●	**2**	30.6 (0.3)	38.5 (0.3)	22.7 (0.2)
●		●	●	3	30.8 (0.3)	38.7 (0.3)	22.9 (0.3)
●	●			2	30.9 (0.3)	38.7 (0.3)	23.1 (0.3)
	●	●		2	31.1 (0.3)	38.8 (0.3)	23.4 (0.3)
●			●	2	31.9 (0.3)	39.9 (0.3)	23.9 (0.3)
●		●		2	32.2 (0.3)	40.1 (0.3)	24.3 (0.3)
●				**1**	33.1 (0.3)	41.3 (0.4)	24.9 (0.3)
	●			1	33.9 (0.3)	41.8 (0.3)	26.0 (0.2)
		●	●	2	35.1 (0.4)	42.9 (0.4)	27.2 (0.4)
		●		1	35.9 (0.5)	44.0 (0.5)	27.8 (0.4)
			●	1	37.5 (0.5)	45.7 (0.6)	29.3 (0.5)
				0	40.9 (0.6)	49.4 (0.6)	32.5 (0.6)

any improvement. If we only want to select two cloud-related features, *highClouds* and *lowClouds* combined score best. If only one cloud-related feature should be taken into account, it should be *cloudiness*, but it scores ca. 10% worse than the best combinations.

D.5.3 Training Sample Intervals

We now examine how sensitive the machine learning models are to the intervals in which training samples are collected. Longer intervals are desirable since they reduce (i) the amount of data to transfer from the IoT devices to the machine learning models in the edge or cloud, (ii) how much data needs to be stored, and (iii) the runtime of the model fitting, i.e., training time.

For the sensitivity analysis, we prepared data sets with different sampling intervals, ranging from data sampled every minute (the original rate), to data sampled every 180 minutes, by resampling and taking the mean values. For each sampling rate, we trained RFR models for each of the 10 crossfolds, using the feature sets consisting of zenith, azimuth and all cloudiness features. We evaluated the resulting models with the test data from the original 1-minute sampled raw data. Figure D.27 shows the mean SMAPE and STAPE for the 10-folds including their standard error. Interestingly, the lowest sampling interval of 1 minute does not result in the most accurate models measured by either SMAPE or STAPE. Instead, we see an optimum for the SMAPE in the range between 30 to 50 minutes, before the accuracy of the prediction decreases again with growing sampling intervals. We attribute this to the short-term fluctuations in the cloud coverage that are present in the more fine-grained data, which are not represented in the hourly weather forecast. For an operational setting, this is significant: more data does not imply higher accuracy, and by choosing a sampling interval closer to 30 minutes we can both increase the accuracy of the prediction and save costs in transmission, data storage and training time.

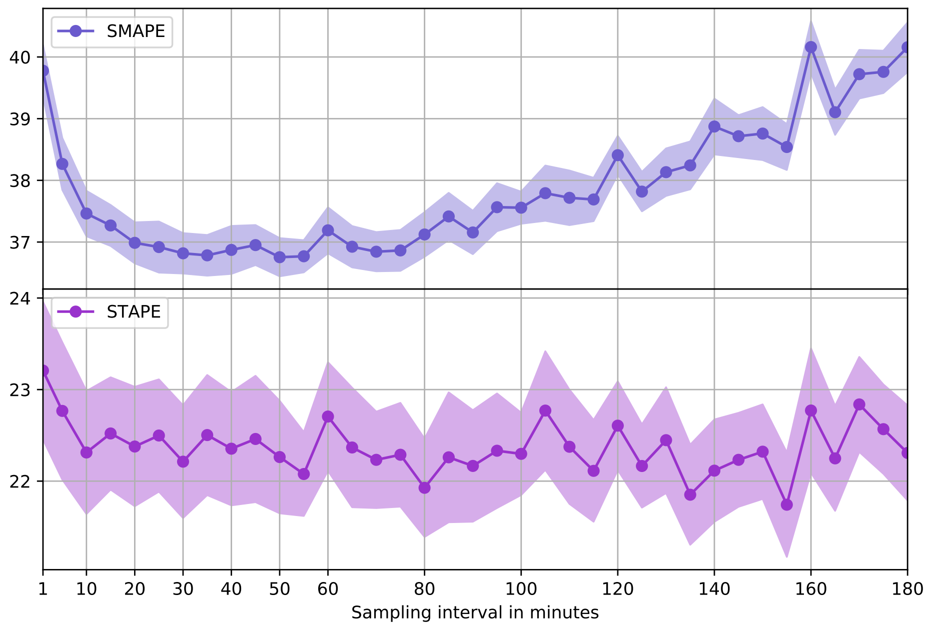

Figure D.27: Scores for models trained with data sampled at different rates. While the STAPE is fairly unaffected once sampling intervals are longer than 10 minutes, the SMAPE is getting better in the range between 30 and 50 minutes.

D.6 Day-to-Day Operation and Evaluation

To ensure the relevance of our results, we now study the performance of the prediction in an operational setting. That means that devices are newly deployed without prior data, and machine learning models are re-trained continuously as training data becomes available. In the previous sections, we analyzed suitable machine learning models, feature sets and sampling frequencies. For the operational settings, we chose trade-offs that allow for quick computation and good performance. We hence chose the random forest regressor (RFR), with *zenith*, *azimuth*, *cloudiness*, *lowClouds*, *mediumClouds*, *highClouds*, as features, and a 30-minute sampling frequency. As training and test data we use the so far unused data of the year 2019.

We assume that an IoT device n is newly deployed on day d^n_{dep} and use d^n_i to describe the i-th day since the deployment of n. At the end of each day d^n_i, the device manager computes a new model M^n_i with the training data from the previous days since deployment, i.e.,

$$\begin{aligned} train^n_i = \{d \mid d \geq d_{dep} \ \wedge \ d \leq d_i + d_{dep} \\ \wedge \ d \geq d_i + d_{dep} - train_{max}\}. \end{aligned}$$

The last conjunct constraints the use to only the last $GHItrain_{GHImax}$ days for training data. Since a device is deployed without prior knowledge, the model M^n_i only contains training data collected from d^n_{dep} to d^n_i. The model M^n_i created for device n at the end of day d_i^n is then used to create a new energy prediction for day d_{i+1}^n, using the public weather forecast and the solar angles that are calculated from time and location. To ensure a realistic and causal setting, we only use weather forecasts for day d_{i+1}^n that were published on the previous day d_i^n. We then calculate error metrics for all days $d_i^n, i > 0$.

D.6.1 Amount of Training Data

As the computational effort of model fitting increases with the number of training samples and hence with increasing $train_{max}$, we examine first how the prediction quality of M_i^n develops with the number of included training days. For that, we calculate the results for $n = 365$ devices, each deployed at a different day in 2019,[2] and study the results of M_i^{355} when predicting day d_i^{365} for increasing $train_{max}$. Figure D.28 shows how the SCAPE reduces with a growing number of training days $train_{max}$. Mean and median error reduce quickly with an increasing amount of training data. Already after around 30 to 50 days with training data the SCAPE is on a level that only marginally reduces with further data.

We have also experimented with data augmentation, similar to [221], which takes the existing training data and generates additional data points through various techniques. However, this approach could not further reduce the number of required training days as desired.

D.6.2 Operational Setting and Comparison

For the final evaluation and comparison with other baselines, we now examine how the metrics evolve over time, i.e., starting with one day of training data on deployment day d_1, and going forward using a maximum of $train_{max} = 30$ days. Figure D.29 shows the results of the operational setting. Again, we simulated N=365, corresponding to the deployment of devices at different days of the year 2019.

Table D.4 compares the results of our proposed approach (RFR) with the approaches described in Sect. D.3. For the EWMA-based approach based on [82] we used 48 timeslots and $\alpha = 0.7$. Similarly, we used the same setting for the approach using cloud-cover forecasts (CCFs), as they provided the best results in [205]. (SP) denotes the smart persistence model. The numbers show the mean and the median of the metrics over all 365 prediction days. As expected, the methods only relying on past data (EWMA and SP) perform worst. The results of CCF show the benefit of taking cloud coverage forecasts into account. The consistently best results are achieved by our proposed RFR model, which scores more than 20 % better than the CCF model for the median SCAPE.

RFR and CCF receive the same weather forecast, but CCF only utilizes the overall cloudiness feature while RFR also takes the other cloudiness features and solar angles into account. The remaining error for RFR is comparatively low, and we suspect that most of it is due to imprecission of the weather forecast, i.e., an inherent problem that only reduces with more accurate weather predictions.

[2]We arranged the days of the year in a circular way, so that periods going beyond the year end draw their days from the beginning of the year.

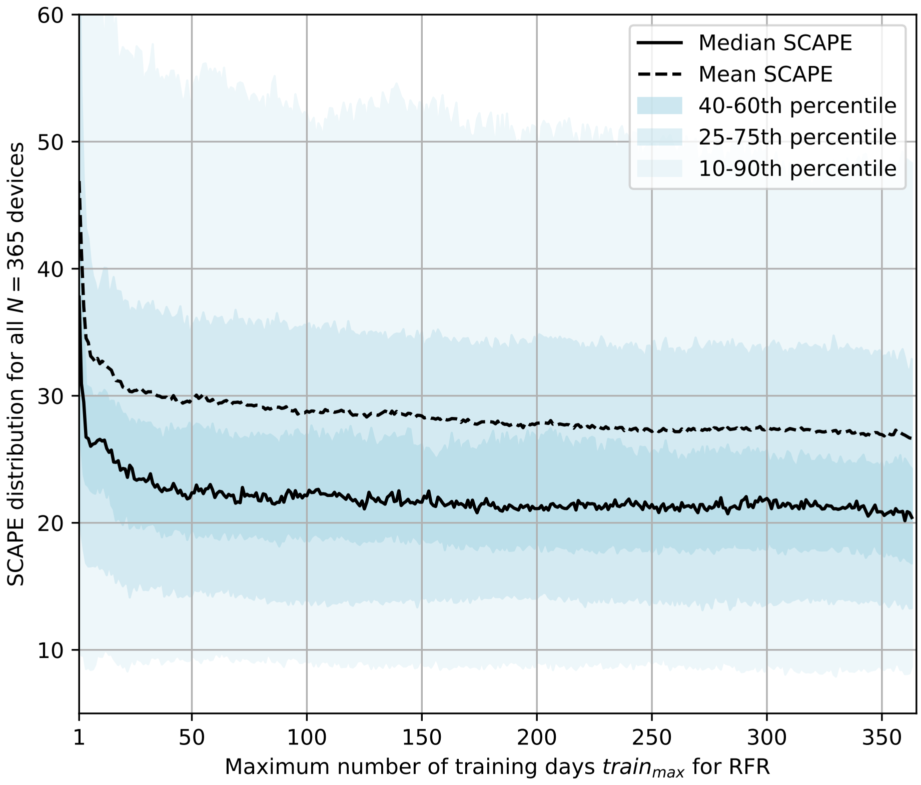

Figure D.28: Distribution of SCAPE with growing number of training days, for N=365 deployments.

D.6.3 Computational Effort

One argument for the EWMA-based approaches is their computational simplicity. We argue, however, that the computational effort needed for the RFR is insignificant in a modern IoT system with device management. The effort for storing training data and training the models is low on cloud- or edge-platforms, especially when compared with computation and storage needed for the actual application data in an IoT system. For example, with the setting chosen above ($train_{max} = 30$, sampling rate 30 minutes and all cloud-related features), training time on a PC-grade CPU is on average below 400 ms per day and per device. The prediction time for a single day with 48 time slots is on average below 9 ms. This seems to be acceptable given the potential gains in prediction accuracy that further increase the energy efficiency of the IoT device, which is a much more urgent problem.

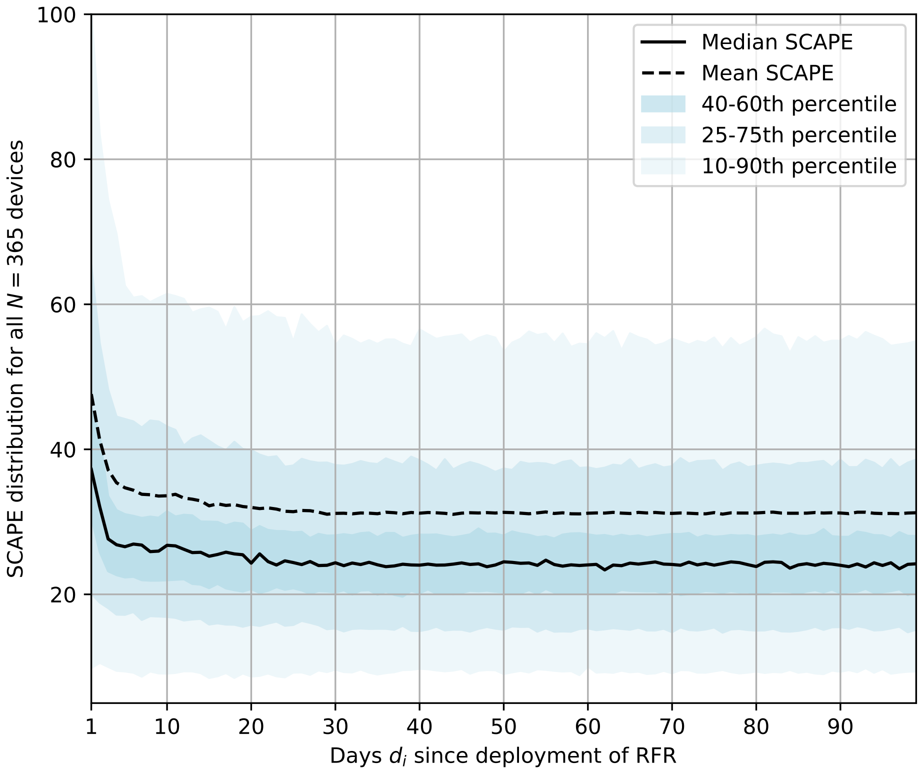

Figure D.29: Distribution of SCAPE for N=365 nodes starting without prior data with growing days d_i since deployment, and a maximum of $train_{max} = 30$ training days.

D.7 Conclusion

We presented and evaluated an approach for solar power energy prediction that is suitable for the application in solar-powered IoT systems. The median prediction scores are more than 20 % better than the current state of the art for IoT energy prediction, which we consider significant for the effectiveness of energy budget planning. The input for the prediction models only uses the public weather forecast and solar angles derived from the current time and location, that means, only data that are easily available. The approach scales well in an IoT setting. No manual tuning was necessary for the individual IoT devices despite any differences, for instance in solar panel size, as these differences are learned by the individual machine learning models. The problem of individual adaptation is hence solved by individual but autonomously learning models.

Once part of the device management, energy harvesting models with better performance open up for more strategic energy planning. As indicated in Sect. D.1, IoT devices can then more proactively schedule energy-intensive tasks in predicted

Table D.4: Results of the prediction models in the operational setting.

	SCAPE		SMAPE		STAPE	
	mean	median	mean	median	mean	median
SP	44.79	38.84	50.56	47.67	39.01	32.96
EWMA	40.30	33.19	45.48	41.06	35.13	27.16
CCF	34.49	30.63	41.53	38.47	27.46	21.11
RFR	**31.20**	**24.10**	**38.00**	**31.50**	**24.40**	**16.10**
Error Reduction from CCF to RFR	-9.54 %	-21.32 %	-8.50 %	-18.12 %	-11.14 %	-23.73 %

periods of energy surplus, avoiding over-dimensioning of systems. This has the potential to reduce the size of solar panels and energy buffers that IoT devices need to operate in a perpetual way, further advancing sustainable operation.

References

[22] Braten, A. E. and Kraemer, F. A. “Towards cognitive IoT: Autonomous prediction model selection for solar-powered nodes”. In: *2018 IEEE international congress on internet of things (ICIOT)*. 2018, pp. 118–125.

[23] Braten, A. E., Kraemer, F. A., and Palma, D. “Adaptive, correlation-based training data selection for IoT device management”. In: *Sixth international conference on Internet of Things: Systems, management and security (IOTSMS)*. 2019.

[82] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. “Power management in energy harvesting sensor networks”. In: *ACM transactions on embedded computing systems (TECS)* vol. 6, no. 4 (2007).

[150] Ahlers, D., Driscoll, P. A., Kraemer, F. A., Anthonisen, F. V., and Krogstie, J. “A Measurement-Driven Approach to Understand Urban Greenhouse Gas Emissions in Nordic Cities”. In: *NIK Norsk Informatikkonferanse* (Nov. 2016), pp. 1–12.

[159] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N., and Palma, D. “Solar energy prediction for constrained IoT nodes based on public weather forecasts”. In: *Proceedings of the Seventh International Conference on the Internet of Things*. New York, New York, USA, 2017, pp. 1–8.

[160] Yadav, A. K. and Chandel, S. “Solar radiation prediction using Artificial Neural Network techniques: A review”. In: *Renewable and Sustainable Energy Reviews* vol. 33 (2014), pp. 772–781.

[184] Shaikh, F. K. and Zeadally, S. “Energy harvesting in wireless sensor networks: A comprehensive review”. In: *Renewable & Sustainable Energy Reviews* vol. 55 (2016), pp. 1041–1054.

[185] Caruso, A., Chessa, S., Escolar, S., Toro, X. del, Kuzman, M., and Lopez, J. C. “Experimenting Forecasting Models for Solar Energy Harvesting Devices for Large Smart Cities Deployments”. In: *2019 IEEE Symposium on Computers and Communications (ISCC)* (2019), pp. 1177–1182.

[186] Hanschke, L., Heitmann, J., and Renner, C. “Challenges of WiFi-Enabled and Solar-Powered Sensors for Smart Ports”. In: *Proceedings of the 4th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems* (2016).

[187] Farooq, M. S., Riaz, S., Abid, A., Abid, K., and Naeem, M. A. “A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming”. In: *IEEE Access* vol. 7 (2019), pp. 156237–156271.

[188] Connection, R. *Prevent illegal deforestation.* 2019. URL: https://rfcx.org/our_work#deforestation (visited on 12/01/2019).

[189] Liu, J., Faulkner, G. E., Choubey, B., Collins, S., and O’brien, D. “An Optical Transceiver Powered by On-Chip Solar Cells for IoT Smart Dusts With Optical Wireless Communications”. In: *IEEE Internet of Things Journal* vol. 6 (2019), pp. 3248–3256.

[190] Sharma, N., Gummeson, J., Irwin, D. E., and Shenoy, P. J. “Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems”. In: *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)* (2010), pp. 1–9.

[191] Renner, C., Unterschutz, S., Turau, V., and Romer, K. “Perpetual Data Collection with Energy-Harvesting Sensor Networks”. In: *ACM Trans. Sens. Networks* vol. 11 (2014), 12:1–12:45.

[192] Yau, C.-W., Kwok, T. T.-O., Lei, C.-U., and Kwok, Y.-K. “Energy Harvesting in Internet of Things”. In: *Internet of Everything.* 2018.

[193] Edalat, N. and Motani, M. “Energy-aware task allocation for energy harvesting sensor networks”. In: *EURASIP Journal on Wireless Communications and Networking* vol. 2016 (2016), pp. 1–14.

[194] Cionca, V., Mcgibney, A., and Rea, S. “MAllEC: Fast and Optimal Scheduling of Energy Consumption for Energy Harvesting Devices”. In: *IEEE Internet of Things Journal* vol. 5 (2018), pp. 5132–5140.

[195] Zou, T., Lin, S., Feng, Q., and Chen, Y. “Energy-Efficient Control with Harvesting Predictions for Solar-Powered Wireless Sensor Networks”. In: *Sensors (Basel, Switzerland)* vol. 16 (2016).

[196] Kraemer, F. A., Alawad, F., and Bosch, I. M. V. “Energy-Accuracy Tradeoff for Efficient Noise Monitoring and Prediction in Working Environments”. In: *Proceedings of the 9th International Conference on the Internet of Things* (2019).

[197] Ashraf, N., Hasan, A., Qureshi, H. K., and Lestas, M. “Combined Data Rate and Energy Management in Harvesting Enabled Tactile IoT Sensing Devices”. In: *IEEE Transactions on Industrial Informatics* vol. 15 (2019), pp. 3006–3015.

[198] Piorno, J. R., Bergonzini, C., Alonso, D. A., and Rosing, T. S. "Prediction and management in energy harvested wireless sensor nodes". In: *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology* (2009), pp. 6–10.

[199] Dehwah, A. H., Elmetennani, S., and Claudel, C. G. "UD-WCMA: An energy estimation and forecast scheme for solar powered wireless sensor networks". In: *J. Netw. Comput. Appl.* vol. 90 (2017), pp. 17–25.

[200] Saidi, K., Ajib, W., and Boukadoum, M. "Adaptive transmitter load size using receiver harvested energy prediction by Kalman filter". In: *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)* (2016), pp. 1–5.

[201] Kaur, A., Nonnenmacher, L., Pedro, H. T. C., and Coimbra, C. F. M. "Benefits of solar forecasting for energy imbalance markets". In: *Renewable Energy* vol. 86 (2016), pp. 819–830.

[202] Antoñanzas, J., Osório, N. M., Escobar, R. A., Urraca, R., Martinez-de-Pison, F. J., and Antoñanzas-Torres, F. "Review of photovoltaic power forecasting". In: *Solar Energy* vol. 136 (2016), pp. 78–111.

[203] Pedro, H. T. C. and Coimbra, C. F. M. "Assessment of forecasting techniques for solar power production with no exogenous inputs". In: *Solar Energy* vol. 86 (2012), pp. 2017–2028.

[204] Ineichen, P. "A broadband simplified version of the Solis clear sky model". In: *Solar Energy* vol. 82 (2008), pp. 758–762.

[205] Renner, C. "Solar harvest prediction supported by cloud cover forecasts". In: *ENSSys '13*. 2013.

[206] Voyant, C., Notton, G., Kalogirou, S. A., Nivet, M. L., Paoli, C., Motte, F., and Fouilloy, A. "Machine learning methods for solar radiation forecasting: A review". In: *Renewable Energy* vol. 105 (2017), pp. 569–582.

[207] Bacher, P., Madsen, H., and Nielsen, H. A. "Online short-term solar power forecasting". In: *Solar Energy* vol. 83 (2009), pp. 1772–1783.

[208] Sharma, N., Sharma, P., Irwin, D. E., and Shenoy, P. J. "Predicting solar generation from weather forecasts using machine learning". In: *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)* (2011), pp. 528–533.

[209] Dahl, A. and Bonilla, E. V. "Grouped Gaussian processes for solar power prediction". In: *Machine Learning* vol. 108 (2018), pp. 1287–1306.

[210] Benali, L., Notton, G., Fouilloy, A., Voyant, C., and Dizène, R. "Solar radiation forecasting using artificial neural network and random forest methods: Application to normal beam, horizontal diffuse and global components". In: *Renewable Energy* (2019).

[211] Feng, C. and Zhang, J. "Hourly-Similarity Based Solar Forecasting Using Multi-Model Machine Learning Blending". In: *2018 IEEE Power & Energy Society General Meeting (PESGM)* (2018), pp. 1–5.

[212] Fouilloy, A., Voyant, C., Notton, G., Motte, F., Paoli, C., Nivet, M. L., Guillot, E., and Duchaud, J.-L. "Solar irradiation prediction with machine learning: Forecasting models selection method depending on weather variability". In: *Energy* (2018).

[213] Tang, N., Mao, S., Wang, Y., and Nelms, R. M. "Solar Power Generation Forecasting With a LASSO-Based Approach". In: *IEEE Internet of Things Journal* vol. 5 (2018), pp. 1090–1099.

[214] Wang, Y., Shen, Y., Mao, S., Chen, X., and Zou, H. "LASSO and LSTM Integrated Temporal Model for Short-Term Solar Intensity Forecasting". In: *IEEE Internet of Things Journal* vol. 6 (2019), pp. 2933–2944.

[215] Zhang, J., Florita, A. R., Hodge, B.-M. S., Lu, S., Hamann, H. F., Banunarayanan, V., and Brockway, A. M. "A suite of metrics for assessing the performance of solar power forecasting". In: *Solar Energy* vol. 111 (2015), pp. 157–175.

[216] Hyndman, R. J. and Koehler, A. B. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* vol. 22 (2006), pp. 679–688.

[217] Chandrashekar, G. and Sahin, F. "A survey on feature selection methods". In: *Comput. Electr. Eng.* vol. 40 (2014), pp. 16–28.

[218] Kira, K. and Rendell, L. A. "The Feature Selection Problem: Traditional Methods and a New Algorithm". In: *AAAI Conference on Artificial Intelligence*. 1992.

[219] institutt, M. *Free meteorological data*. 2019. URL: https://www.met.no/en/free-meteorological-data (visited on 12/01/2019).

[220] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* vol. 12 (2011), pp. 2825–2830.

[221] Li, Y., Hu, H., and Zhou, G. "Using Data Augmentation in Continuous Authentication on Smartphones". In: *IEEE Internet of Things Journal* vol. 6 (2019), pp. 628–640.

Towards Cognitive IoT: Autonomous Prediction Model Selection for Solar-Powered Nodes

Anders Eivind Bråten, Frank Alexander Kraemer

Abstract

The future devices connected to the IoT must have the ability to solve their tasks intelligently, optimize their operations and adapt to changes autonomously. Machine learning, executed as part of device management, is the key to such intelligence. We discuss an evolution of device management systems, that also takes meta-knowledge into account, i.e., has the ability to manage the learning in the system. This is a step towards the vision of *Cognitive IoT*. As one important function for device management, we identified the selection among various prediction models. As an example, we use autonomous energy management of solar-powered sensor devices in an environment with volatile weather and seasonal changes. We investigate an algorithm that selects among a set of prediction models based on historic performance. Our results show that even though a classical machine learning model is able to learn quickly in some periods, it has trouble generalizing well over the data in other periods, compared to a simple physical model. We show that providing a system with the ability to select among a set of predictors can mitigate the bootstrapping problem for constrained devices and also help them stay in operation in periods when training data is missing.

E.1 Introduction

The next leap for IoT applications is a closer integration of artificial intelligence and machine learning with the fabric of the IoT system. By acting more intelligently, the system will not only increase its utility to its domain, but also be able to manage its resources more efficiently and act autonomously, i.e., require less human control. Wu et al. [61] describe this as *Cognitive IoT*. One challenge of this vision are the constraints within typical IoT devices that restrict the amount of computation and storage, two generally important factors for the application of machine learning. A solution to resource constraints in the IoT nodes is to offload learning processes into the cloud [12] or fog [222], where resources are virtually unlimited. This opens a path towards the vision of Cognitive IoT through an evolution of cloud-based device management platforms into what we refer to as *Cognitive Device Management* platforms.

Over the last years, several IoT platforms have evolved, and are now an integral part of any IoT solution [105]. In their current form, most of them focus on the enablement of remote device management, illustrated in Fig. E.30. Besides taking care of data aggregation, device management platforms also offer support for the operation of IoT devices, including monitoring their operational status, for instance their battery level, location and firmware version. They also manage the remote update of configurations and firmware. This ensures that bug fixes, security patches and new features can be deployed remotely. To address the large number of devices as part of a deployment, device management allows to address many devices at the same time, perform batch updates, addressing for instance all devices with a certain firmware version or within a given physical location [114]. This still constitutes a rather mechanic approach towards device management that is focused on remote operation. Specific situations still need to be addressed for each device

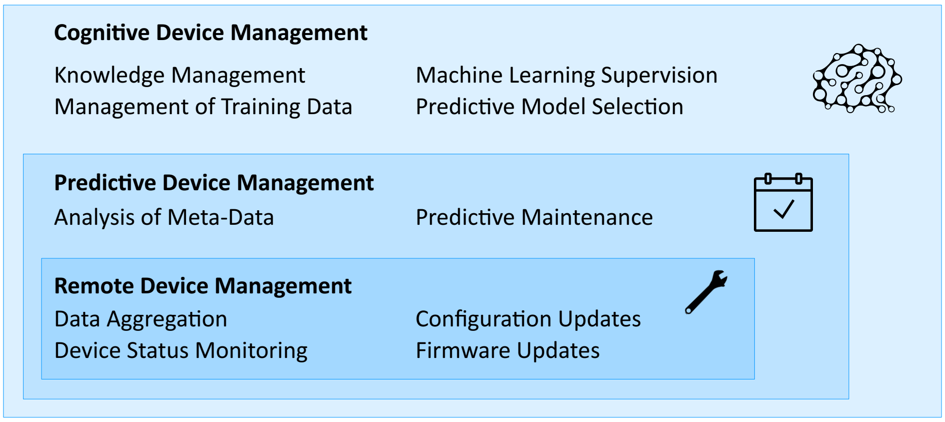

Figure E.30: Evolution of remote device management towards a Cognitive IoT

individually [223]. To mitigate this problem, device management can be extended towards *predictive* maintenance, which also automatically analyzes the operational data, with the aim to detect anomalies or trends that require action, to diagnose problems as early as possible [224, 102]. Instead of just logging the state of the battery, for instance, predictive management can foresee when a device requires maintenance or replacement in the future.

In many settings, even predictive analytics is not enough to ensure proper and scalable operation of an IoT deployment. Instead, IoT applications need to close the autonomic and cognitive loop, and become autonomous [52] and self-managed. For nodes subject to energy harvesting, for instance, this means to keep track of the current state of the energy buffer, predict their future energy supply, and have the ability to select the best viable strategy to operate energy neutral in the long run [82]. This can be treated as classical optimization and learning problems. However, due to the scale of IoT, a manual supervision of these learning processes is unfeasible. Instead, a device management system needs to be able to manage these learning processes, i.e., operate on a meta-learning level. For instance, it needs to monitor the knowledge that is available about a device, its operation and its environment, manage training data, and supervise machine learning processes.

Within this area, we have explored the mechanism of selecting the best available prediction model at a given time (also called *predictive model selection*), and identified this as an important task for cognitive IoT device management. In particular, we show how such a model selection can be performed autonomously, and how a set of models can solve the bootstrapping problem during the start of a new system when training data is scarce or non-existent. We discuss the tuning of the model selection to manage the tradeoff between reactivity and stability. We also show how proper model selection can contribute to the robustness of a system and provide some degree of self-healing. The experiments underpinning our selector and the discussions are done in a realistic setting and with real data.

In the following, we describe the concept of cognitive device management, motivated by a case study of autonomous energy management, explained in Sect. 2. We detail the functions of cognitive device management in Sect. 3 and present a set

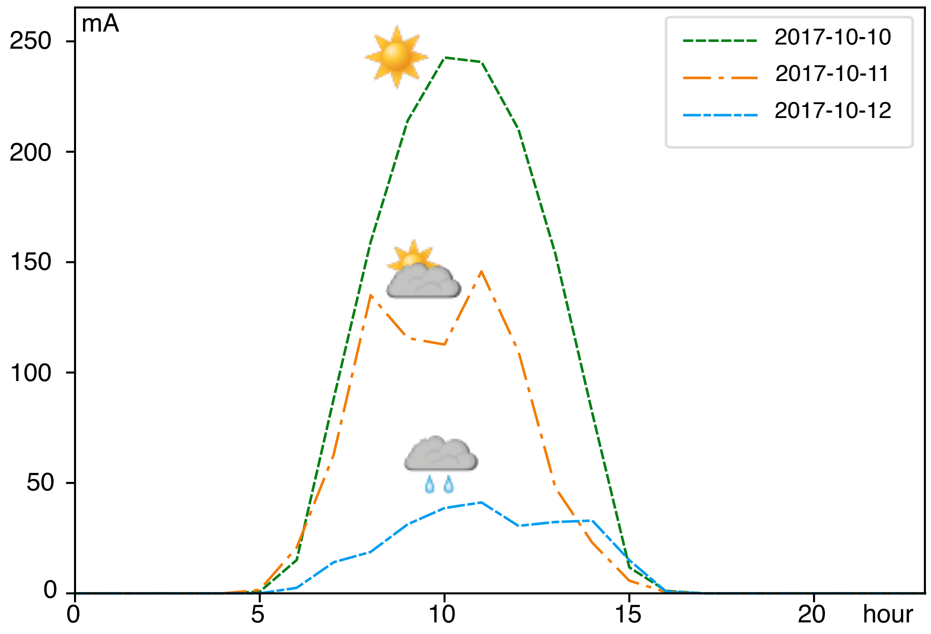

Figure E.31: Actual produced solar power, in mA, for three consecutive days in October

of prediction models for solar energy in Sect. 4. In Sect 5 we describe an algorithm to select among the different prediction models so the overall accuracy is improved, especially during the critical bootstrapping phase of the system. We conclude with a discussion of our results and related approaches.

E.2 Machine Learning for Energy Management

We illustrate the concept of cognitive device management by means of energy planning for constrained IoT sensors. The IoT sensors are Waspmotes, off-the-shelf components from Libelium intended for monitoring air quality, in particular NO_x, CO_2 and dust. To simplify deployment, they are powered by solar panels, with a lithium polymer battery as energy buffer. During deployment [150] it became apparent that the energy harvesting opportunities under Nordic conditions varies significantly between summer and winter. CO_2 and dust measurement require a significant amount of energy, because the sensors require heating and a motor for airflow, respectively. Using more energy than can be harvested obviously results in a declining battery charge and eventual node failure. At the same time, saving too much energy is not optimal either; once the battery is full, any further arriving solar energy beyond current consumption is lost, and cannot be spent to collect more data, or improve data quality. For this reason, planning the future energy budget pays off [174].

In Norway, seasonal changes between summer and winter are drastic. Collecting measurements every 5 minutes during summer is no problem, but it is necessary to

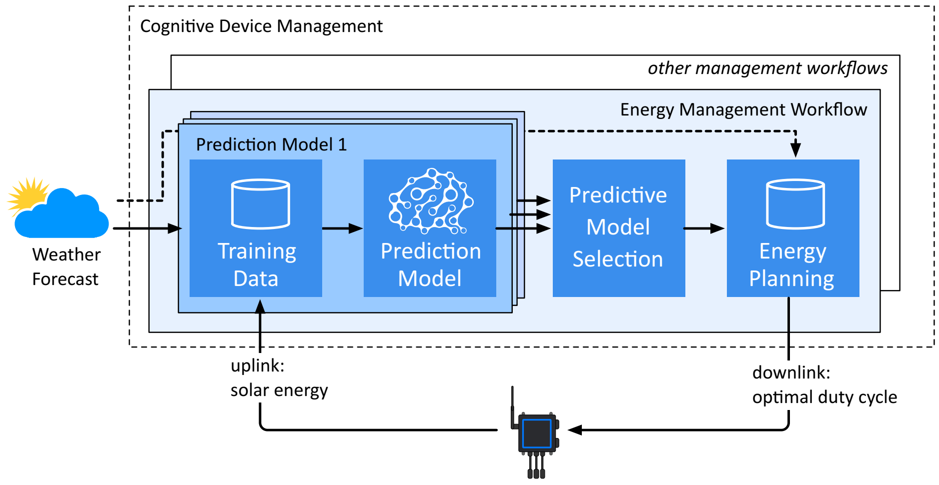
Cognitive Device Management
other management workflows
Energy Management Workflow
Prediction Model 1
Training Data
Prediction Model
Predictive Model Selection
Energy Planning
Weather Forecast
uplink: solar energy
downlink: optimal duty cycle

Figure E.32: Workflow for energy planning

reduce the sampling rate to around once every two hours during winter, when days are short and the sun barely looms over the horizon. In addition, the available energy also varies considerably with the current weather from day to day, especially with the level of cloudiness [225]. Fig. E.31 shows three consecutive days in October. The area under the curve is proportional to the harvested energy, and varies considerably.

To improve prediction, we describe in [159] how machine learning can predict the amount of energy available in the future based on public weather forecasts, illustrated in Fig. E.32. Since the Waspmotes are only equipped with an 8-bit micro controller, the machine learning as well as the prediction and planning are offloaded in the cloud, as part of the device management. The sensor nodes communicate via the low-bandwidth LoRaWAN protocol. With every payload message, the sensor nodes also attach the amount of harvested solar energy. This data is combined with the current weather forecast and makes up the training data for a regression model. Details about this model are discussed in Sect. E.4. The trained model is then used as a predictor for a planning algorithm that determines a good sensing strategy for the sensor node, so that it achieves energy-neutral operation, ENO [82]. This property secures an optimal utilization of the available energy relative to the data collected.

While the prediction based on machine learning works well, it also comes with additional tasks for the operation of the system, which we will discuss in the following.

E.3 The Need for Cognitive Device Management

We argued in the previous section that optimal control of an IoT system requires functions that go beyond remote device management, and also include more advanced resource planning and adaptation. Consistent with the vision of cognitive IoT, these functions can be supported by various machine learning models, the one for

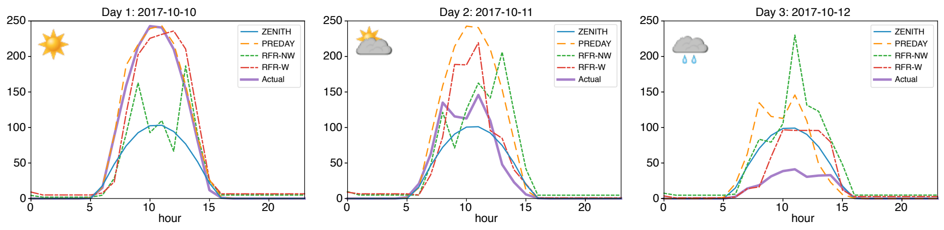

Figure E.33: Predicted and actual produced solar power, in mA, for three consecutive days in October

solar energy prediction taken as example in this paper is only one of many. Other examples are optimal routing strategies, selection of radio interfaces, decisions to offload computation, to name a few [12].

This expands the tasks for device management, which now also has to take the acquisition of knowledge and the management of it into account. The autonomous orchestration and supervision of such a workflow is done by a *cognitive* device manager. While there can be several management tasks, we focus in the following only on energy management.

For each workflow, there are a number of functions to monitor and control:

- A first task is to manage the collection of meta-data concerning the device's operation, in addition to domain data, as this is also a source for machine learning.
- A second task is to trigger the machine learning process, i.e., model training, whenever significantly new training data becomes available.
- A third task is to run new iterations of energy planning, as shown in Fig. E.32. This can for instance be triggered by the availability of updated weather forecasts for the next hours and days, or by the detection of large discrepancies between assumed and actually observed energy budget.

These tasks are on a meta-level compared to the primary tasks of the IoT application. Not only do they regard knowledge about the system's own operation, but they also consider how much knowledge the system has about itself, i.e., perform the meta-learning. Similar to other functions of IoT device management, these operations must operate constantly and autonomously, without supervision or interventions. Their tight integration with the device state and communication suggests an integration as part of the device management [55].

Machine learning models are of statistical nature and depend on the availability of sufficient training data. This means that a model cannot make an accurate prediction when it doesn't have sufficient data [226]. When the operation of the system depends on the prediction, this implies a bootstrapping problem: We need a prediction model to start the system, but good prediction models will only be available after we have collected enough training data, as shown in [227]. Hence, a cognitive device manager needs to provide self-configuration and self-optimization already in the deployment phase. To solve this problem, we employ a set of models instead, so that the statistical models dependent on training data are complemented by one or more physical models that don't require training data. The task of a cognitive device manager is then to monitor the accuracy of these models, and find the model that is best suited to support its current operation, in its current context, at any given time. Intuitively, the system will start with the models initially available and switch over to more advanced ones when they become available and accurate enough.

Having a set of models instead of only a single one also makes the system more robust. A cognitive device manager that detects when the accuracy of a model suddenly drops, can switch to a backup model. This may happen when necessary input data is missing, for instance a missing weather forecast, or if the environment of a sensor changes, for instance when a solar panel is turned into a different orientation. We will now take a closer look into different models, the metrics to rank them and how to implement selection algorithm.

E.4 Set of Prediction Models

In the following, we describe a set of models for the prediction of solar power. The models are of different natures, i.e., physical or statistical.

In previous experiments [159][228], we have observed that on average the available solar energy correlates with the sun's angle and the cloudiness, as visible in Fig. E.33. A simple physical model, based on the sun's zenith, is therefore the following:

$$I_{\text{ZENITH}} = \begin{cases} I_{\text{MAX}} \cos(\theta_s) & \theta_s > 0 \\ 0 & \text{otherwise} \end{cases} \tag{E.13}$$

where I_{MAX} is the maximum current of the solar panel, in this case set to 315 mA. The sun angle θ_s is calculated from the geographic coordinates and the time. Fig. E.33 shows the predictions, compared with the actual solar power. The actual measured solar energy is shown by the solid bold line. Since the ZENITH model is independent of weather, it underestimates the available solar energy on sunny days (day 1), scores well on average days (day 2), and overestimates on days with dense clouds (day 3).

PREDAY is a very different model and of a more statistical nature, since it simply uses data observed from a day d_i as prediction for day d_{i+1}. The benefit is that the model does not require any domain knowledge or specific considerations. However, the model only works from the second day onwards and its prediction accuracy highly depends on the stability of the weather conditions [173]. The sunny conditions on day 1 in Fig. E.33, for instance, lead to an overestimation on day 2.

To take the weather into account, we employ a machine learning model *RFR-W* that is trained with a random forest regressor, using the sun and a hourly public weather forecast as input features. The weather forecast consists of the cloudiness, and a weather symbol (as a discrete value). This model can be very accurate, but only with precise weather forecast and sufficient training data. The model was trained for each day, only taking past observations into account. For a more detailed discussion of this prediction model, we refer to [159]. Similar to the *PREDAY* model, it only delivers predictions from the second day onwards.

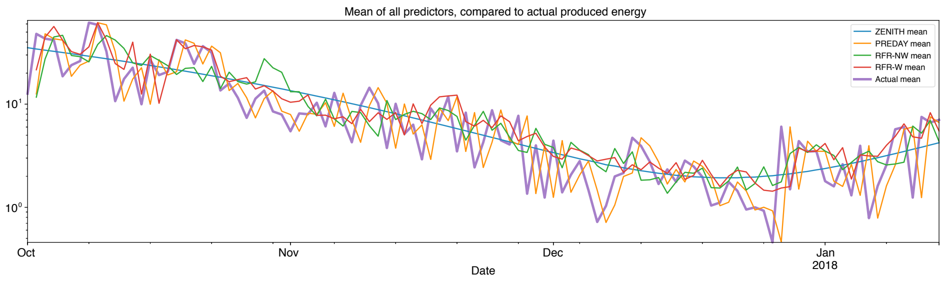

Figure E.34: Daily mean power predictions of all predictor models, together with the actual mean daily solar power

Table E.5: Prediction Model Summary

Model	Type	Input features	Tr. data	MAE*
ZENITH	Physical	Zenith	None	14.675
PREDAY	Statistical	Energy produced	1 day	16.803
RFR-NW	Statistical	Sun angle	>1 day	17.933
RFR-W	Statistical	Sun pos. + weather	>1 day	**14.242**

*Daily mean absolute error (MAE), averaged over all 105 days.

The last model is *RFR-NW*, which is a random forrest regressor that only takes the sun's position into account. It was trained similar as the *RFR-NW* predictor, but with the weather data removed.

For a comparison of the prediction models, Fig. E.34 shows the predicted mean predicted solar power for each day given by each of the predictors, in addition to the actual power produced by the solar panel. The harvested energy is gradually declining, reaching the lowest point at the winter solstice in late December, i.e., the point in time where the sun's daily maximum elevation is at it's lowest point. It also displays a high variance throughout the period due to volatile changes in weather. We observe that the different predictors follow the same trend, but that which predictor is closest to the actual value varies from day to day. We can also see that the three statistical models don't produce any value for the first day, due to the lack of training data.

Our observations are based on data from 105 days, starting at October 2, 2017 and ending at January 14, 2018. Since we are especially interested in the startup of the system once it is deployed, we defined seven overlapping periods of 21 days within those 105 days. To ensure causality, the prediction models were calculated independently for these periods, i.e., without knowledge of days outside of that period. Thus, this corresponds to seven independent deployments. Figure E.35 shows the predictions for periods 1 and 5.

As metric for the accuracy for the daily prediction, we calculate the mean absolute error (MAE) of each prediction model compared to the actual measured power for each day. Table E.5 provides an overview of all prediction models. In its last column, it shows the average over all daily MAEs as an indication for the overall performance of a predictor. The numbers imply that the RFR-W model (machine

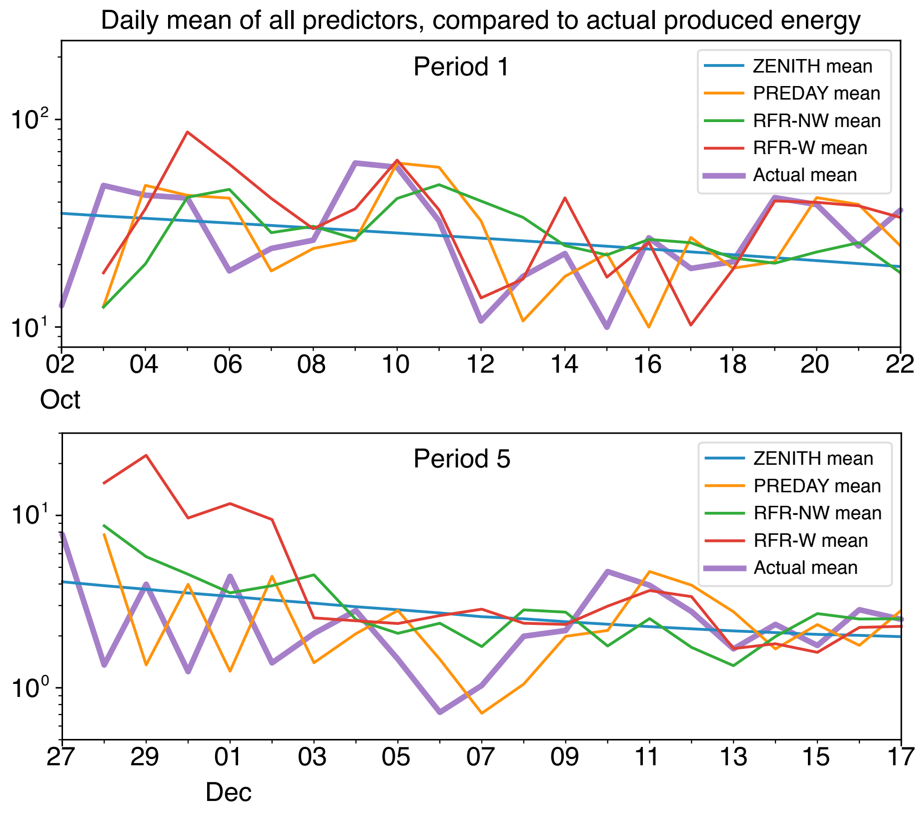

Figure E.35: Mean of predicted vs actual produced energy, for set 1 and 5

learning with weather forecast) is the best overall predictor, with regard to the entire 105-day period. During the system startup, however, the exemplified periods in Fig. E.35 indicate that other models perform better during system startup, which we will address in the next sections.

E.5 Autonomous Predictive Model Selection

The different characteristics of the models above, especially during the first days after deployment, motivate our search for a good prediction model selector. The intuition is that a device manager is equipped with a set of prediction models of different nature, and that an autonomous selection algorithm evaluates the models and selects the best one for energy prediction and planning. This selection can in principle be a machine learning task as well. However, in our case this is not a feasible solution, since we want to address the bootstrapping problem and support the system especially during the phase where no or little training data is available. Instead, we examine a selector that does not require any training, since it is solely based on the past performances of the predictors. The performance is measured with the mean average error as introduced in the previous section. Intuitively, the predictor with the lowest MAE is the best one and should be selected.

The question is how much past performances should count compared to more recent ones. To investigate this, we calculate the exponential weighted moving average (EWMA) of the MAE, using the weights $w_i = (1-\alpha)^i$, which gives:

$$y_t = \frac{x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + ... + (1-\alpha)^t x_0}{1 + (1-\alpha) + (1-\alpha)^2 + ... + (1-\alpha)^t} \tag{E.14}$$

Where α is set as:

$$\alpha = \frac{2}{s+1}, \text{ for span } s \geq 1 \tag{E.15}$$

The EWMA smoothes the data, and ensures that the most recent prediction is weighted more than the predictions in the past. The decaying factor, set as a span that corresponds to an N-day EW moving average, decides how far in the past the selection algorithm should look. When setting the decaying factor for the EWMA, there are several factors that must be considered. If the time span is too short, the selection will pick up any fluctuations in the weather and thus change predictor too rapidly to account for trends in the long-term weather pattern On the other hand, if the span is too long, the selection algorithm will need longer time to adapt to changes.

To investigate this tradeoff, we implemented different selection algorithms that used decaying factors with span from 1 to 10 days. They are listed as *S-EWMA-x* in Table E.6. The x corresponds to the decaying factor, in number of days. Thus, the S-EWMA-1 corresponds to the arithmetic mean, while S-EWMA-10 smoothes the MAE over 10 days.

For each predictor in each period, we calculated the mean of the daily MAE for all predictors and counted the number of times they were the most accurate, i.e., having the lowest MAE for that day. The result can be seen in Table E.6. For comparison, we also defined additional selectors:

- For each predictor from Table E.5, we implemented a selector that always chooses this predictor, listed as S-*predictor* in Table E.6.

- *S-ORACLE* always chooses the best prediction model in retrospect. Though this selection cannot be implemented in practice, it serves as a best-case performance boundary for our selection algorithm.

To explain the selection mechanism, we look at Figure E.36. We see the calculated EWMA of the MAE for all the predictors, smoothed over a span of 5 days, for the first and fifth period in our test set. The graphs represent the MAE for each predictor, i.e., the lower the graph, the higher accuracy of the predictor. This means that the predictor can be ranked accordingly, which results in a diagram like the one shown in Figure E.37. This rank is used to select a predictor for the next day, where the selector simply picks the predictor that at any given time has the lowest MAE.

Table E.6: Performance of Prediction Selectors (from 7 overlapping periods, total of 147 days)

	Mean MAE for individual 21-day periods							Overall results for all periods		
Selector	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period 7	Correct Days	Percent	Mean MAE
S-EWMA-1	36.279	24.144	14.418	12.473	5.503	4.365	7.316	55	37.41 %	14.928
S-EWMA-2	34.571	23.736	14.178	10.654	5.110	4.597	7.951	63	42.86 %	14.399
S-EWMA-3	34.641	23.333	14.031	10.424	5.194	4.597	7.832	66	44.90 %	14.293
S-EWMA-4	36.577	22.993	14.121	10.424	5.351	4.490	7.624	68	46.26 %	14.511
S-EWMA-5	35.555	22.993	14.211	10.424	5.351	4.490	7.330	71	48.30 %	14.336
S-EWMA-7	32.483	23.308	14.382	10.424	5.334	**4.586**	7.440	71	48.30 %	13.994
S-EWMA-10	32.483	23.308	14.802	**10.424**	5.256	**4.586**	**7.048**	71	48.30 %	**13.987**
Trivial selectors that always choose the same predictors, for comparison										
S-ZENITH	34.886	23.119	15.091	**10.424**	**5.155**	4.606	7.316	**72**	**48.98** %	14.371
S-PREDAY	38.328	21.094	15.502	14.941	6.370	5.408	7.879	37	25.17 %	15.646
S-RFR-NW	42.872	25.383	13.575	15.351	6.982	6.371	8.763	12	8.16 %	17.042
S-RFR-W	**31.598**	**20.405**	**13.001**	13.832	6.665	6.127	8.007	26	17.69 %	14.234
Oracle selector that always chooses the best predictor (with future knowledge), for comparison										
S-ORACLE	25.485	16.826	11.677	9.713	4.193	3.838	4.870	**147**	**100 %**	**10.943**

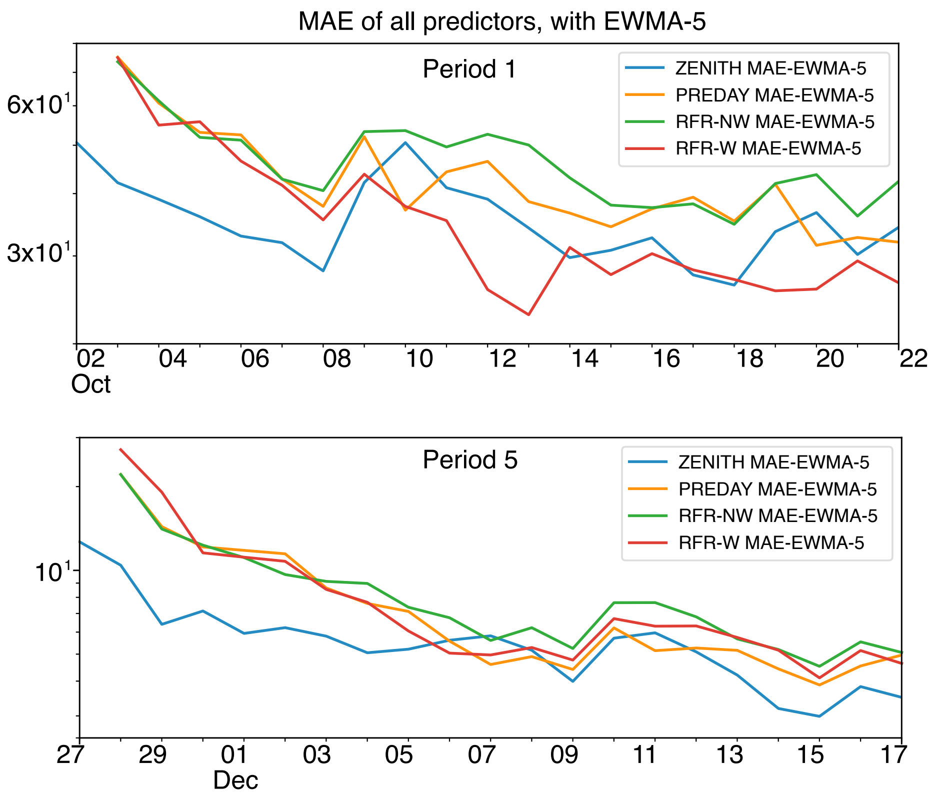

Figure E.36: MAE of predicted energy, for set 1 and 5, smoothed over the 5-day EWMA

E.6 Discussion

For each 21-day period, we calculate the mean MAE, shown in Table E.6. The last three columns show the overall results for all 7 periods, where the first two columns show how often the selectors picked the predictor that was actually best, as an absolute value and a percentage, respectively. The last column shows the mean of the mean MAE of all seven sets. Looking at the overall results, we see that S-EWMA-10 and S-EWMA-7 have the lowest MAE, with 13.987 and 13.994, respectively. They are also able to predict correctly which model that was the most accurate 48.30% of the time. S-ZENITH (always selecting the ZENITH model) is the predictor that is most often closest to the actual produced energy, having the closest prediction 48.98% of the time. However, it has a higher MAE, at 14.371. The overall best predictor is the S-RFR-W, with a MAE of 14.234, even though it was only able to come with the closest prediction in 26 out of 147 days.

When we look at Fig. E.34, we observe that the predictions of PREDAY and RFR-NW are often out of sync with the actual produced energy. This is not surprising, given the strong influence of the volatile weather on the solar energy. The result is a high MAE, when compared to both the stable ZENITH model, and

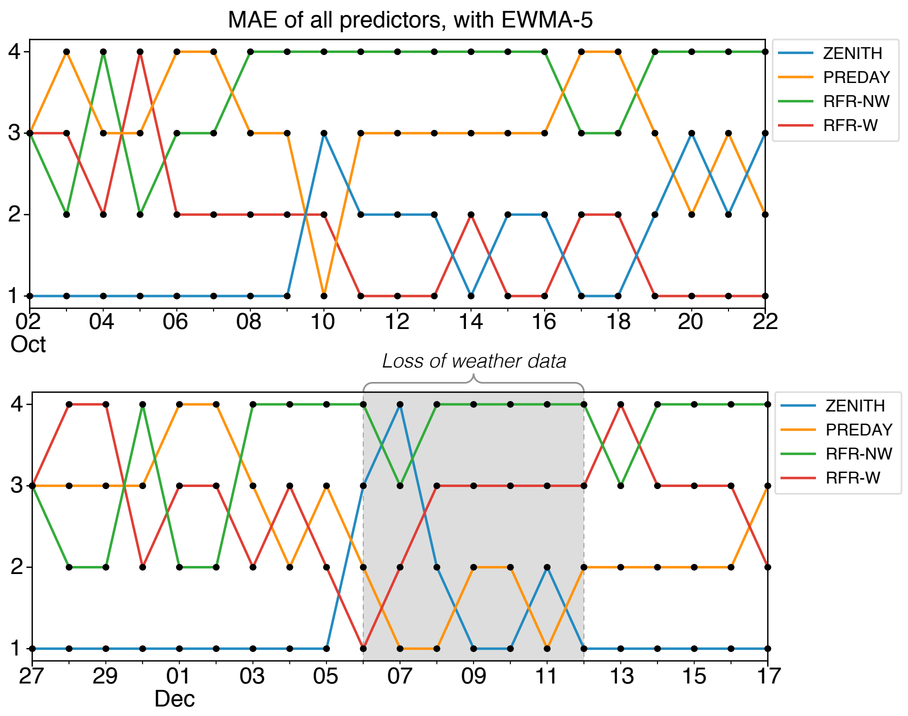

Figure E.37: All predictors, for set 1 and 5, ranked by S-EWMA-5. The predictor with rank 1 is chosen by the selection mechanism for the next day.

the RFR-W model, which is able to take the shifting weather into account.

Looking at the MAE in each of the seven periods individually, RFR-W is the best single predictor in the first three periods. However, in the last four periods, both the ZENITH model and the selection algorithms are performing better. And in the two last periods even the PREDAY model has a lower error rate than RFR-W. This indicates that even though RFR-W was able to learn fast in some periods, it had trouble generalising well over the data in other periods, further confirming the need for model selection.

To study the bootstrapping phase in each period in closer detail, we take a look at the MAE of each predictor for periods 1 and 5. For both periods, we choose to study the EWMA of the MAE with a span of 5 days. In Fig. E.36, we see that the ZENITH model has the lowest MAE for the first 8 and 9 days, respectively, and that the RFR-W model is gradually making more accurate predictions, before it becomes the preferred predictor in day 10 in both of the periods. Looking at the absolute values in Fig. E.35, it seems that this is because RFR-W is too optimistic in the first days after deployment. This trend can be seen in all periods. As time goes by, and the selector builds experience, it will put more weight to the overall trend, and thus change selected predictor less often. Overall, this benefits the ZENITH model in the beginning, while the RFR-W model is usually preferred after it has gained enough training data to give more accurate predictions. This indicates that using a set of predictors is beneficial for the overall prediction in the bootstrapping phase.

Period 5 includes a span of days, marked in Fig. E.37, where the weather forecast data was missing due to a server error, and the RFR-W model relied on old, and therefore imprecise forecasts. This can explain the sudden rise of the MAE for RFR-W, which causes the autonomous selection algorithm to change the preferred predictor, switching between the ZENITH and PREDAYmodels for some days until it stabilise on the ZENITH model. We can also see that the RFR-W improves when it get access to updated weather forecast data again. This is an indication that using a set of selectors will also provide a system with self-healing abilities.

Finally, looking at Table E.6, the data suggest that the decaying parameter should be set to keep a long span, i.e., favour stability over reactiveness. This will still allow the selector to adapt rapidly in the beginning, when the period is shorter than the decaying parameter.

E.7 Related Work

In autonomic computing, self-configuration, self-optimisation, self-healing and self-protection, are important aspects [52]. Our results indicate that these tasks can be mitigated by giving a cognitive device manager the ability to select among different prediction models. In statistics, this problem is known as *model selection* [229]. An example of model selection applied in a comparable context can be found in [230], where Doan et al. use tree-based linear regression for predicting the performance of different algorithms, in order to select the predictor that has proven most accurate on a test set. However, the choice of using machine learning for prediction inhibits the usage until a training set is available. Another example of a model selection mechanism is [231], where Dhiman et al. implements a dynamic device management controller that employs a machine learning algorithm, which is trained to select the expert policy that has the best chance to perform well for the coming period. Their experiment shows that the controller converges to select the best performing expert in the set, at a given time. This is in accordance with the observations we made in our experiment. However, again we see that the use of machine learning to train the selection controller limits the usefulness in a bootstrapping scenario.

Real-time predictive analytics, i.e., knowledge of an event before it actually occurs in order to mitigate the event or prepare for the outcome, is also an important aspect of cognitive device management. In [227], Derguech et al. propose an architecture for decision support of a energy management system in a building, using predictive analytics and open data sources. They approach this problem by implementing a selection mechanism placed inside a controller, which is responsible for choosing the data source that is best suited for prediction. Their solution is based on two steps, data management, i.e., data collection, -filtering and -storage, and data analytics, i.e., source selection and prediction. This setup is comparable to our own. However, in their paper they show that the machine-learning model they have implemented is unable to give an useful prediction until it has been fed a certain level of training data. Hence, the bootstrapping problem is not addressed here either.

An example of a specialized type of device manager for solar energy prediction can be found in [232]. Here Shresthamali et al. describe an experimental setup where an adaptive power manager uses reinforcement learning to predict the energy production of a solar energy harvesting device, consisting of a solar panel, an ideal battery and a general sensor node. In addition to historical data about harvested energy, meta-knowledge (i.e., battery level and weather data from a public weather forecast) is used to predict the future solar energy production, with the aim to achieve energy neutral operation. They address the need for sensor devices to be able to adapt to changes (i.e., season, climate and battery degradation). However, even though their machine learning algorithm predicts the energy intake with good accuracy, it neither addresses the bootstrapping problem nor how to cope with periods with missing training data. They also rely on a single machine learning model, which means the robustness of their power manager is not considered.

So far we have seen examples of device managers that incorporate model selection or other forms for cognition in their operation. However, in order to manage the device manager itself, some sort of framework that is able to handle cognitive tasks autonomously is needed. Savaglio et al. look in [233] at four relevant autonomic and cognitive architectures for management of IoT devices, of which two of them are of particular interest in a cognitive IoT setting: Inox [234] is an IoT-oriented platform that includes autonomic management, scalability, federation, adaptability and continuous optimization. It also supports virtualization, which is an important aspect of IoT device management. Device management, located in the platform layer, includes several autonomic functions, specifically self-management, self-monitoring, self-configuration, self-optimisation, self-healing, self-protection and self-adaptation. Focale [235] is an autonomic, network oriented architecture that includes a learning- and reasoning engine. It is implemented around the idea that an autonomic system must have the ability to react to events and conditions that it senses in the environment, and manage its operations accordingly. In addition, Focale includes separate models for the node itself and the environment in which it is operating. This is in contrast to many other IoT architectures and platforms which are limited to a predefined model where the functionality is known beforehand.

The notion that the environment in which the device is operating is important for autonomous and cognitive management, is also present in the management framework for smart cities that is proposed by Vlacheas et. al. in [91]. Among other things, it has the ability to select behaviour dynamically through self-management and self-configuration, based on domain knowledge and knowledge about the context of the operation itself.

E.8 Conclusion

We have proposed an evolution of device management platforms towards cognitive device management as a path towards the vision of Cognitive IoT. In particular, we presented and discussed several prediction models, and proposed an algorithm to select among those. We argue that the management of knowledge, and in particular the supervision of prediction models and the selection among them, is an important task for IoT device management. We expect in the future many more cognitive processes to guide the operation of IoT applications; energy management is only one of them.

In addition, we looked into the specific problem of operating solar-powered nodes under volatile weather and seasonal conditions. Our experiment and discussion is based on real data gathered in realistic settings. The results show that predictive model selection is a suitable means to supervise machine learning algorithms, and that having several models of different nature, i.e., physical vs. statistical, can solve or mitigate the inherent bootstrapping problem. The autonomous selection algorithm switches between initially available, but in the long run less accurate and less adaptive physical models, and statistical models based on machine learning, that are more accurate once they have gathered enough training data. The results also revealed how the multi-model approach with selection increases system robustness and provides some degree of self-healing.

References

[12] Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. "Machine learning in wireless sensor networks: Algorithms, strategies, and applications". In: *IEEE Communications Surveys & Tutorials* vol. 16, no. 4 (2014), pp. 1996–2018.

[52] Kephart, J. O. and Chess, D. M. "The vision of autonomic computing". In: *Computer* vol. 36, no. 1 (2003), pp. 41–50.

[55] Sheth, A. "Internet of Things to smart IoT through semantic, cognitive, and perceptual computing". In: *IEEE Intelligent Systems* vol. 31, no. 2 (2016), pp. 108–112.

[61] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. "Cognitive Internet of Things: A new paradigm beyond connection". In: *IEEE Internet of Things Journal* vol. 1, no. 2 (2014), pp. 129–143.

[82] Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. "Power management in energy harvesting sensor networks". In: *ACM transactions on embedded computing systems (TECS)* vol. 6, no. 4 (2007).

[91] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, F., Poulios, G., and Demestichas, G. "Enabling smart cities through a cognitive management framework for the Internet of Things". In: *IEEE Communications Magazine* vol. 51, no. 6 (2013), pp. 102–111.

[102] Kibria, M. G. and Chong, I. "Context-awareness provisioning to support user-centric intelligence in Web of Object platform". In: *2015 international conference on information and communication technology convergence (ICTC*. IEEE. 2015, pp. 388–392.

[105] Derhamy, H., Eliasson, J., Delsing, J., and Priller, P. "A survey of commercial frameworks for the Internet of Things". In: *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)*. IEEE. 2015, pp. 1–8.

[114] Nascimento, N. M. do and Lucena, C. J. P. de. "FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things". In: *Information Sciences* vol. 378 (Feb. 2017), pp. 161–176.

[150] Ahlers, D., Driscoll, P. A., Kraemer, F. A., Anthonisen, F. V., and Krogstie, J. "A Measurement-Driven Approach to Understand Urban Greenhouse Gas Emissions in Nordic Cities". In: *NIK Norsk Informatikkonferanse* (Nov. 2016), pp. 1–12.

[159] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N., and Palma, D. "Solar energy prediction for constrained IoT nodes based on public weather forecasts". In: *Proceedings of the Seventh International Conference on the Internet of Things*. New York, New York, USA, 2017, pp. 1–8.

[173] Kansal, A., Hsu, J., Srivastava, M., and Raghunathan, V. "Harvesting aware power management for sensor networks". In: *Proceedings of the 43rd annual Design Automation Conference*. 2006, pp. 651–656.

[174] Hsu, R. C., Liu, C.-T., Wang, K.-C., and Lee, W.-M. "QoS-aware power management for energy harvesting wireless sensor network utilizing reinforcement learning". In: *Computational Science and Engineering, 2009. CSE'09. International Conference on*. Vol. 2. IEEE. 2009, pp. 537–542.

[222] Kraemer, F. A., Braten, A. E., Tamkittikhun, N., and Palma, D. "Fog Computing in Healthcare-A Review and Discussion". In: *IEEE Access* vol. 5 (Jan. 2017), pp. 9206–9222.

[223] Fortino, G. and Trunfio, P. *Internet of Things Based on Smart Objects*. Technology, Middleware and Applications. Apr. 2014.

[224] Hussein, M., Han, J., and Colman, A. "An Approach to Model-Based Development of Context-Aware Adaptive Systems". In: *2011 IEEE 35th Annual Computer Software and Applications Conference - COMPSAC 2011*. 2011, pp. 205–214.

[225] Vieira, R. G., Guerra, F. K. O. M. V., Vale, M. R. B. G., and Araújo, M. M. "Comparative performance analysis between static solar panels and single-axis tracking system on a hot climate region near to the equator". In: *Renewable and Sustainable Energy Reviews* vol. 64 (Oct. 2016), pp. 672–681.

[226] Shmueli, G. "To Explain or to Predict?" In: *Statistical Science* vol. 25, no. 3 (Aug. 2010), pp. 289–310.

[227] Derguech, W., Bruke, E., and Curry, E. "An Autonomic Approach to Real-Time Predictive Analytics Using Open Data and Internet of Things". In: *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*. 2014, pp. 204–211.

[228] Braten, A. E., Tamkittikhun, N., Kraemer, F. A., and Ammar, D. "Towards Cognitive Device Management: A Testbed to Explore Autonomy for Constrained IoT Devices". In: *Proceedings of the Seventh International Conference on the Internet of Things*. 2017, pp. 1–2.

[229] Greenberg, E. and Parks, R. P. "A Predictive Approach to Model Selection and Multicollinearity". In: *Journal of Applied Econometrics* vol. 12, no. 1 (1997), pp. 67–75.

[230] Doan, T. and Kalita, J. "Selecting Machine Learning Algorithms Using Regression Models". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. Oct. 2015, pp. 1498–1505.

[231] Dhiman, G. and Rosing, T. S. “Dynamic power management using machine learning”. In: *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design.* ICCAD ’06. 2006, pp. 747–754.

[232] Shresthamali, S., Kondo, M., and Nakamura, H. “Adaptive Power Management in Solar Energy Harvesting Sensor Node Using Reinforcement Learning”. In: *ACM Transactions on Embedded Computing Systems* vol. 16, no. 5s (Oct. 2017), pp. 1–21.

[233] Savaglio, C. and Fortino, G. “Autonomic and Cognitive Architectures for the Internet of Things”. In: *Internet and Distributed Computing Systems.* 2015, pp. 39–47.

[234] Clayman, S. and Galis, A. “INOX: A Managed Service Platform for Inter-Connected Smart Objects”. In: *IoTSP ’11 Proceedings of the workshop on Internet of Things and Service Platforms.* 2011, pp. 1–8.

[235] Strassner, J. “Chapter 11 - Knowledge Representation, Processing, and Governance in the FOCALE Autonomic Architecture”. In: *Autonomic Network Management Principles.* 2011, pp. 253–274.

Adaptive, Correlation-Based Training Data Selection for IoT Device Management

Anders Eivind Bråten, Frank Alexander Kraemer, David Palma

F

Abstract

Device management can enhance large-scale deployments of IoT nodes in non-stationary environments by supporting prediction and planning of their energy budget. This increases their ability for perpetual operation and is a step towards maintenance-free IoT. In this paper we consider how to accelerate the collection of relevant training data for nodes that are introduced into an existing deployment to increase the accuracy of their predictions. In particular, we investigate how nodes powered by solar energy can learn their energy intake faster and more accurately by using data from selected nodes that are working in similar conditions. We explore an architecture that utilizes different training data selection policies to manage the learning processes. For validation, we perform a case study to explore how nodes with correlated data can contribute to the learning process of other nodes. The obtained results indicate that this approach improves the accuracy of the predictions of a new node by 14 %.

F.1 Introduction

One step towards maintenance-free IoT systems is to provide devices with solar panels or other energy-harvesting power-supplies to ensure perpetual operation. Since these power sources often are stochastic in nature [167], nodes can benefit from planning their energy budget ahead [236], and hence align their power consumption with the expected incoming energy for improved overall performance. The incoming energy depends on the specific node instance, for example the type of solar panel, orientation and location, and is often non-stationary, i.e., it changes its characteristics over time. Therefore, each node requires individual adaptation, which implies to configure each device separately and at run-time [55], taking current context and previous experience into account [7, 237]. As IoT nodes are typically constrained with regard to computation power, memory and scope of data, we examine how fine-grained individual energy planning can be part of the device management for an IoT system, for instance as part of a cloud service.

The scale of IoT systems makes it unfeasible to tune the required processes manually for each device [238], which is why autonomous operation and self-adaptation are required. In [159] we have shown how the energy for a node with solar panels can be predicted using publicly available weather forecasts and relatively simple machine learning models, in an autonomous and scalable way. The prediction can be directly used by planning algorithms like the ones presented in [232, 239]. A remaining challenge is that such approaches require training data. Hence, we argue that managing the acquisition of suitable training data is an important task of device management.

In this paper, we investigate how solar-powered nodes introduced into an existing deployment can accelerate their learning. Our approach is to give them training data collected from selected nodes that have been working in similar situations, as illustrated in Fig. F.38. This figure shows the energy intake of eight solar-powered IoT nodes facing towards four different directions. Nodes facing the same direction have similar intake profiles.

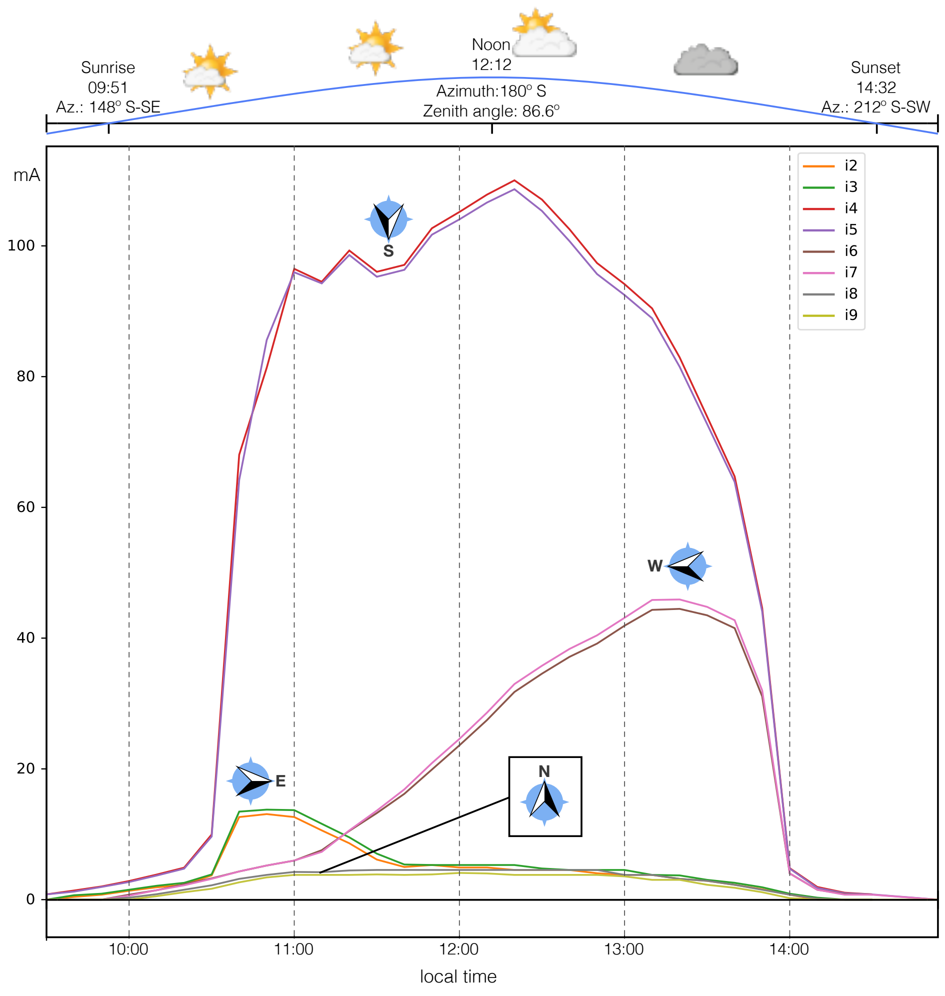

Figure F.38: Energy intake from eight solar panels located in Trondheim, Norway, on December 12th, 2018. The solar panels are pointing pairwise in four different directions.

When preparing machine learning models for the energy intake prediction for a newly deployed node, taking training data from any other node indiscriminately leads to inaccurate predictions. To temper this, we study how to identify suitable training data for a node by selecting data from nodes with correlated data. The results show that by selecting the most relevant training data based on the correlation between devices, we can train a model that decreases the error of the predictions by 14 %, compared to using data from all previously deployed nodes.

We continue with a discussion of related work in Sect. F.2 and describe the device management architecture we designed as context for the training data selection mechanism in Sect. F.3. In Sect. F.4, we present the experimental setup and elaborate the different selection policies in Sect. F.5. We then present the results and discuss them in Sect. F.6 and Sect. F.7, respectively.

F.2 Related Work

Several papers focus on autonomous, adaptive device management within the IoT. Zhang et al. [240] describe the concept of *cognitive IoT*, which aims at improving performance in the network as a whole by monitoring network conditions, analyze the collected data, make intelligent decisions based on the incurred knowledge, and perform adaptive actions. Further, they propose a network architecture that makes use of cognitive nodes that have the ability to autonomously adjust their network performance to current conditions. Sheth [55] argues that future IoT management will need to handle a large variety of devices and applications, that are subject to unplanned and unexpected events and is using predictive processing to solve their problems at the edge of the network. In this light, situational awareness is important for IoT devices, to derive value from data and learn from experience. Afzal et al. [111] advocate that cognition must be extended to incorporate IoT specific design challenges, like energy harvesting, cognitive spectrum access and mobile cloud computing technologies. Vlacheas et al. [91] address the challenges of technological heterogeneity and propose a framework that shows not only how, but also why and when devices should be connected to a network. Their framework was later implemented by Sasidharan et al. [241], who added a learning- and reasoning engine that takes contextual and situational parameters in consideration in order to improve the decision process. With their architecture they investigate connectivity issues and run performance analyses, using a network of solar energy harvesting devices deployed with central coordinators to control the management. However, the main focus in these works is either on managing the network as a whole or investigating the connectivity between the devices, while we investigate the operational part of management of the individual devices with the aim of achieving self-management.

We can also find relevant research in the domain of autonomous operation for IoT devices located in non-stationary environments. Wu et al. [61] argue that cognitive mechanisms should be used for more than network management and connectivity. Their main argument is that in an IoT framework, objects need to have the capability to reason about their physical and social environment in an independent fashion. They provide a conceptual framework based on a perception-action cycle, data analytics, knowledge discovery, decision making and service provisioning. In their framework, physical and virtual things are represented as agents that enable smart resource allocation, automatic network management and intelligent service provisioning through interaction. Foteinos et al. [7] state that support for smart and self-adaptive applications and objects are factors that need to be in place before the process of connecting heterogeneous IoT devices can be managed in a dependable, scalable and autonomous manner. To solve this, they present a cognitive management framework that adapts the configuration and behavior of devices according to the current status and context. They also show that their framework is able to improve situation awareness, reliability, and energy efficiency of IoT applications. This is in alignment with our work. However, their focus is to overcome the technological heterogeneity and complexity of the underlying networks and IoT infrastructure, while we look at the heterogeneity and complexity that is found in the physical environments of the devices. Preden et al. [237] stress the importance of situation awareness and attention when monitoring overall system performance in a dynamically changing environment, since selecting the most proper action in a given situation is highly dependent on the context of the device in question. They propose a conceptual architecture that explores these aspects in a self-aware health monitoring prototype and show that both are critical to self-awareness.

We found few works addressing the process of selecting training data for machine learning or the mechanisms that are needed to achieve this. Han et al. [242] demonstrate the challenge of estimating the disturbance covariance matrix based on an available secondary data set, when the number of secondary datasets is large and the data is heterogeneous due to non-stationary environments. This results in a combinatorial problem that is computationally expensive or even infeasible. To mitigate this, they present an algorithm based on the minimal covariance determinant that chooses training data with similar disturbance properties and discards vectors that contain possible outliers. Fraternali et al. [243] discuss the challenge of tuning individual IoT devices for perpetual performance, when there is a need to adapt to changing environmental conditions. Their approach focuses on autonomous configuration of learning algorithms on constrained devices, based on identifying the environmental context for solar-powered devices. They argue that it is unfeasible to train a different reinforcement learning policy for each individual node. Instead, they propose to use a single policy for nodes that share similar lighting conditions. In a case study, they conduct an indoor experiment that shows the performance of the devices dropping significantly when using a single policy across all devices, due to the differences in lightning conditions. This means that identify devices that experience similar conditions is an important task when managing a large number of devices. However, they only state that auto-configuration is an important aspect, but do not answer the problem of identifying devices that experience similar conditions.

F.3 An Architecture for Managing Learning and Planning Processes

To address the problem of large-scale, maintenance-free IoT systems, we explore an architecture that manages the learning and planning processes on behalf of connected devices and sensor nodes autonomously. With this approach we can empower constrained IoT devices with the ability to adapt to different situations occurring in their environment. This opens a path towards cloud-based, *cognitive* device management platforms [91], which in turn is a step towards the vision of self-managed computing systems [52] for IoT.

According to Vernon [29], autonomous operation in non-stationary environments requires that devices have the ability to see themselves in relation to their context, learn from experience, predict the outcome of future events, act to pursue goals and adapt to changes in the environment. Vernon refers to this as *artificial cognitive systems*. Figure F.39 shows an abstract model of the underlying cognitive process. It contains two cycles, a perception-action cycle and a cycle of learning, predicting and adaptation through planning. The planning activity is in the center of the perception-action cycle, binding the two cycles together. The architecture for device management presented in this paper is based on these principles. The main elements are autonomous agents hosted in the device management as part of a cloud or fog computing service. They provide constrained IoT nodes with the capability to create plans to handle future events, and thus adapt to different situations occurring in their environment [244].

To illustrate the concept, we designed an architecture that models the behavior of a cognitive device manager responsible for energy planning for solar-powered, constrained devices. Figure F.40 shows a diagram of the architecture. It is built around two components:

- The *planning manager* represents the perception-action cycle seen in Fig. F.39, centered around the planning component in the associated learn-predict-plan cycle. Its main responsibility is to keep track of the status of the devices, observe events occurring in their environment and act if there is a need to adapt by sending new configurations, for instance by adjusting the power consumption to the predicted energy intake. The actual configurations are handled by an internal configuration manager. The need to adapt may be caused by a change in predictions, or by sudden or planned events detected by the planning manager.

- The *learning manager* is responsible for handling the data, policies and actions needed for managing the learning and prediction processes. It represents the learn-predict-plan cycle in Fig. F.39 and have two sub-managers. The prediction manager is charged with the task of training the machine learning models and produces the actual predictions. It may contain a number of different models, depending on the purpose of the system. In our case, it has access to models for predicting the solar energy intake, the energy consumption and the energy buffer. The training data manager analyses training data and evaluates the accuracy of different subsets of training data, with the aim of feeding the most relevant data for training a model to the prediction model manager. This task is important to support devices operating in a non-stationary environment.

To ensure adaptation, all managers have mechanisms to trigger different decisions, shown as $T_1 ... T_5$ in Fig. F.40. For the planning manager, the trigger is a decision to send new configurations to a device when there is a need to adjust energy consumption to the anticipated energy harvest. The trigger for a prediction model manager fires if the performance of its model decreases significantly, which may indicate a change in the environment. For the training data manager, the trigger is an assessment that a different subset of training data will produce more accurate predictions for a given model. This change will cause the prediction model manager to select that training data subset next time it trains that particular model.

In the following section, we focus on the internal mechanisms of the training data manager. To this end we have performed a case study to explore how to identify suitable training data for a node by selecting data from nodes with correlated data.

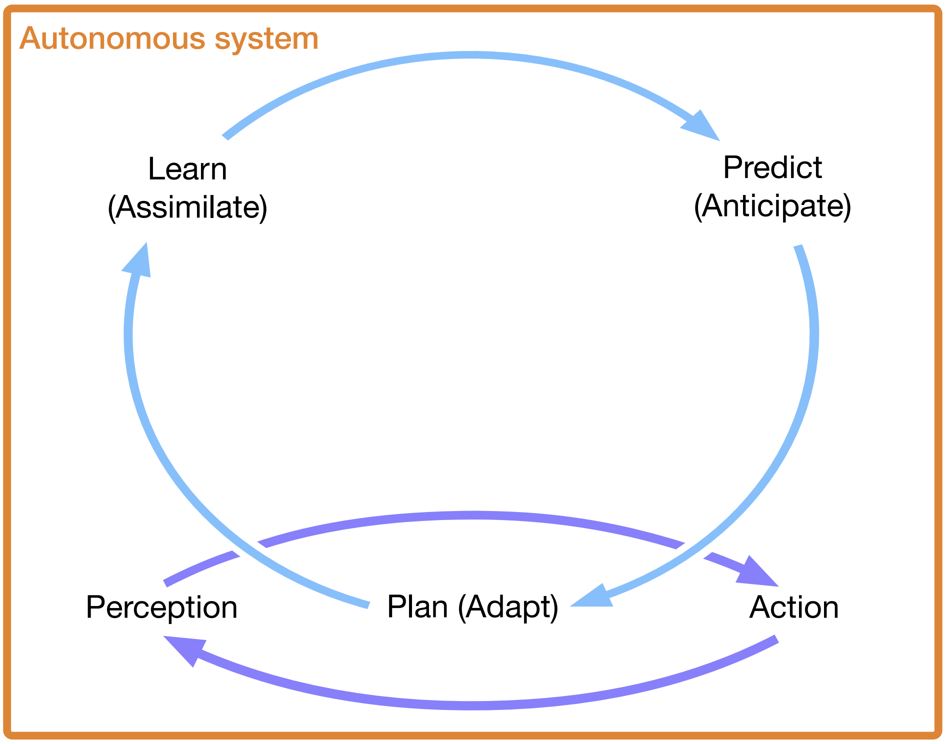

Figure F.39: Abstract model of an autonomous system, based on a perception-action cycle, which in turn is contributing to and maintained by a second cycle of learning, predicting and planning. Adapted from [29].

F.4 Experimental Setup

F.4.1 Data Collection

We collect data from a testbed consisting of eight solar panels, with two panels facing east, south, west and north, respectively. A ninth panel is mounted horizontally, i.e., in plane with the ground, for reference. The setup of these panels is shown in Fig. F.41. From the panels we collect data about the actual energy that is produced. Previously, we have identified that the position of the sun and the amount of clouds that is blocking and scattering the direct sunlight are the two most important features for predicting the energy produced by a solar panel [159, 22]. The sun position is represented by the features *zenith* and *azimuth*. Cloudiness at three different altitudes is obtained from a public weather forecast service and added to the dataset. Finally, we resample the data to ten-minute intervals.

Our observations are based on data collected between October 12th, 2018 and May 9th, 2019 [245]. To ensure that the data is useful and relevant to the experiment, we do some data cleaning. Data points where none of the panels register any energy intake are removed, since only periods with actual energy production are relevant. Snow covering the solar panels causes noise in the training data, so we also remove

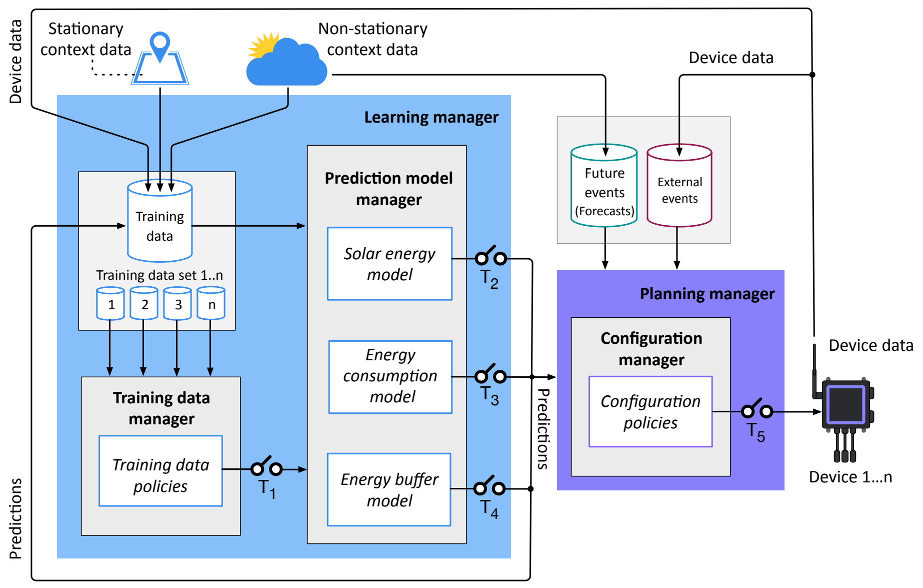

Figure F.40: Architecture of a cognitive device manager responsible for energy planning for constrained IoT devices.

those days from the dataset. In the period when the sun is lowest, cloudy weather may cause very few data points to be registered during a day. This makes it hard to find correlations. We therefore take away days where the number of collected data points is below a 150, which is a manually selected threshold. This results in a dataset that consists of a total of 168 days.

The data is then organized in nine overlapping subsets. This increases the number of observations and make it possible to find patterns and generalize the results. Each set is made up of 28 days with initial training data and 28 days where we train the model and test the algorithm. We define the start of period $\mathrm{P}_{\mathrm{n}+1}$ to be 14 days after period P_{n}.

The training data is collected from nodes I_2, I_4, I_6 and I_8, pointing east, south, west and north, respectively. In addition, we add data from the horizontal panel I_0.

At day 29, we deploy four additional devices, named I_3, I_5, I_7 and I_9, which are oriented pairwise in the same direction as I_2, I_4, I_6 and I_8, respectively. These are the nodes we want to predict the energy intake for. We then monitor the energy intake for all nodes for 28 days. This makes up the test period. Thus, each set is made up of data from 56 days.

The result can be seen in Table F.7. Columns *1st-* and *2nd batch deployment* show the dates for the first and second deployment, respectively, while column $Zenith_{noon}$ shows the zenith on the date of the second deployment. Note that high zenith values mean that the sun elevation is low.

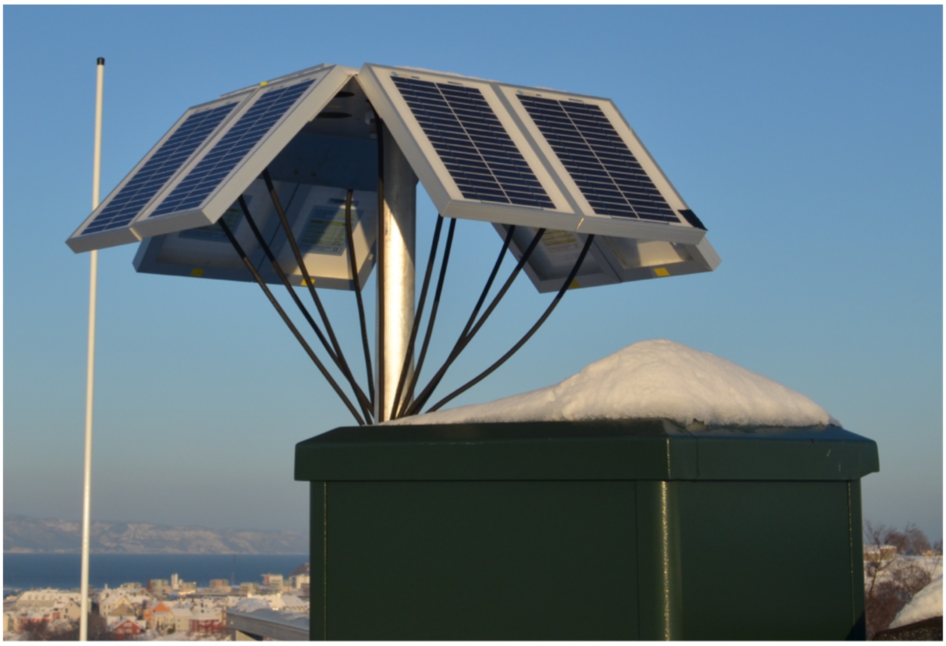
Figure F.41: Setup of the nine solar panels.

Table F.7: Overview of dates used to define periods

Period	Weeks	1st batch deployment $I_0, I_2 ... I_8$	2nd batch deployment $I_3, I_5 ... I_9$	Zenith$_{\text{noon}}$ (2nd depl. date)
P_1	Week 1-4	2018-10-12	2018-11-09	80.65°
P_2	Week 3-6	2018-10-26	2018-11-23	84.05°
P_3	Week 5-8	2018-11-09	2018-12-12	86.61°
P_4	Week 7-10	2018-11-23	2019-01-02	86.37°
P_5	Week 9-12	2018-12-12	2019-01-26	82.20°
P_6	Week 11-14	2019-01-02	2019-02-15	76.17°
P_7	Week 13-16	2019-01-26	2019-03-04	69.94°
P_8	Week 15-18	2019-02-15	2019-03-28	60.53°
P_9	Week 17-20	2019-03-04	2019-04-12	54.85°

F.4.2 Machine Learning Models

For the prediction models, we use a random forest regressor from scikit-learn [220]. We choose this model since earlier experiments has proved it to be suitable [159, 22]. In addition, the model is relatively fast to train, which is an important factor when we need to train several models for each device, possibly on an agent located

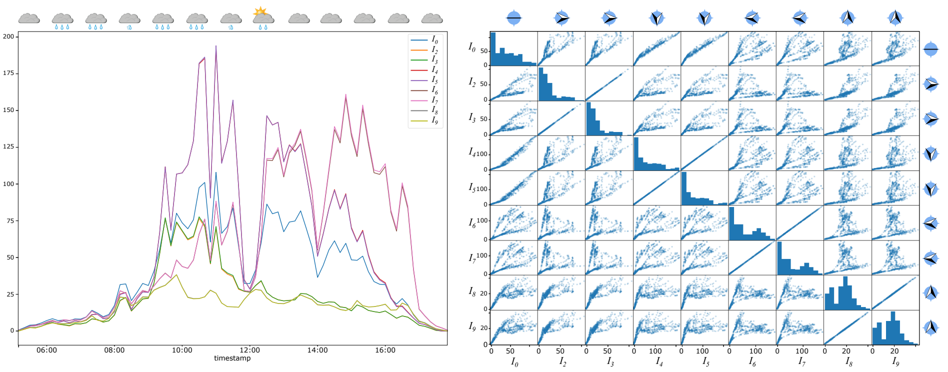

Figure F.42: Weather conditions, energy intake and correlation graphs of all nodes for March 28th, 2019.

close to the edge of the network.

One of the limitations of statistical learning algorithms is that they are unable to extrapolate beyond the range of data that has been used to train the models [246]. To temper this, we train the models regularly. This way, previously unseen weather conditions and seasonal changes in sun position are added to and reflected in the training data. Thus, the learning process is handled autonomously and continuously.

F.4.3 Metric for Prediction Performance

To assess the accuracy of the predictions, we need a metric suitable to compare the prediction performance in various seasons. With $a_{i,d,n}$ we denote the n-th value measured for solar energy of sensor i on day d, and with $p_{i,d,n}$ a prediction for the corresponding value. Since measurement points are equally spaced in time, we calculate the total energy collected during a day by summing over the individual measurements:

$$E_{total}(i,d) = \sum_n a_{i,d,n}$$

and likewise, for the prediction:

$$\hat{E}_{total}(i,d) = \sum_n p_{i,d,n}$$

We further calculate the exponentially weighted moving average (EWMA) of the daily measured energy:

$$s(i,d) = \begin{cases} E_{total}(i,1), & d = 1 \\ \alpha \cdot E_{total}(i,d) + (1-\alpha) \cdot s(i,d-1), & d > 1 \end{cases}$$

With $\alpha = 0.095$ we consider the average of the last 20 days. As an error metric for each day, we consider the scaled absolute difference between the total energy predicted and observed:

$$STAPE(i,d) = \frac{100}{s(i,d)} \left| E_{total}(i,d) - \hat{E}_{total}(i,d) \right|$$

We call this the scaled total absolute percentage error, STAPE. It describes the error of the total daily energy in percent, relative to the energy one expects on average at that day. For example, a STAPE of 20 % means that the prediction was 20 % off the actual value, relative to the average daily energy $s(i, d)$ that corresponds to 100 %. This scaling allows to compare prediction performances from different seasons, where the total energy varies considerably. At the same time, by using the EWMA, it prevents outliers on days with exceptionally low solar energy.

F.5 Training Data Selection Policies

In Figures F.42 and F.43 we can see the weather conditions, the energy intake of all nine nodes, and a scatter diagram showing the correlation between the energy intake of the nodes at the day when we deploy the second batch of nodes, for periods 8 and 9, respectively. From the weather symbols and energy intake pattern, we can see that on March 28th the weather was volatile with both cloudy and sunny periods, while on April 4th the conditions were stable with sunny weather.

Looking at the two correlation matrices, we see that for the day with volatile weather the correlation between the nodes has a high variance (the data-points are spread), while it has less variance on the day with stable weather (the data-points form a curved line). However, in both diagrams we see that for the nodes that are pointed pairwise in the same direction, the correlation graph can be seen as a straight line with a constant slope close to 1. This is a sign that the energy intake between these nodes has a high positive correlation both in stable and unstable conditions. Using Pearson's correlation, we can express this as a single number between 1 and -1, where 1 means the data are fully positively correlated, while -1 means the data are completely negatively correlated:

$$r(i_1, i_2, d) = \frac{\sum_n (a_{i_1,d,n} - \bar{a}_{i_1,d})(a_{i_2,d,n} - \bar{a}_{i_2,d})}{\sqrt{\sum_n (a_{i_1,d,n} - \bar{a}_{i_1,d})^2 \sum_n (a_{i_2,d,n} - \bar{a}_{i_2,d})^2}}$$

where $\bar{a}_{i,d}$ and $\bar{p}_{i,d}$ are the daily averages of the actual measured values and the predictions, respectively.

Using the correlation, we define the selection policy CORR-MOD. To provide comparison, we also define three other policies SELF-MOD, REF-MOD and ALL-MOD, that obtain training data using other methods for training data selection. In addition, we define a control algorithm CONTROL-MOD as a baseline. The total set of selection policies is then:

- The CORR-MOD policy collects data from a single, previously deployed device that displays the highest correlation with the newly deployed device on the date of deployment, starting from the date of the first deployment until the date of the second deployment. The data produced by the device itself is then collected for the second half of the period.

- The SELF-MOD policy collects data from the device itself, starting from the date of the second deployment.

- The REF-MOD policy collects data from the previously deployed reference node I_0, from the date of the first deployment until the date of the second deployment. The data produced by the device itself is then collected for the second half of the period.
- The ALL-MOD policy collects data from all previously deployed devices (I_0, I_2, I_4, I_6 and I_8), starting from the date of the first deployment. From the date of the second deployment it adds the data collected by the device itself.
- The CONTROL-MOD policy resembles SELF-MOD, except it is given data collected from the entire period, starting at the date of the first deployment. Thus, it uses future knowledge not available in a real setting.

All five policies are applied once for each of the four devices deployed in the second batch (I_3, I_5, I_7 and I_9), before we use the models for predicting future energy intake. This results in five series of predictions, for each device and each day. Each series of predictions is then assessed by the STAPE metric. This results in one measure for each selection policy, for each device, for each day. Since we want to compare the overall accuracy of the models that we train, the next step is to calculate the arithmetic mean of the STAPE for each policy, for all devices during the whole test period. Lastly, we calculate the arithmetic mean for all periods. Thus, we end up with five measures for each period, plus five measures representing all nine periods, as shown in Table F.8.

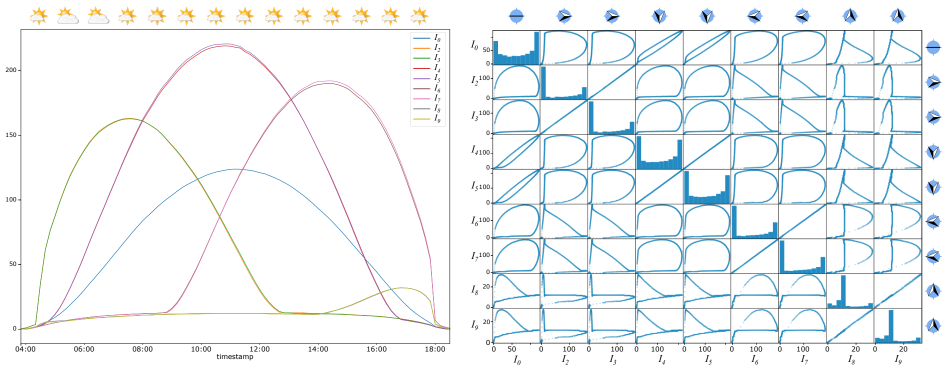

Figure F.43: Weather conditions, energy intake and correlation graphs of all nodes for April 12th, 2019

F.6 Results

Table F.7 shows the zenith at noon, on the day we deploy the second batch of devices. Figures F.42 and F.43 show the weather condition, energy intake and the correlation graphs of the deployed nodes, for two of those days. From this we can identify some of the challenges related to selecting training data for IoT devices working in a non-stationary environment. Firstly, seasonality and volatile weather conditions have a large influence on the number of data points that is collected on a

Table F.8: Performance of four RFR-models, trained on different sets of training data

	Mean STAPE for each 4 week period									Mean STAPE for all periods
Selection Policy	Week 1-4	3-6	5-8	7-10	9-12	11-14	13-16	15-18	17-20	
SELF-MOD	35.67 %	51.04 %	45.94 %	*73.98 %*	**46.65 %**	**44.70 %**	**37.51 %**	29.11 %	17.49 %	42.45 %
REF-MOD	*24.49 %*	36.03 %	50.91 %	79.61 %	56.90 %	52.36 %	48.21 %	29.79 %	16.13 %	43.83 %
CORR-MOD	**23.82 %**	**25.01 %**	**30.06 %**	**65.96 %**	*50.11 %*	*45.19 %*	*46.19 %*	**24.74 %**	*15.73 %*	**36.32 %**
ALL-MOD	28.85 %	*30.53 %*	*39.83 %*	77.81 %	61.83 %	52.97 %	50.17 %	25.95 %	**14.00 %**	*42.44 %*
CONTROL-MOD	25.44 %	24.91 %	32.96 %	66.54 %	54.11 %	48.56 %	47.75 %	26.33 %	16.06 %	38.07 %

given day. A high value for the zenith means that we have fewer hours of daylight, which again means that there is less data collected. Also, heavy clouds might block the sun completely, and thus further decrease the number of data points. Secondly, we see that the correlation of the data is more spread on days with volatile weather than on days with stable conditions. Even so, on days with stable weather and plenty of sun, the orientation of the solar panels has a great influence on how well the data correlates. These observations indicate that transferring training data indiscriminately between devices operating in a non-stationary environment should be avoided.

Table F.8 shows the mean STAPE for each policy for each 4-week period and for all periods overall. We see that the prediction accuracy given by this metric is highly dependent on the training data that is fed to the machine learning algorithm. If we look at all periods as a whole, the mean STAPE of the predictions is lowest for CORR-MOD, i.e., the model that is fed training data from a node that displays high correlation with a newly deployed node (36.32 %). This means it has the highest overall accuracy. It also closely resembles CONTROL-MOD, which is as expected since the panels used to collect training data for these two policies are pointing pairwise in the same direction. For the three other models the performance is less accurate, but on about the same level, with a STAPE of 42.45 %, 42.44 %, and 43.83 %, respectively. When we calculate the percentage decrease in STAPE of CORR-MOD (36.32 %) compared to ALL-MOD (42.44 %), the second-best overall selection policy, we find that the CORR-MOD policy improves prediction accuracy by around 14 %.

For all four models, the accuracy is worst in the 4th period (P_4). This is when the sun elevation at noon is lowest, that is, when we have the fewest hours of daylight and when the beams from the sun hit the solar panels from the lowest angle. Surprisingly, the results indicate that in periods where training data is collected while the sun is at the lowest (P_5, P_6 and P_7), the overall accuracy is best for SELF-MOD, i.e., the model that is *not* given any extra training data. However, CORR-MOD still has the next best overall accuracy in these three periods.

F.7 Discussion

In some periods the best training data selection policy is the one where the model is trained from scratch, without feeding the machine learning model any extra training data. There can be several reasons for this behavior:

1. For these specific periods, there are relatively few data points per day in the transferred training data. Thus, predictions are based on fewer observations, which in turn might lower the accuracy.
2. As seen in Table F.7, if we look at the two periods preceding the deployment of P_4, P_5 and P_6, respectively, we see that the zenith at noon is near 90°, that is, the sun is low. These are the periods used to collect extra training data. Meanwhile, for the days being predicted, the zenith at noon is rapidly decreasing, that is, the sun height is increasing. Since the zenith is the most important feature for predicting the energy intake [159], this means that the training data has little relevance for the predictions made in these periods.
3. The weather conditions in November and December in Trondheim are often volatile and can change from one hour to the next. This might introduce noise in the training sets.
4. Both the seasonality and the weather conditions cause more light to be scattered, and thus less energy hits the solar panels directly. This has a big influence on how much energy that can be harvested.

Since all these explanations are closely connected to the training data used to produce the predictions, it supports the hypothesis that selecting training data for transfer learning is an important task for an architecture that handles non-stationary environmental data.

The results are especially interesting when we look at them from an architectural point of view. Self-configuration, self-optimization, self-healing and self-protection, are important aspects of autonomic computing [52]. The proposed training data selection can support these aspects in several ways:

- By adding relevant learning data when training a model, it is possible to improve the prediction accuracy in the first period after deployment. This will improve the systems ability to perform self-optimization.
- If for some reason a node is unable to report the data that is collected, data collected from a correlated node can be used to substitute the missing data. This will improve the self-healing of the system.
- By applying methods for comparing data from two or more correlated devices, we can detect nodes that suddenly deviate from expected behavior, which might indicate that the device in question is faulty or hi-jacked. This will improve the overall ability for self-protection.

Thus, the case study also illustrates how training data selection and continuous learning is a possible method for improving the ability of constrained IoT devices to adapt to changes and act more autonomously.

F.8 Conclusion

We discussed a step towards maintenance-free IoT device management for large deployments of constrained IoT nodes working in non-stationary environments. In particular, we have investigated how solar-powered nodes that are introduced into an existing deployment can accelerate their learning by giving them training data collected from selected nodes that are working in similar situations. To illustrate the problem, we designed an architecture that models the behavior of a cognitive device manager that is responsible for energy planning for solar-powered constrained devices. For validation, we performed a case study where we studied how to identify suitable training data for a node by selecting data from nodes with correlated data. The experiment and discussion were based on real data collected under realistic conditions. Our results indicate that by using our data training selection algorithm we can train a model that decreases the error of the predictions by 14 %, compared to using data from all previously deployed nodes. This shows that managing the acquisition of suitable training data is an important task of device management when new devices are deployed and introduced into an existing system.

References

[7] Foteinos, V., Kelaidonis, D., Poulios, G., Vlacheas, P., Stavroulaki, V., and Demestichas, P. "Cognitive management for the Internet of Things: A framework for enabling autonomous applications". In: *IEEE Vehicular Technology Magazine* vol. 8, no. 4 (2013), pp. 90–99.

[22] Braten, A. E. and Kraemer, F. A. "Towards cognitive IoT: Autonomous prediction model selection for solar-powered nodes". In: *2018 IEEE international congress on internet of things (ICIOT)*. 2018, pp. 118–125.

[29] Vernon, D. *Artificial Cognitive Systems: A Primer.* 2014.

[52] Kephart, J. O. and Chess, D. M. "The vision of autonomic computing". In: *Computer* vol. 36, no. 1 (2003), pp. 41–50.

[55] Sheth, A. "Internet of Things to smart IoT through semantic, cognitive, and perceptual computing". In: *IEEE Intelligent Systems* vol. 31, no. 2 (2016), pp. 108–112.

[61] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. "Cognitive Internet of Things: A new paradigm beyond connection". In: *IEEE Internet of Things Journal* vol. 1, no. 2 (2014), pp. 129–143.

[91] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, F., Poulios, G., and Demestichas, G. "Enabling smart cities through a cognitive management framework for the Internet of Things". In: *IEEE Communications Magazine* vol. 51, no. 6 (2013), pp. 102–111.

[111] Afzal, A., Zaidi, S. A. R., Shakir, M. Z., Imran, M. A., Ghogho, M., Vasilakos, A. V., McLernon, D. C., and Qaraqe, K. "The cognitive Internet of Things: A unified perspective". In: *Mobile Networks and Applications* vol. 20, no. 1 (Feb. 2015), pp. 72–85.

[159] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N., and Palma, D. "Solar energy prediction for constrained IoT nodes based on public weather forecasts". In: *Proceedings of the Seventh International Conference on the Internet of Things.* New York, New York, USA, 2017, pp. 1–8.

[167] Moser, C., Thiele, L., Brunelli, D., and Benini, L. "Adaptive power management for environmentally powered systems". In: *IEEE Transactions on Computers* vol. 59, no. 4 (2010), pp. 478–491.

[220] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* vol. 12 (2011), pp. 2825–2830.

[232] Shresthamali, S., Kondo, M., and Nakamura, H. "Adaptive Power Management in Solar Energy Harvesting Sensor Node Using Reinforcement Learning". In: *ACM Transactions on Embedded Computing Systems* vol. 16, no. 5s (Oct. 2017), pp. 1–21.

[236] Buchli, B., Sutton, F., Beutel, J., and Thiele, L. "Dynamic Power Management for Long-term Energy Neutral Operation of Solar Energy Harvesting Systems". In: *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 2014, pp. 31–45.

[237] Preden, J. S., Tammemae, K., Jantsch, A., Leier, M., Riid, A., and Calis, E. "The Benefits of Self-Awareness and Attention in Fog and Mist Computing". In: *Computer* vol. 48, no. 7 (2015), pp. 37–45.

[238] Chatzigiannakis, I., Hasemann, H., Karnstedt, M., Kleine, O., Kroller, A., Leggieri, M., Pfisterer, D., Romer, K., and Truong, C. "True self-configuration for the IoT". In: *2012 3rd International Conference on the Internet of Things (IOT)*. 2013, pp. 9–15.

[239] Murad, A., Kraemer, F. A., Bach, K., and Taylor, G. "Management of Energy-Harvesting IoT Nodes Using Deep Reinforcement Learning". In: *13th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2019)* (2019).

[240] Zhang, M., Zhao, H., Zheng, R., Wu, Q., and Wei, W. "Cognitive internet of things: concepts and application example". In: *International Journal of Computer Science Issues* vol. 9, no. 3 (2012), pp. 151–158.

[241] Sasidharan, S., Somov, A., Biswas, A., and Giaffreda, R. "Cognitive management framework for Internet of Things:—A prototype implementation". In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014, pp. 538–543.

[242] Han, S., De Maio, A., Carotenuto, V., and Huang, X. "A novel radar training data selection method based on the minimal covariance determinant criterion". In: *International Conference on Radar Systems (Radar 2017)*. 2017.

[243] Fraternali, F., Balaji, B., and Gupta, R. "Scaling Configuration of Energy Harvesting Sensors with Reinforcement Learning". In: *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*. ENSsys '18. Shenzhen, China, 2018, pp. 7–13.

[244] Sheng, Z., Wang, H., Yin, C., Hu, X., Yang, S., and Leung, V. C. M. "Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things". In: *IEEE Internet of Things Journal* vol. 2, no. 5 (2015), pp. 402–411.

[245] Braten, A. E. *Dataset for solar energy intake correlated with weather forecast data (Version v1.0.0) [Data set]*. June 2019.

[246] Bowden, G. J., Maier, H. R., and Dandy, G. C. "Real-time deployment of artificial neural network forecasting models: Understanding the range of applicability". In: *Water Resources Research* vol. 48, no. 10 (Oct. 2012), p. 480.

Autonomous IoT device management systems: Structured review and generalized cognitive model

Anders Eivind Bråten, Frank Alexander Kraemer, David Palma

G

Abstract

Research on autonomous management for large-scale deployments of constrained devices is still a maturing field in the Internet of Things (IoT). Although much research has been conducted on how to achieve autonomous management in specific cases, there is a need for literature investigating which mechanisms can achieve such behavior in a generalized way. In this review, we present a comprehensive and structured study of the mechanisms for autonomous device management of constrained IoT devices in the light of management tasks, operational environment, network topology, resource constraints, scalability and management categories. Data extracted from 32 relevant cases is first organized and analyzed according to a synthesized taxonomy of observed adaptation mechanisms, and then combined with state-of-the-art models of autonomous operations, identifying common patterns for autonomous management. Based on our findings we substantiate best practices for designing and implementing solutions around adaptation mechanisms. We then present a generalized model for autonomous device management that describes and explains the processes required for autonomous operation, unifying the insights from previous works as one cohesive archetype.

G.1 Introduction and Background

Large-scale deployments of IoT devices are held together by device management platforms. These systems aggregate data collected by the devices and also monitor and control their operation. They are critical since insufficient device management can increase the need for expensive manual interventions and cause downtimes, waste system resources or reduce the reliability and functionality of the system by not appropriately detecting and reacting to problems [3].

Due to the scale, heterogeneity and constraints inherent to IoT systems, the architecture of device management systems is itself critical, and we observe an increasing interest in the use of principles from autonomic computing for device management over the last decade. Traditional management theory defines autonomy as *"the degree to which one may make significant decisions without the consent of others"* [4]. Applied to the IoT, this emphasizes the use of autonomously acting devices. However, the actions of autonomous agents are usually guided by a strategy or objective set by a manager. Autonomous IoT device management hence implies a division of tasks, with a central manager controlling the strategic direction and policies of the system, and the devices or agents acting on their behalf, deciding how to reach their goals [5]. The autonomy of devices can help to overcome challenges with the scale and heterogeneity of the systems. Autonomy can also reduce the need for communication, which can make the system more dependable in cases of intermittent communication. On the other hand, the support by a central management can free resource-constrained devices from complex analysis tasks, provide them with context and guide their operation.

Although an increasing amount of research has been conducted on how to achieve autonomous management of IoT devices, current research on the design and implementation of these systems is chaotic and sporadic, and does not account for the variance in adaptation mechanisms found within this domain. In addition, existing solutions are often highly specialized toward solving one or two particular tasks within a single use case. Conversely, these solutions often discuss architectural challenges in general terms, and the proposed management systems are only partially implemented. The nature of the employed mechanisms that allow autonomous behavior is rarely discussed, and alternative approaches are seldom considered. In fact, we have not found any papers that study the specific mechanisms that are used to achieve this goal in a generalized way, a challenge also identified in [35]. This shows that research on autonomous management for constrained IoT devices is still a maturing field, and that there is a need for a standardized, unified view or methodology that can advance the goal of achieving management systems for IoT that require a minimum of human intervention [6].

This is the first structured review on the topic of *autonomous* IoT device management. Existing literature within device management discusses only niche topics, lacking an overarching perspective. Sinche et al. conducted a survey on IoT management [31], where they identify key requirements for IoT device management, and give an overview of management frameworks and protocols. They stress that management solutions must be able to control IoT devices efficiently with regard to constraints and complexity. They also identify that there is a strong need for a common IoT management architecture. Chowdury et al. surveyed resource management in IoT [247]. One of their contributions is the classification of three different management activities found within this domain, namely resource discovery, resource provisioning and resource scheduling. Further, they state that ensuring automatic management and handling different architectural requirements are among the biggest challenges in distributed computing in general and IoT in particular. However, neither of these studies look at the mechanisms that are needed for autonomous device management. Colakovic et al. published a comprehensive review on enabling technologies, challenges, and open research issues within IoT [15]. Although their study is too broad to investigate in detail the challenge of autonomous IoT device management, they discuss aspects of management related to monitoring, control and configuration. They found that due to the inherent complexity of IoT, autonomous adaptation to changes in the environment requires the presence of context-aware management mechanisms.

The lack of a comprehensive review on autonomous management for constrained IoT devices means that there is a need to identify 1) which problems within this domain are addressed by the current state of the art; 2) which mechanisms are most useful for solving these problems; 3) best practices to use when designing and implementing solutions around these mechanisms; and 4) a generalized model that describes and explains the processes needed for autonomous operation. To investigate the issues pertaining to autonomous IoT device management, we chose to study models of architectures presented in previous research in the field of IoT management with the goal of unifying used techniques and lay the foundation for a structured methodology for handling autonomous device management. In particular, our review contributes to the research and development of device management for IoT with the following:

- An overview of different aspects that must be considered when designing IoT device management systems.

- A taxonomy of adaptation mechanisms that are used to allow autonomous IoT device management.
- An overview of different patterns and models that are used to achieve adaptive, autonomous management in the reviewed cases and in general literature on autonomous behavioral systems, respectively.
- The five best identified practices for designing and implementing autonomous IoT device management systems.
- A generalized cognitive model for autonomous IoT device management that unifies the key requirements identified in literature with approaches found in general autonomic computing, with an emphasis on adaptive control loops.

The rest of the paper is structured as follows: In Sect. G.2 we describe the research methodology that we used for the literature review. We continue with an overview of different aspects to consider when designing and implementing IoT device management in Sect. G.3. In Sect. G.4 we describe different adaptation mechanisms that are used to solve specific IoT management problems, before we take a closer look at patterns and models used to achieve autonomous management in Sect. G.5. Afterwards, we present the five best practices for designing solutions for autonomous management for constrained IoT devices that we identified through the review process in Sect. G.6. Finally, in Sect. G.7 we present a generalized cognitive model for adaptive, autonomous management for constrained IoT devices, based on the mechanisms, patterns and models discussed earlier, followed by our conclusions.

G.2 Methodology

This review generally followed the guidelines for performing systematic literature reviews in software engineering [248]. This process includes developing a review protocol, identifying and selecting primary studies based on pre-defined inclusion and exclusion criteria, and defining the data extraction and data synthesis activities.

Our search process is shown in Table G.9. First, we identified relevant papers (123) that were already in our possession through previous research. We then conducted a search in the digital libraries offered by IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink and Wiley Online. The search was conducted by putting together phrases using the terms in Table G.10. We used the same phrases for all 5 libraries. If a search returned more than 100 papers, the result was sorted by relevance and the first 100 papers were included for further assessment. This initial search resulted in 1703 unique papers. The third step was to read titles and exclude papers that were irrelevant for the study. This left us with 322 papers, of which we read abstracts to identify relevant papers. The resulting 188 papers were then browsed for relevance, which yielded 111 papers that we studied in detail using the inclusion and exclusion criteria presented next. The whole process resulted in 32 relevant papers, listed in Table G.11.

The review includes articles published between 2009 and 2019 on the topic of *Autonomous or adaptive device management for constrained IoT devices.* Articles with any of the following traits were excluded:

Table G.9: Selection process of papers included in the study

Step	Included Papers
Relevant papers from former research	123
Total papers after electronic search	2037
Inclusion after removing duplicates	1703
Inclusion based on title	322
Inclusion based on abstract	188
Inclusion based on skimming through text	111
Final inclusion, based on criteria	**32**

Table G.10: Terms used in search for relevant publications

Mechanisms				Abstraction level		Awareness level	Subject
autonomous	adaptive	organizing	energy harvesting	architecture	system	self-aware	internet of things / iot
dynamic	machine-learning	optimizing	management	framework	wireless	context aware	constrained device
intelligent	resource(-allocation)	configuring	smart	platform		situation aware	constrained sensor
cognitive	orchestration	healing		cyber physical		environment aware	constrained network

- Articles that do not provide a detailed model describing the components involved in device management.
- Articles that do not include an explicit description of adaptation mechanisms.
- Articles where there is no communication between nodes or between a sensor node and a central node.
- Articles with a dominating focus on security, robotics or autonomous vehicles.
- Articles describing systems where human intervention is a part of the device management process.
- Articles shorter than 5 pages or not subject to peer review.

If a topic was published in several journals or conferences by the same authors, we selected the version that contained the most detailed description of the underlying model.

All papers included after browsing through them were subject to a data extraction process. From the 111 relevant papers we first documented the main author, publisher, year published and the search term that was used for identification. We then carefully read each paper and recorded the main management problem being addressed; if the authors addressed challenges related to *resource constraints*, *scalability and technical heterogeneity*; a classification of the *environment*, the *network topology* and the *management task*; and finally a short description of the *reasoning, learning and planning mechanism* employed to solve the problem. For the 32 papers included in the review, we also classified the architectural models according to their detail level. We synthesized the data into Table G.11. This process was done incrementally, as some patterns emerged during the extraction phase.

We identified one threat to construction validity. Initially, we planned to conduct a snowball search strategy to identify relevant publications that were not caught in the manual or automated search. However, initial rounds using this strategy failed to show any papers from the selected publishers that were not already identified. Based on this we assumed that a sufficient sample of papers was already available, and therefore elected to cancel this strategy.

We did not identify any threats to internal or external validity. Any causal relationships involved in the study are discussed in an open-ended manner. We do emphasize though that the study can only be generalized within the domain that is defined through the inclusion criteria. For the assessment of conclusion validity, the chosen methodology helps to ensure that the collection procedure is repeatable. Regardless, there is a risk that relevant papers were overlooked when browsing through the titles or abstracts, since the selection is partly based on subjective reading. That said, for both identified threats we surmise that the sample of selected papers is large enough to capture the main patterns within the studied domain.

G.3 Overview of IoT Device Management

Table G.11 summarizes the reviewed use cases [10-41] and lists the different aspects to contemplate when designing IoT device management systems. We observe that autonomous management of constrained IoT devices is a composite problem related to the context in which the devices operate, the device topology, available resources, the scale of the deployment and the problem that the system solves, as stipulated in [23]. In the following, we will introduce and explain these aspects in detail. The synthesized data will then aid our analysis and guide the discussion.

G.3.1 Operational Context and Environment Type

All papers included in the final review cope with problems related to management of devices operating in settings where conditions change over time. Local conditions can vary considerably between individual devices within the same network, too. Device management therefore address operation in a context that is dynamic in temporal and spatial dimensions [20]. This means that it becomes complicated to plan proper corrective actions to a previously unseen event, since the same corrective action applied to two different devices can have different outcomes [21]. A high variance in environmental conditions thus implies that it is necessary to *individually* manage each device, to allow for operational adaptation in accordance to the varying conditions that each device experiences. We observed two types of dynamic temporal environments:

- **Stationary:** In a stationary environment, the variance is within a known distribution, that is, the changes in variation can usually be predicted stochastically. An example of device management in a stationary environment can be seen in [251], where Sahni et al. demonstrate energy-aware task allocation.
- **Non-stationary:** Non-stationary environments are characterized by dynamic statistical properties, that is, unstable conditions and distributions that change over time. An example of device management in a non-stationary environment

Table G.11: Comparison of adaptive management mechanisms for IoT Device Management

1	2	3	4	5	6	7	8	9	10	11	12
Environment[1]	Topology[2]	Constraints[3]	Scale[3]	Heterogeneity[3]	Management task	Category[4]	Reasoning mechanism	Learning mechanism	Planning mechanism	Detail level of model	Reference
S	D	●	○	○	Adaptive energy management	R	model-driven (linear prog.)	—	predictive controller model	high	[167]
S	C	●	●	○	Energy-aware network management	N, R	model-driven (stoch. geometry)	—	policy-based (obj. engine coord.)	low	[37]
S	C	●	●	●	Adaptive comp. offloading	N, R	semantic (knowledge-graph)	—	dependency-tree (directed graph)	low	[249]
S	C	●	●	●	Context-aware self-management	A, R	semantic (rule-based)	—	policy-based inference	high	[250]
S	C	●	●	●	Energy-aware task allocation	R	model-driven (nonlinear algorithm)	—	model-driven task allocation matrix	low	[251]
S	S	●	●	○	Adaptive config. management	N	semantic (rule-based algorithm)	—	inference-based algorithm	low	[252]
S	S	◐	●	●	Adaptive access control	N	semantic (ontology-based)	semantic (new rules)	event-based inference engine	high	[253]
S	S	◐	○	●	Dynamically change exec. environment	A	semantic (ontology-based context search)	semantic (stored context info.)	query built from context search	low	[254]
S	S	●	◐	●	Context-aware QoS-management	A	model-driven (stoch. model checker)	semantic (update context model	goal-directed MAPE-K control loop	high	[255]
S	S	●	●	○	Autonomous policy determination	A, R	data-driven (ML classifier)	semantic & data-driven (text-mining & SVM)	event-triggered action plan	high	[256]
N	C	●	◐	○	Dynamic energy balancing	R	model- & data-driven (game theory + RL)	data-driven (RL)	reward-based utility function	high	[20]
N	C	●	◐	●	Context-aware self-management	N	semantic (ontology-based)	—	event-based inference engine	high	[257]
N	C	●	●	●	Network lifetime optimization	N, A	semantic & data-driven (CBR + RL)	data-driven (RL)	goal-directed action plan	high	[258]
N	C	●	●	●	Context-aware self-management	N, R	semantic (ontology/context-based)	model-driven (game theor. learning)	decision based on game theory	low	[61]
N	C	●	●	●	Adaptive device orchestration	N, R	semantic (ontology-based)	semantic (learned facts)	goal-directed action plan	high	[259]
N	C	●	●	●	Context-aware self-management	N, R	model-driven (game theory)	model-driven (weighted obs.+ univ. approx.)	maximized utility function	low	[260]
N	C	●	◐	●	Adapt to recognized activity	A	model-driven (fuzzy logic)	data-driven (machine-learning)	goal-directed action plan	high	[237]
N	C	●	◐	○	Resource-aware data collection	A, R	data-driven (RL)	data-driven (RL)	utility look-up table	high	[261]
N	C	●	●	●	Adaptive comp. offloading	R	model-driven (markov dec. proc.)	data-driven (deep Q-learning)	learned policy	low	[262]
N	C	●	○	○	Energy-aware self-management	R	model-driven (modal logics)	—	goal-directed action plan	high	[263]
N	C	●	●	◐	Energy-aware QoS-management	R	semantic & data-driven (dynamic prog. + SVM class.)	data-driven (machine-learning)	QoS- and policy-based service provisioning	low	[264]
N	C	◐	●	●	Context-aware self-management	R	semantic (rule-based inf. engine)	data-driven (machine learning)	event-triggered action plan	high	[241]
N	S	●	●	●	Energy-aware self-management	A, R	semantic (rule-based inf. engine)	data-driven (RFR)	policy-based action plan	high	[23]
N	S	●	●	●	Autonomous network resource discovery	N	model-driven (MAPE-K)	semantic (stored context info.)	policy-based MAPE-K control-loop	high	[265]
N	S	◐	●	●	Adaptive config. management	A, R	semantic (pattern recogn.)	data-driven (machine-learning)	policy-based action plan	high	[7]
N	S	◐	●	●	Adaptive config. management	A, R	data-driven (RL)	data-driven (RL w/back-propg.)	goal-directed control loop)	high	[114]
N	S	●	●	○	Adaptive appl. management	A, R	semantic (semantic modeling)	data-driven (deep-learning+RL)	goal-directed action plan	high	[266]
N	S	●	○	○	Energy-aware task allocation	R	model-driven (smart persistence)	—	semantic task allocation algorithm	high	[193]
N	S	●	○	●	Autonomous service discovery	R	model-driven (prob. reasoning)	model-driven (prob. distr. learning)	event-based service provisioning	high	[267]
N	S	●	○	○	Adaptive energy management	R	data-driven (RL)	data-driven (RL w/back-propg.)	reward-based utility function	high	[232]
N	S	●	●	○	Energy-aware self-management	R	data-driven (ML classifier)	data-driven (RFR+ANN)	policy-based utility function	high	[159]
N	S	◐	●	●	Context-aware self-management	R	semantic (context ontology)	semantic (stored episodes)	goal-based action plan	high	[268]

1) Environment: S. . . stationary, N. . . non-stationary
2) Topology: D. . . distributed, C. . . clustered, S. . . star
3) addresses the concern ●. . . directly, ◐. . . indirectly, ○. . . not at all
4) Category: N. . . network A. . . application, R. . . resource

Abbreviations:
machine learning (ML), case-based reasoning (CBR), reinforcement learning (RL), support vector machine (SVM), random forrest regressor (RFR), artificial neural network (ANN)

can be seen in [262], where Alam et al. demonstrate adaptive computational offloading.

As we will show in Sect. G.6, this difference plays a significant role for the employed adaptation mechanism. We therefore classified the reviewed cases according to the type of environment in which they operate, as shown in column 1 of Table G.11.

G.3.2 System Topology

The system topology describes how the devices are connected. This is a key design decision that influences the organization of the management processes, which are organized in three different ways, as shown in column 2 of Table G.11.

- **Fully distributed topology (D):** In a distributed topology, each device is responsible for all actions needed to operate and adapt, including storing knowledge and initiating learning processes. Even though a central node is usually present, its only task is to collect data that is sensed by the devices. This means it has no power to manage the operation of the device. In the reviewed papers, we found only one case with a distributed topology [167].
- **Clustered topology (C):** In a clustered topology, two or more parent nodes share the responsibility for storing data and managing processes on behalf of separate subgroups of devices. Each parent node sends instructions to the devices belonging to designated subgroups. The devices may have some autonomous responsibilities, such as basic reasoning, but learning is usually offloaded to the parent node. Often, a parent node can share knowledge with other parent nodes. Communication between devices within each own subgroup is allowed, but uncommon. We found this topology in 16 cases.
- **Star topology (S):** In a star topology, a single central node stores all data and manages all processes for all devices. The central node sends instructions to the managed devices, which are responsible for receiving and storing instructions, sensing and sending data, and executing actions. There is usually no direct communication between the managed devices. We found 15 cases with a star topology.

G.3.3 Resource Constraints

Column 3 of Table G.11 indicates if the cases address resource constraints directly or indirectly. Despite advances in capabilities of IoT devices, they are constrained in terms of available energy, memory and processing capability. In addition, they usually have limited access to contextual information. These constraints make it hard for the devices to solve their own problems, since they lack resources needed to analyze the current situation and predict future events that might influence their operation. To ensure that the devices are able to operate at their optimum and plan corrective and adaptive actions, such tasks are therefore often moved to nodes with better access to resources [269]. The problem of resource constraints are thus often tied to the system topology.

G.3.4 Scale and Technical Heterogeneity

Traditionally, device management for wireless IoT nodes has been done manually, where the devices have been configured and updated individually or in bulk, either on-site or over a communication channel. However, this method does not work well in large-scale deployments characterized by many devices and high heterogeneity, which means that the devices and the networks connecting them vary in form, function and functionality [270]. Maintenance throughout the full device lifecycle (planning, configuring, deploying, operating, repairing, and recycling) is a major challenge. Any architecture or framework that supports large-scale device management must therefore be able to operate autonomously with a minimum of human intervention [15]. We indicate to which degree the papers address challenges related to resource constraints, scale and heterogeneity in columns 3 to 5 of Table G.11.

G.3.5 Management Tasks

All the systems in the reviewed cases are directed toward management of constrained IoT devices. Within this domain we find a broad spectrum of operations. Usually, the main problem that is addressed in an article maps to a specific management task. We describe the main management task performed by each reviewed system in column 6 of Table G.11. Each of these specific tasks can be further mapped to three distinct categories. We chose to categorize device management in the reviewed cases as network, application or resource management, or a combination of these. This categorization is in contrast to Gurgen et al. [271], who divide management of networked sensing devices in network-, system- and application management. This is due to the fact that we did not find any cases that focused on system management, while many cases went beyond application management and focused on how to manage the resources that are available for the devices directly. The category that the main management task belongs to is shown in column 7 of Table G.11.

- **Network management** is concerned with how to initiate, monitor and maintain the infrastructure of a wireless sensor network, to ensure that the devices are connected and able to send the collected data. Some typical management tasks include discovery, that is, registering devices when connected to a wireless sensor network for the first time [265], ensuring they maintain a stable connection [272], and reconnecting them when they drop out of the network or move between base stations [261]. In many networks the connections between the devices, or even the topology of the network itself, change over time. Such networks can be regarded as a dynamic environment with non-stationary properties. Research related to this problem area is often referred to as *Cognitive IoT* (CIoT). The main idea of CIoT is that interconnected devices are able to analyze their context, learn from experience and develop hypotheses based on their knowledge base with a minimum of human intervention [266]. Typical management tasks within CIoT are aimed at maximizing network performance by analyzing current network conditions, and then deciding and executing adaptive actions [240].

- **Application management** runs processes to configure, monitor and maintain the applications that run on the devices. These are typically extrovert processes, that is, they focus on the purpose of the system that is managed. The main input for application management is the sensor data acquired by the IoT devices. Typical management tasks are off-loading, distributed computing and data collection, in addition to high-level tasks like processing and analyzing environmental data sensed by the devices. Two examples of application management processes are activity adaptation [237] and action recommendation [256].

- **Resource management,** in contrast to application management is an introvert process, as it typically looks at internal processes related to the maintenance and optimization of the operation itself. The main input for resource management is operational data about the devices and their current status. The focus is typically on low-level tasks like ensuring an acceptable quality of service, managing the energy consumption and scheduling sensing cycles. In a dynamic environment, resource management needs to be context-aware, to accommodate for variations in the environment [68]. Examples of resource management are resource-aware data collection [261] and energy balancing [20].

G.4 Adaptation Mechanisms

Autonomy is a complex behavior characterized by the capacity an agent has to achieve a goal while adapting to changes in the environment without human intervention [43]. Further, Sifakis et al. [43] list five complementary aspects required to achieve full autonomy: 1) *perception*, or interpretation of stimuli from the environment; 2) *reflection*, that is, building a model of the environmental context; 3) *goal management*, i.e., choosing the best among possible goals given the environmental model; 4) *planning*, or deciding which actions to take to achieve the chosen goal, and 5) *self-adaptation*, i.e., to adjust the autonomous behavior through learning and reasoning.

Self-adaptation and self-management are core concepts in autonomous systems [32]. According to Kephart et al., the goal of self-management is to free system administrators from the tasks of system operation and maintenance and provide systems with the ability to configure, optimize, heal and protect themselves [52]. However, for systems that operate under conditions that vary over time, this means that the devices have to adapt, i.e., adjust operation in accordance with the current situation. Sheth et al. argue that adaptive decision mechanisms under such conditions require situation awareness, that is intelligent mechanisms that can convert raw data into something that is contextual meaningful [55].

Vernon goes further in [29], discussing the concept of self-awareness, i.e., the extent to which a system can reflect about itself. He states that self-awareness can be seen as a device's ability to see itself in relation to its context, learn from experience, predict the outcome of future events and act to pursue goals. Preden et al. support this view in [237], claiming that for devices operating in a dynamically changing environment, self-awareness is needed for devices to understand their own state in relation to the environmental conditions that influence their operation. Thus, a device manager can achieve autonomous self-management by combining situation-awareness with adaptation mechanisms to dynamically select its behavior, taking previous experience, contextual parameters, internal status and designated policies into account, as discussed by Foteinos et al. [7] and Sezer et al. [273].

G.4.1 A Taxonomy of Observed Adaptation Mechanisms

We see the considerations above confirmed in the reviewed cases, and observe a general pattern with autonomous device management based on adaptation mechanisms that analyze input data, reason about the current situation and produce some output data that result in a corrective action or plan, when needed. Many models include learning mechanisms as well, to expand the knowledge base of the system. These mechanisms are often encapsulated in separate modules, to reduce complexity and separate concerns. We will discuss this aspect further in Sect. G.6. In particular, we identified three distinct types of adaptation mechanisms, indicated in columns 8, 9 and 10 of Table G.11:

- **Reasoning mechanisms** analyze sensed events and device states, reflect upon the current situation and control the internal data flow of the manager node. They also decide if a perceived situation requires adaptation.
- **Planning mechanisms** produce corrective actions or action plans, send instructions to the managed devices, and make sure the manager and the devices are in sync.
- **Learning mechanisms** make sure that the knowledge base is up to date, that it reflects the state of the devices in the context of the environment in which they operate.

In the following, we will take a closer look at reasoning mechanisms, since these are paramount for autonomous behavior. Perera et al. [36] classify reasoning mechanisms into six categories: supervised learning, unsupervised learning, rules, fuzzy logic, ontological reasoning and probabilistic reasoning. However, this classification neither differs between reasoning and learning mechanisms, nor covers all the different adaptation mechanisms we found in the reviewed papers. We therefore chose to categorize the different reasoning and learning mechanisms according to the underlying principle that the mechanisms use to infer an understanding of the situation, as shown in Fig. G.44.

- **Model-driven mechanisms** capture knowledge and derive decisions through representation and rules that are declared explicitly. This means they are based on logics that are inherent to the model, which also implies that all variables that are part of the reasoning must be declared within the model itself. Thus, the output is a result of a logical analysis of a context change. Linear and nonlinear programming, probabilistic analysis and fuzzy logic can usually be placed in this category.

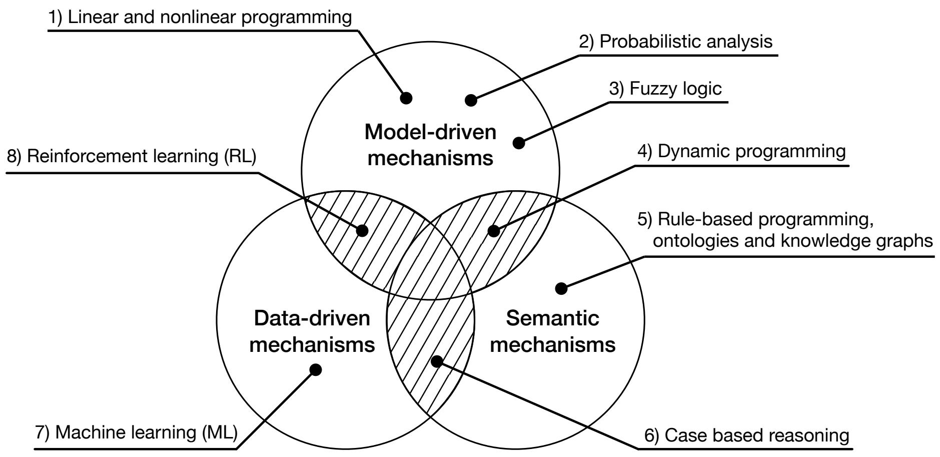
1) Linear and nonlinear programming
2) Probabilistic analysis
3) Fuzzy logic
Model-driven mechanisms
8) Reinforcement learning (RL)
4) Dynamic programming
5) Rule-based programming, ontologies and knowledge graphs
Data-driven mechanisms
Semantic mechanisms
7) Machine learning (ML)
6) Case based reasoning

Figure G.44: Reasoning and learning mechanisms used in device management

- **Semantic mechanisms** analyze structures where meaning is associated with the data. This allows data to be interpreted in context, regardless of differences in syntax or structure [55]. Knowledge is usually stored in a knowledge base that holds facts about the world, often in the form of ontologies, knowledge graphs or episodes. The semantic reasoning mechanism is an inference engine that applies logical rules to the knowledge base to deduce new information. Its output is thus based on inferring a deeper meaning or finding similarities to the input variable. Case-based reasoning, ontology-based inferring and rule-based programming are often placed in this category.
- **Data-driven mechanisms** are based on some statistical analyses that attempt to identify patterns in the data. This can be historical data collected by the devices themselves, or it can be other relevant contextual data collected from external sources. The output is statistically inferred from the data itself. Most machine-learning methods can be placed in this category.

We will now take a closer look at the particular mechanisms used in literature with examples for their usage.

G.4.1.1 Linear and Nonlinear Programming

Linear and nonlinear programming are mathematical models often used for optimization purposes. The main difference is that in nonlinear programming, a change in input is not necessarily proportional to the change in output. Often, nonlinear problems are approximated using linear equations and algorithms to reduce complexity. Linear and nonlinear programming rely on all decisions being in place up front. As a consequence, this family of mechanisms is rarely used to solve complex problems where the answer is inferred from the data, like cluster analysis. A typical example of linear programming is found in [167] where Moser et al. use multiparametric programming for power management of solar harvesting wireless sensor nodes. They present a formal model for solving the optimization problem offline in different environmental conditions and system states, maximizing the utility in a long-term perspective.

G.4.1.2 Probabilistic analysis, Markov-modeling and Bayesian inference

Applying probability theory is a common method for quantitative modeling and analysis of large, stochastic data sets. Probabilistic models use graphs to represent stochastic variables, where edges represent assumptions that are conditionally independent. This means they are well suited to solve problems related to joint probability distributions. Bayesian inference uses the Bayes' theorem to update the probability for a given result as more data becomes available. Thus, by applying structural learning, knowledge can be retained within the model itself. Probabilistic models of both types are often used for dynamic analysis of sequences of data. However, these mechanisms often come with a high computational cost, especially in models with a large number of parameters. Also, the process of choosing the prior distribution is time-consuming and often requires in-depth expertise of the problem to solve. This makes it difficult to apply such methods on constrained devices or in domains characterized by non-stationary properties. An example of a probability-based reasoning mechanism can be seen in [262] where stochastic randomness in available resources and numerous allocation options, in conjunction with reinforcement learning, make a Markov decision process a good fit to solve the problem of computational offloading.

G.4.1.3 Fuzzy logic

Fuzzy logic is used to model logical reasoning on sets that amount to degrees of membership, fuzzy sets. It is based on the idea that a preposition can be partly true and partly false at the same time, with a degree of truth usually defined as a real number in the interval [0,1]. This allows a continuous range of choices [274]. A central aspect of fuzzy logic is the mapping to linguistic variables, like *slow* or *tall*. In a fuzzy expert system, such linguistic variables are used to produce fuzzy rules, which are used to infer a decision from the model. New knowledge can be retained within the model itself, by manipulating fuzzy sets, or by applying new fuzzy rules semantically. Fuzzy logic is suitable for domains where the input is imprecise and the outcome is uncertain. However, fuzzy systems lack the capability of learning from experience or recognizing patterns, and extensive testing is often needed for validation and verification of fuzzy knowledge-based systems. In addition, the iterative process of defining fuzzy rules and membership functions is time consuming and requires expert knowledge [275]. Preden et al. [237] present an example of fuzzy logic, where an application first associates situation parameters like sleeping time, breathing patterns, heart rate and movement with degrees of membership in fuzzy sets before fuzzy rules estimate the quality of sleep.

G.4.1.4 Dynamic programming and recursive optimization

The main principle of dynamic programming is to break a problem down into smaller sub-problems, often to find an optimal score using recursion. Each sub-problem is then solved sequentially, and the result of each iteration is retained in a dynamic programming matrix and used as input to the remaining sub-problems. Finally, the algorithm does a traceback of the matrix to recover the structure of the optimal solution. Although dynamic programming shares properties with semantic and model-driven mechanisms, it is not particularly suited for solving causal inference since it might align unrelated sequences [276]. Thus, dynamic programming is often used as part of, or as a complement to, other adaptation techniques. An example of recurrent dynamic programming can be seen in [264], where Samie et al. use it for energy-aware QoS management of IoT devices under bandwidth, battery, and processing constraints.

G.4.1.5 Rule-based inference, ontologies and knowledge graphs

These methods are all based on semantics, associating meaning to collected data. In systems that rely on semantic adaptation mechanisms, knowledge is usually represented by defining a set of concepts and the relationship between them [257]. In this way data can be interpreted contextually, that is, detached from the syntax or structure, by using annotation techniques to infer knowledge from the interpreted data [55]. Reasoning typically attempts to derive facts that are not explicitly expressed in the ontology or knowledge graphs [277]. New knowledge obtained this way can be retained and stored as a new rule, a new ontology or as an expansion of the knowledge graph. However, since the logic is based on semantics these methods are not particularly suited to solve problems that are based on quantifiable data. An example of ontology-based reasoning can be seen in [268] where knowledge related to a situation is represented in a semantic-based context mode that contains definitions of basic concepts and relations. This provides a common vocabulary that can be used to manage and share context data among users, devices and services.

G.4.1.6 Case-based reasoning

The assumption behind case-based reasoning is that similar problems have similar solutions [57]. Knowledge is usually captured and retained as episodic data in a case base. This means it can be categorized as a hybrid mechanism, part semantic and part data-driven. To solve a problem, a reasoning mechanism typically uses the principle of analogy to retrieve and reuse episodes that match the current situation. Since case-based reasoning considers what happened rather than on how or why it happened, it is well suited for domains where the context is not explicitly defined [278]. This sets it apart from mechanisms based on semantics. We can see an example of case-based reasoning in [258], where information stored in a knowledge base is used as a case base to find an appropriate action to improve network lifetime and quality of information.

G.4.1.7 Machine learning (ML)

In machine learning, data is analyzed statistically using analytical or mathematical models that identify patterns in the data. The ML algorithms are trained using sample data, and can then be used to make predictions or decisions without explicit programming [21]. In *supervised learning* the data is labeled and an output is mapped to an input. Thus, it infers a function from labeled training data. The output can be a continuous numerical variable found through regression, or a discrete or categorical value found through classification. In *unsupervised learning* the data is unlabeled and the algorithms learn to find unknown patterns or structures in the data. It is typically used for clustering, to partition data sets into groups [273]. For both methods, learning happens when previously unseen data is added to the training data and the algorithm is retrained on the new data set. Some limitations of ML origin in its reliance on statistical data. One problem with ML is the need to train each model to fit the particular application. Also, the relationship between output and data is encoded as correlations, but causality or relationships cannot be inferred. In IoT management we can see examples of supervised learning in [256], where Megahead et al. use a support vector machine classifier for autonomous policy determination, and in [159], where Kraemer et al. use random forest regressors and artificial neural networks for energy-aware self-management of solar-powered IoT devices. We did not see a clear example of clustering using unsupervised learning in any of the reviewed cases.

G.4.1.8 Reinforcement learning

Reinforcement learning (RL) is commonly used for control optimization problems with many states and complex stochastic structures. It employs a reward function and learns through interaction of an agent with its environment, with no need for a complete control model or explicit supervision [258]. An RL agent is trained to improve a task by learning from experience, that is, interacting with that particular task in context [239]. The algorithm is trained with the goal to maximize the cumulative reward. The agent thus learns the policy that produces the highest reward while avoiding policies that produce low or negative rewards. We categorize RL as a hybrid of data-driven and model-driven mechanisms, since environments that provide rewards are often based on a mixture of explicit models and data. An example can be seen in [193], where Edalat et al. use reinforcement learning for network lifetime optimization. Challenges with RL are the design of the reward function, as this requires an in-depth knowledge of the domain and the system goals, as well as a potentially high training effort [239].

G.4.2 Handling Complexity by Combining Mechanisms

Table G.11 shows that reasoning, learning and planning mechanisms often are combined to solve a particular problem. We also see that some models employ a mix of different categories of adaptation to achieve adaptation. A synthesis of this observation is presented in Table G.12. Here we indicate the category of reasoning and learning mechanisms, grouped by the type of operational environment, for each of the 32 cases presented in Table G.11.

For systems operating in a stationary environment, we see that only 1 system uses a data-driven mechanism, most likely because behaviors can be adjusted in a deterministic manner. Also, just 5 out of 10 cases include a learning mechanism, 4 of which are purely semantic and 1 has a strong semantic component. For systems that are deployed in a non-stationary environment, we see that there is more variation in which type of reasoning mechanisms is used. Here, 7 cases used a model-driven

Table G.12: Overview of observed reasoning and learning mechanisms

	Stationary environment		Non-stationary environment	
	Reasoning	Learning	Reasoning	Learning
Model-driven	4	0	7	3
Data-driven	1	0	4	13
Semantic	5	4	8	2
Mixed	0	1	3	0
Not included	0	5	0	4
Total	10	10	22	22

approach, 4 used data-driven, and 8 used semantics. In addition, 3 cases used a mixed approach to reasoning, with data-driven reasoning being a component in all of them. 18 of 22 systems incorporate learning mechanisms, 13 of which are primarily data-driven.

G.5 Patterns and Models for Autonomous Management in IoT

We now turn our attention to the combination of the various adaptation mechanisms and how their interactions lead towards a cognitive system. The 32 reviewed cases expose a wide variety in description style and rigor. Most detail only some aspects while neglecting other components or processes that are necessary to understand the bigger picture. 19 models mainly focus on component composition and interactions, while the other 13 models focus on a description of the process. This variety makes it difficult to interpret the adaptation process as a whole in each case. We therefore extract partial models from the reviewed cases and expand them with more general literature and cognitive models.

G.5.1 Patterns Observed in Literature

The 32 reviewed architectures typically divide reasoning, learning and planning mechanisms into separate components. The descriptions in 23 papers were so detailed that we could extract the relation between these three mechanisms (marked with *high* in column 11 of Table G.11). Our study reveals the following patterns:

1. The managing process is centralized, that is, the network is organized in either a star or a cluster topology, as for instance in [237, 253, 114].

2. Sensed events are sent from the device to a reasoning mechanism for analysis, for instance [258, 255, 266].

3. Reasoning processes are initiated either from an observed event, an internal process or a prediction, for instance in [7, 23, 261].

4. Reasoning and learning mechanisms are distributed throughout the architecture, for instance in [257, 256, 263, 268].

5. Learning mechanisms are placed in conjunction with a component responsible for assessing the situation, for instance in [259, 267, 241].

6. There is a separation of concerns between two main processes, namely understanding the current situation and planning a corrective action, for instance in [20, 250, 167].

7. Task allocation, goals and policies are managed by a planning component, e.g., in [265, 193, 232, 159].

Combining multiple adaptation mechanisms is a common strategy. A reason for this is that distributed processes often complement each other, especially in systems that need to solve tasks that require understanding on a higher cognitive level, that is, self-, context- or situation-awareness [36]. In these cases we see that the interaction between the components, and the interplay between the different adaptation mechanisms, define and produce the internal cognition.

G.5.2 The MAPE-K Autonomic Control Loop

A general autonomic system architecture is based on sensors and actuators, controlled by a feedback loop [56]. A typical feedback-control-loop involves four steps: 1) Collecting and monitoring sensing and contextual data; 2) Processing and analyzing the collected data, which may trigger a need for adaptation; 3) Making a decision on what to change, based on an adaptation goal; and 4) Executing adaptation through an appropriate mechanism. The new, adapted state of the system is then returned into the feedback loop by a self-reflective mechanism that is used throughout the adaptation cycle. The accumulated data is stored for future reference in a knowledge base, and used to provide a more accurate model of past and future states, in an attempt to identify symptoms and infer trends that go into the decision planning [279].

Many autonomous and self-adaptive systems that make use of sensory input are based on the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) autonomic feedback loop [50, 49]. A model of this type of control loop is shown in Fig. G.45. This architecture allows a device to manage itself and dynamically adapt to changes based on predefined policies and objectives. Some learning is inherent in the model by retaining sensed information and saving the effect of an executed action. Few of the mechanisms identified in Sect. G.5.1 directly refer to MAPE-K, but we observe that the pattern of the four reasoning processes, i.e., monitor, analyze, plan and execute, is present in many of the reviewed papers. It is therefore natural to incorporate this pattern in a generalized model, which in IoT corresponds to a system where devices receive events through sensors or internal processes, and respond to these events through an analysis of the situation and planning of adaptive actions.

G.5.3 Cognitive Models

From Tables G.11 and G.12 we observe that the adaptation mechanisms found in purely autonomic systems tend to be preconfigured, while situation-aware systems more often are able to create their own rules through learning-by-experience.

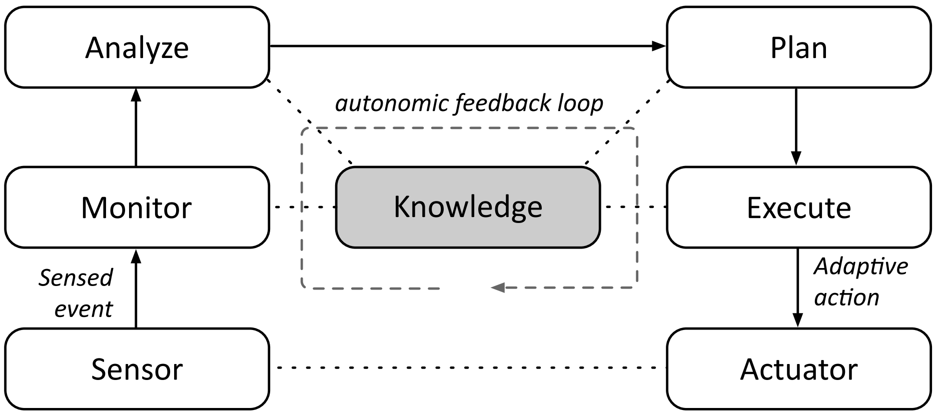

Figure G.45: Basic model of a MAPE-K autonomic control loop. Adapted from [49].

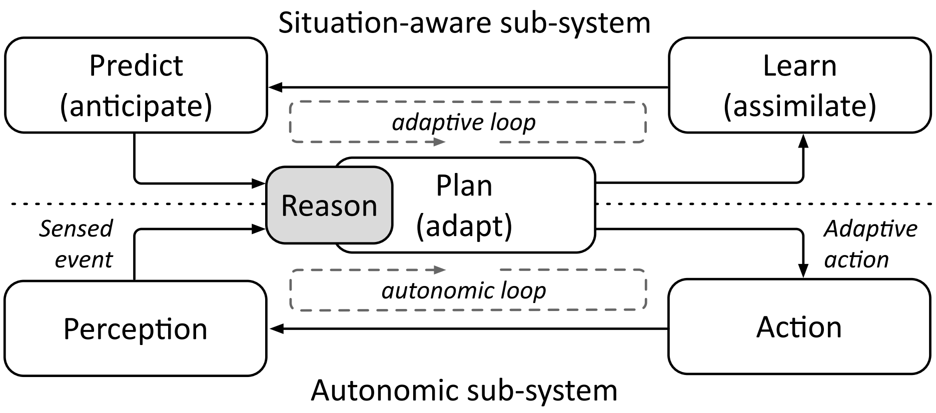

Figure G.46: A model of cognitive planning. Adapted from Vernon's cognitive cycle [29].

Pramanik et al. redefine the concept of Cognitive IoT found in network management as a process where a stateful and probabilistic system adapt to dynamic changes through situation-awareness and iterative self-learning [32]. This pattern closely resembles Vernon's cognitive cycle, which is based on two independent cycles of (1) perception and action; and (2) anticipation, assimilation and adaptation. In his model, planning is implicit in the process and intelligent behavior thus emerges through circular causality, where global system behavior influences local behavior of system components, while local interactions between components in turn determine global behavior [29]. In contrast to this model, all models in the reviewed cases include planning as a separate component. To reflect this, we created a pattern based on the cognitive cycle that we named *cognitive planning*, shown in Fig. G.46.

In the cognitive planning model, the planning process is the central component.

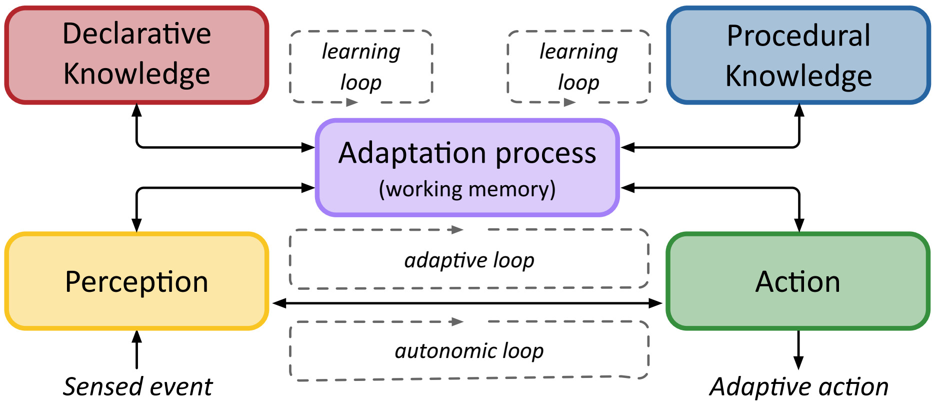

Figure G.47: A standard model of human-like minds. Adapted from [40].

The functionality is divided in an autonomic and a situation-aware subsystem, which are executed through two separate control loops. The *autonomic loop* registers an event and initiates a response in accordance to its active policy. The *adaptive loop* analyzes the event and changes the policy if the situation calls for adaptation. This division has two implications: First, the autonomic subsystem can operate without the need for learning, that is, as a simple stimuli-reaction loop. Second, since the situation-aware subsystem receives stimuli from the autonomic subsystem, it continuously learns from experience. We will later see that this increases autonomy and robustness of the process.

The cognitive reasoning process is initiated either from an observed event or from a prediction. Depending on stimuli received, the reasoning processes can trigger the planning process. This implies either initiating a learning process, performing an adaptive action, or both. To reflect this, we placed reasoning as a sub-process that precedes planning. This pattern matches both the reviewed architectures that actively use learning to adapt to changes in their context, and those that do not include a separate learning process, i.e., purely autonomic systems.

Cognitive architectures are often organized around a working memory that uses a cognitive cycle to collect sensory input, retrieve appropriate declarative or procedural knowledge and initiate adaptive actions. Laird et al. describe this architectural style in a standard model [40], shown in Fig. G.47. They state that a key characteristic of intelligent behavior is that changes in a working memory correspond to steps in an abstract reasoning process or internal simulation of an external action. Thus, adaptation emerges from a combination of the implemented architecture, acquired knowledge and learned skills. Implicitly, this corresponds to a system with three different types of control loops. The autonomic and adaptive loops have the same traits and behavior as in the cognitive planning model above. However, in the cognitive architecture the learning processes are outsourced to separate control loops that are responsible for updating declarative and procedural knowledge that is stored within the system, in accordance to dedicated learning policies.

Sifakis et al. concretize the idea of separating knowledge in [43] with a computational model for agents. This model includes declarative knowledge that represents facts about the world, i.e., the entities found in a domain and the relations between them. This knowledge can for instance be stored as system properties, training data, logical formulas or patterns. Procedural knowledge, on the other hand, is represented by executable methods like behavioral descriptions of components, analytical algorithms or prediction techniques like machine learning models.

G.6 Best Practices for Autonomous IoT Device Management Systems

Our literature study shows that complexity is a challenge in most large-scale deployments of autonomous IoT device management systems, due to scale, constraints and heterogeneity within the system, and non-stationarity of the environments. Based on our observations, we were able to identify five best practices (BP) that can guide the design and implementation of such systems. The first three BPs are a direct result from our observations, and BP4 and BP5 are derived from an overall analysis of the reviewed papers and the patterns observed in them. In the following paragraphs, we will describe these practices, and give recommendations for when and how to apply them. For clarity, we use imperative language.

BP1: Employ adaptation mechanisms according to environmental stationarity. We observe that the selected adaptation mechanisms correlate with the type of environment: In *stationary environments*, there is usually little need for explicit learning, as the statistical properties of the environment are constant. Adaptive actions can hence be pre-programmed and automated, using semantic methods or purely model-driven mechanisms. Learning processes in such systems are mainly used to identify new rules or policies. In contrast, when operating in *non-stationary environments*, environmental factors often vary from device to device, and they need to adjust their operation to unexpected events. This means that systems must be able to retain and store knowledge of observed events, the action that was taken, and the corresponding effect. They also have to reason about the implication of said action, to make the most suitable corrective action the next time the same situation arise. In other words, they have to employ advanced adaptation mechanisms grounded in previously collected data.

Therefore, assessing the environment of devices is critical: For systems managing devices operating in a stationary environment, the often lower complexity of a purely autonomic architecture may be sufficient. For systems operating in non-stationary environments, one should consider adding a self-aware subsystem to allow explicit learning, since this is often a prerequisite for adaptation under such conditions.

BP2: Select topology according to the inherent systemic constraints and requirements. System topology and locus of computation are influenced by constraints inherent to IoT. Constraints in connectivity favor computation on the device at the edge of the system. We found only one distributed topology [167], where Moser et al. use linear programming for adaptive energy management directly on the IoT device.

The majority of the papers are concerned with constraints inherent in the device rather than the network. This favors cluster or star topologies, as they have the potential for better access to memory, processing power, energy and contextual information. The main argument for employing a star topology is that a fully centralized management improves elasticity [280], allowing easier access to resources-as-a-service, which can add flexibility for systems where the management needs to vary over time. Another argument is that since the managing nodes also need to be managed autonomously, a star topology reduces overall complexity. Regarding the cluster topology, we see four benefits: 1) Clustering can help reducing latency, since processes are placed closer to the sensor devices [281]. 2) A topology that allows responsibility to be shared among the managing nodes in the network can be more suited to handle high variances in the network conditions or frequent changes in application requirements [250]. 3) With cluster organization we can connect devices that share similar characteristics. This adds flexibility, since it allows that management nodes can use resources on the aspects of management that are most relevant for that particular group [112, 282]. 4) Many IoT systems operate under conditions that may cause instability in the operation. A clustered topology can improve dependability since it reduces the risk that the whole network is shut down in case of internal or external events, like handling too many active connections at once or a power break [260].

We conclude that a star topology fits systems that require variable access to processing power, memory or storage. For systems where low latency or high dependability are key concerns, or where manager nodes are specialized for specific management tasks, a clustered topology may be preferable.

BP3: Separate concerns and reduce complexity with modularization. Modularization is a general mechanism to handle system complexity that also applies for device management systems. Consequently, the majority of the reviewed architectural models divide reasoning, learning and planning mechanisms into separate components. Planning is often done in a component with a central, coordinating role. Systems operating in stationary environments often employ a basic architectural model made up of few components, while higher environmental complexity is often handled by adding more components and using distributed reasoning mechanisms throughout the architecture to control the data flow between components. Such modularization allows specialized subcomponents for a particular task. This enables a better understanding and control of the data flow and of the different states of the system, making it easier to integrate mechanisms into managers and to support black-box designs [283]. Another benefit is that modularity ensures that parts of the system can be replaced or extended independently if the requirements or the understanding of the problem change, which can reduce the risk of project failures due to high complexity.

BP4: Control parallelism and data flow with triggers. Parallelism is a concern in architectures that manage many devices simultaneously. Adaptation processes can be active in several components at the same time. Models need hence to include a detailed description of how these processes work internally, that is, how they are activated and which components they activate in turn. However, few papers mention how and when components are activated.

We suggest to explicitly present triggering mechanisms that describe how knowledge is transferred from an originator to a consumer and how adaptation processes are activated. Such triggers should possess both *push* and *pull* directions, so that both originator and consumer component can trigger the knowledge flow. For instance, an observed event or a predicted future state can trigger the creation of a new plan, corresponding to an originator pushing knowledge towards a consumer. Vice versa, a planning component as a consumer of knowledge may trigger a specific procedure in pull direction if it needs a specific prediction for a future state. This mechanism also works for learning where, for example, a previously unseen event can trigger a command to update the declarative knowledge connected to that event.

BP5: Represent devices by digital twins. A challenge in autonomous management of constrained IoT devices is to keep track of the past, current and future state of each device *individually.* For such tasks, the concept of digital twins is used in industrialized IoT. A digital twin is a virtual representation which reflects a specific physical device. Such complete and holistic representation, as opposed to a more fragmented organization, makes it easier to model the behavior of the devices individually. A central manager can adjust the operation of each device based on their actual experience. The twin can then serve as a platform for simulating behavior and recommend optimal actions in a given situation [284].

Though we have not seen this concept explicitly in IoT device management, virtual representations of devices are central in several of the reviewed cases to allow autonomous self-management and context awareness [285]. One example can be seen in [241], where virtual objects are used to represent both end devices and devices that are responsible for adaptive management. This is challenging because it implies that a device and its twin need to stay synchronized for the representation and its result to be valid.

G.7 A Generalized Cognitive Model for Autonomous IoT Device Management

We synthesized a generalized cognitive model for autonomous management of constrained IoT devices, shown in Fig. G.48, based on the observations in Table G.11, the MAPE-K loop in Fig. G.45, the adapted model of cognitive planning from Fig. G.46, and the standard model of a cognitive architecture in Fig. G.47. It provides a blueprint that describes the reasoning, learning and planning processes for autonomous adaptation, the interaction between these processes, and the declarative and procedural knowledge involved in such a system.

G.7.1 Component Structure and Digital Twins

Apart from the physical device instances deployed in the field, the model is structured by two types of managers that serve a *system* and a *device* perspective:

- The *System manager* is responsible for assessing the past, present and future states of the system as a whole. It contains knowledge that explains the world, that is, the nature of the environment in which the system is placed, and how the environment influences the operation of the devices. When new data is collected, the system manager applies the necessary filters, before it separates the data into declarative knowledge belonging to either the world or a device,

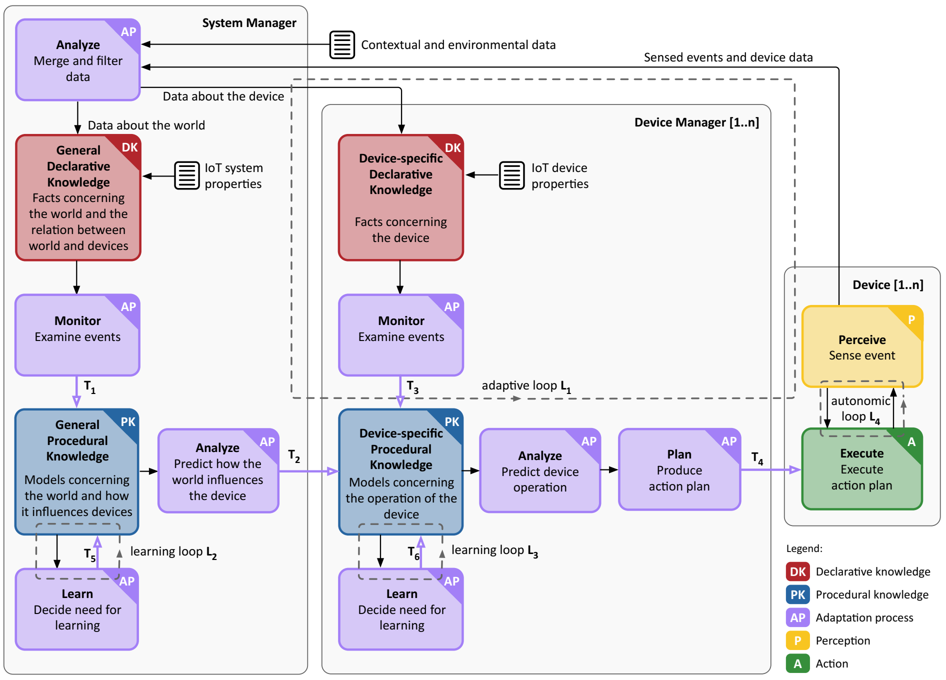

Figure G.48: A generalized cognitive model for autonomous management of constrained IoT devices.

respectively. The main output from the system manager are hence predictions of how external conditions will influence each device in the future.

- The *Device managers* are responsible for assessing and planning the operation of the devices themselves. Following the pattern of digital twins, there is one device manager instance for each physical device. Their main tasks are to monitor the knowledge related to each device and produce plans that reflect the context and environment in which each device is operating.

This separation between situation awareness and planning is prominent in many of the reviewed architectures, for example in [258], where Al-Turjman et al. use context awareness to optimize network lifetime in a wireless sensor network. Other examples are adaptive configuration management [7] and energy-aware self-management [159]. This organization facilitates sharing of information among devices, for instance as value distributions that model the environment in a given context or as generalized episodes that can be used to index a structure for matching and retrieval of similar cases [57]. Sharing of data may also be beneficial for devices with common properties [286], as for example solar energy intake patterns of devices with similar location and orientation [23].

G.7.2 Separation of Declarative and Procedural Knowledge

Both top-level components contain subcomponents to handle declarative and procedural knowledge separately. The system manager contains a component to handle declarative knowledge about the world, and the device managers one to handle declarative knowledge about the device. The more interpretive procedural knowledge is encapsulated in the respective modules PK.

This separation is beneficial as these different types of knowledge often use different implementations, as discussed in Sect. G.4. For machine learning, for instance, declarative knowledge can be stored in the form of training data, and procedural knowledge is represented by trained machine learning models. Many papers separate these types of knowledge, too. An example can be seen in [250], where Minh et al. separate policies from contextual knowledge. Other examples are dividing knowledge in contextual models, adaptation options, adaptation goals and plans [255], and separating a context repository from logical rules [257].

G.7.3 Control Loops for Adaptation

The model contains in total four control loops that handle adaptive behavior in the system.

- L_1 is the main control loop. It follows the MAPE-K pattern from Fig. G.45. However, since we have added procedural knowledge components that contain prediction models to the loop, it acts more like the adaptive loops seen in Figs. G.46 and G.47. L_1 can take two paths through the model, depending on whether a change originated in the environment or in a device. The monitor component in the system manager will detect a change in the environment, which will trigger an analysis of the contextual change. Likewise, the monitor component in the device manager will detect any change related to a single device. This triggers an analysis that evaluates to which degree the change influences the operation of the device, taking current and future environmental context into account.
- L_2 and L_3 are learning loops that control the incremental learning processes of the system manager and device manager, respectively. They follow the pattern of the situation-aware subsystem (Fig. G.46), as they act on incoming events and decide if there is a change that requires learning. Since declarative knowledge is added continuously, the learning loops are only specified for the procedural knowledge components in the model. This is a distinction from Fig. G.47.
- L_4 represents the autonomic subsystem from Fig. G.46. It controls the device in a short-term perspective based on the last plan sent from the device manager.

G.7.4 Explicit Triggers

In line with the best practice identified before, the model identifies explicit triggers that guide the data flow to the correct component and control the behavior of the management system in general, based on local decisions. The triggers are shown as arrows pointing from the originator of knowledge to its consumer, but they can be activated from both directions, as described by BP4 in Sect. G.6.

- A change in the general declarative knowledge base may trigger T_1, which in turn activates a model of the world or a model of how the environment influences a device.
- After running a general procedural model, the system manager analyzes the result and triggers T_2 if there is a change in the situation that can influence the operation of a device. This analysis may need to pull specific device data stored in the device manager, to run the analysis. The prediction or result of the analyzed situation is then sent to the device manager for further processing.
- Trigger T_3 monitors device-specific declarative knowledge to see if there is a need for activating a particular behavioral model of that device. After running a device-specific model or procedure, the device manager will produce a plan based on the result from this process. A policy-based planning procedure may for instance analyze the goal of the device, and then select the policy that addresses the present problem in the best way, taking recent results and predictions into account.
- If this new plan deviates from the previous, trigger T_4 informs devices that should adapt to this new plan, as we will outline later.
- In each manager there is a special reasoning process that triggers T_5 resp. T_6 if there is need for learning. Other types of learning triggers are for instance previously unseen data, or the discovery that an executed plan did not have the anticipated effect.

We see examples of such distributed reasoning processes in many of the reviewed cases. For instance, Shah et al. use this pattern to select which task to execute [261]. Other examples are matching similar situations based on previous experience [7], selecting training data based on correlation [23] and context-aware planning of corrective actions [268].

G.7.5 Adaptive Instructions vs. Actions

Trigger T_4 is a special adaptation process that serves two purposes: (1) It triggers the creation of an adaptation plan for a device and (2) controls the transfer of that plan to the mirrored device. The latter reflects the *motor action* commonly employed in cognitive architectures [40]. This combined mechanism effectively decouples the adaptive instruction, which is managed by a manager node, from the adaptive action, which is managed by the device. The purpose of this separation is to ensure that the digital twin and the device always are in sync. If they are not synchronized, the manager risks to wrongly interpret the effect of a corrective action, which again will cause it to make incorrect assumptions about the device. In addition, the learning processes may suffer, since unsynchronized states may cause noise or anomalies in the training data. One way to mitigate this problem is to keep the last instruction sent to the device in the working memory of the device manager until it receives an acknowledgment that the instruction was consumed by the device. Another argument in favor of decoupling is that a device capable of some basic reasoning might discover that it has a need for adaptation due to a sudden external event. When this happens, the decoupling allows a device to take an adaptive action on its own, and then send an instruction to the manager that informs it about the action that was made. Again, the device needs to keep a copy of the instruction in its memory until the device manager has acknowledged that the states are synchronized.

None of the reviewed cases explicitly address the difference between an adaptive instruction and an adaptive action. However, Sifakis et al. mention the importance of synchronizing agents and devices in [43].

G.7.6 Example Use Case: Solar-Powered IoT Devices

We illustrate the proposed model with a management system for solar-powered air quality sensing devices. It provides feedback to devices about their expected future solar energy intake based on the weather forecast, so that they can adjust their operation to the availability of solar energy and maintain perpetual, energy-neutral operation.

Apart from air quality data, the system manager collects data about the solar energy intake from the devices and the weather forecasts for the devices' locations. Data relevant to the individual device is passed on to the respective device managers. With this data, they can use a prediction model [287] to estimate the energy intake for each device. This estimation is used to plan how many measurements each device can take, also considering the current state-of-charge of the device. To speed up learning, the device manager may also employ training data from other devices [23] provided via the system manager. They may also employ a set of prediction models from which the currently best one is selected [22].

The four control loops guide the information flow through the system. The adaptive control loop L_1 examines the weather forecasts in respect to the device status, and instructs devices to change their operation if the anticipated energy budget changes significantly. L_2 and L_3 guide the learning process. Prediction models for energy intake can for instance be retrained at midnight, or when the training data significantly changes. The autonomic loop L_4 follows the policy sent by the device manager unless it detects a state-of-charge of the battery that is lower than anticipated by the device management. If that happens, it executes a local policy that restricts the amount of energy that is consumed until it receives updated instructions from the device manager.

G.8 Discussion

We developed the proposed reference model bottom-up, emerging from the systems found in literature. Still, there is no guarantee that it fits every specific device management use case perfectly. A challenge may hence be the application of the model to specific use cases. However, as we managed to unify the emerging model with the principles for autonomic computing (Sect. G.5), we argue that the resulting model with its loops, main components and responsibilities is likely to be relevant for a wide range of device management tasks. Further, the reference model should not be seen as a rigid design, but rather a blueprint for an architecture that can be further refined following the best practices listed in Sect. G.6.

Our survey reveals the wide range of adaptation and learning mechanisms, often more than one within a single system, see Sect. G.4. This diversity is good, but developers are often only familiar with a subset of techniques, and not necessarily the most suitable ones. Hence, while our model covers the overall system, the initial selection, application and detailed design of the specific learning and adaptation mechanisms can remain a challenge. Here we expect a maturation of the field, where best practices also regarding the detailed learning and adaptation mechanisms become commonplace and ultimately off-the-shelf components. The current rise in interest and competence within machine learning among developers makes this a likely scenario.

For further research and development, we see the mapping of the reference model to standard components of commercial device management platforms and platforms that automate machine learning tasks. Alongside the expected maturing of the field regarding the adaptation and learning mechanisms for specific concerns, we expect this to be the main driver for consolidation and further progress.

Even though handling security is out of scope for this review, we stress the importance of considering this aspect when designing device management systems. Security should be handled as an integral part of an architecture. Apart from integrating mechanisms like authentication, authorization and certification, security functions are increasingly subject to learning and adaptation, and hence drawing benefit from a better management of them.

G.9 Conclusion

We reviewed the state of the art for autonomous device management in IoT. First, we conducted a comprehensive and structured study on the aspects that need to be considered when designing and implementing systems for autonomous management of constrained IoT devices. We further synthesized a taxonomy of the most commonly used adaptation mechanisms in IoT device management and studied how and when they are applied. We combined these findings with general state-of-the-art models of autonomous systems to identify common patterns for autonomous management. From the conducted work, we made two major contributions that help advancing the field: first, we managed to summarize insights of the literature by identifying five best practices for design and implementation; second, we synthesized a generalized cognitive model for autonomous management of constrained IoT devices. This generalized model follows the identified best practices, adheres to the general principles of autonomic computing and connects them with the requirements for the IoT domain. In this way, the proposed model contributes to the vision of efficient and sustainable IoT systems that reduce or prevent expensive downtime or human intervention.

References

[3] Jantunen, E., Di Orio, G., Hegedus, C., Varga, P., Moldovan, I., Larrinaga, F., Becker, M., Albano, M., and Malo, P. "Maintenance 4.0 world of integrated information". In: *Enterprise interoperability VIII*. 2019.

[4] Brock, D. M. "Autonomy of individuals and organizations: Towards a strategy research agenda". In: *International Journal of Business and Economics* vol. 2, no. 1 (2003).

[5] Lewandowski, T., Henze, D., Sauer, M., Nickles, J., and Bruegge, B. "A software architecture to enable self-organizing, collaborative IoT resource networks". In: *2020 Fifth international conference on fog and mobile edge computing (FMEC)*. IEEE. 2020, pp. 70–77.

[6] Sifakis, J. "System design in the era of IoT — Meeting the autonomy challenge". In: *1st International workshop on methods and tools for rigorous system design (MeTRiD 2018)*. 2018, pp. 1–22.

[7] Foteinos, V., Kelaidonis, D., Poulios, G., Vlacheas, P., Stavroulaki, V., and Demestichas, P. "Cognitive management for the Internet of Things: A framework for enabling autonomous applications". In: *IEEE Vehicular Technology Magazine* vol. 8, no. 4 (2013), pp. 90–99.

[15] Colakovic, A. and Hadzialic, M. "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues". In: *Computer Networks* vol. 144 (2018), pp. 17–39.

[20] Zheng, J., Cai, Y., Shen, X., and Zheng, Z. "Green energy optimization in energy harvesting wireless sensor networks". In: *IEEE Communications Magazine* vol. 53, no. 11 (2015).

[21] Tien, J. M. "Internet of Things, real-time decision making, and artificial intelligence". In: *Annals of Data Science* vol. 4, no. 2 (2017).

[22] Braten, A. E. and Kraemer, F. A. "Towards cognitive IoT: Autonomous prediction model selection for solar-powered nodes". In: *2018 IEEE international congress on internet of things (ICIOT)*. 2018, pp. 118–125.

[23] Braten, A. E., Kraemer, F. A., and Palma, D. "Adaptive, correlation-based training data selection for IoT device management". In: *Sixth international conference on Internet of Things: Systems, management and security (IOTSMS)*. 2019.

[29] Vernon, D. *Artificial Cognitive Systems: A Primer*. 2014.

[31] Sinche, S., Raposo, D., Armando, N., Rodrigues, A., Boavida, F., Pereira, V., and Silva, J. S. "A survey of IoT management protocols and frameworks". In: *IEEE Communications Surveys & Tutorials* (2019).

[32] Pramanik, P. K. D., Pal, S., and Choudhury, P. "Beyond automation: the cognitive IoT. Artificial intelligence brings sense to the Internet of Things". In: *Cognitive computing for big data systems over IoT: Frameworks, tools and applications*. Ed. by Sangaiah, A. K., Thangavelu, A., and Meenakshi Sundaram, V. 2018, pp. 1–37.

[35] Tahir, M., Ashraf, Q. M., and Dabbagh, M. “Towards enabling autonomic computing in IoT ecosystem”. In: *2019 IEEE intl conf on dependable, autonomic and secure computing, intl conf on pervasive intelligence and computing, intl conf on cloud and big data computing, intl conf on cyber science and technology congress (DASC/PiCom/CBDCom/CyberSciTech)*. 2019, pp. 646–651.

[36] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. “Context aware computing for the Internet of Things: A survey”. In: *IEEE Communications surveys & Tutorials* vol. 16, no. 1 (2014), pp. 414–454.

[37] Chen, M., Herrera, F., and Hwang, K. “Cognitive computing: Architecture, technologies and intelligent applications”. In: *IEEE Access* vol. 6 (2018), pp. 19774–19783.

[40] Laird, J., Lebiere, C., and Rosenbloom, P. “A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics”. In: *AI Magazine* vol. 38, no. 4 (2017).

[43] Sifakis, J. “Autonomous systems – An architectural characterization”. In: *Models, languages, and tools for concurrent and distributed programming. lecture notes in computer science*. 2019.

[49] Corporation, I. *An architectural blueprint for autonomic computing*. Vol. 31. 2006. 2006, pp. 1–6.

[50] Arcaini, P., Riccobene, E., and Scandurra, P. “Modeling and analyzing mape-k feedback loops for self-adaptation”. In: *IEEE/ACM 10th international symposium on software engineering for adaptive and self-managing systems (SEAMS)*. 2015.

[52] Kephart, J. O. and Chess, D. M. “The vision of autonomic computing”. In: *Computer* vol. 36, no. 1 (2003), pp. 41–50.

[55] Sheth, A. “Internet of Things to smart IoT through semantic, cognitive, and perceptual computing”. In: *IEEE Intelligent Systems* vol. 31, no. 2 (2016), pp. 108–112.

[56] Sterritt, R. “Autonomic computing”. In: *Innovations in Systems and Software Engineering* vol. 1, no. 1 (2005), pp. 79–88.

[57] Aamodt, A. and Plaza, E. “Case-based reasoning: Foundational issues, methodological variations, and system approaches”. In: *AI Communications* vol. 7 (1994), pp. 39–59.

[61] Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., and Long, K. “Cognitive Internet of Things: A new paradigm beyond connection”. In: *IEEE Internet of Things Journal* vol. 1, no. 2 (2014), pp. 129–143.

[68] Delicato, F. C., Pires, P. F., Batista, T., et al. *Resource Management for Internet of Things*. Vol. 16. Springer briefs in computer science. 2017.

[112] Fortino, G., Guerrieri, A., Russo, W., and Savaglio, C. “Middlewares for smart objects and smart environments: overview and comparison”. In: *Internet of Things based on smart objects: Technology, middleware and applications*. Ed. by Fortino, G. and Trunfio, P. 2014.

[114] Nascimento, N. M. do and Lucena, C. J. P. de. “FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things”. In: *Information Sciences* vol. 378 (Feb. 2017), pp. 161–176.

[159] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N., and Palma, D. “Solar energy prediction for constrained IoT nodes based on public weather forecasts”. In: *Proceedings of the Seventh International Conference on the Internet of Things*. New York, New York, USA, 2017, pp. 1–8.

[167] Moser, C., Thiele, L., Brunelli, D., and Benini, L. “Adaptive power management for environmentally powered systems”. In: *IEEE Transactions on Computers* vol. 59, no. 4 (2010), pp. 478–491.

[193] Edalat, N. and Motani, M. “Energy-aware task allocation for energy harvesting sensor networks”. In: *EURASIP Journal on Wireless Communications and Networking* vol. 2016 (2016), pp. 1–14.

[232] Shresthamali, S., Kondo, M., and Nakamura, H. “Adaptive Power Management in Solar Energy Harvesting Sensor Node Using Reinforcement Learning”. In: *ACM Transactions on Embedded Computing Systems* vol. 16, no. 5s (Oct. 2017), pp. 1–21.

[237] Preden, J. S., Tammemae, K., Jantsch, A., Leier, M., Riid, A., and Calis, E. “The Benefits of Self-Awareness and Attention in Fog and Mist Computing”. In: *Computer* vol. 48, no. 7 (2015), pp. 37–45.

[239] Murad, A., Kraemer, F. A., Bach, K., and Taylor, G. “Management of Energy-Harvesting IoT Nodes Using Deep Reinforcement Learning”. In: *13th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2019)* (2019).

[240] Zhang, M., Zhao, H., Zheng, R., Wu, Q., and Wei, W. “Cognitive internet of things: concepts and application example”. In: *International Journal of Computer Science Issues* vol. 9, no. 3 (2012), pp. 151–158.

[241] Sasidharan, S., Somov, A., Biswas, A., and Giaffreda, R. “Cognitive management framework for Internet of Things:—A prototype implementation”. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014, pp. 538–543.

[247] Chowdhury, A. and Raut, S. A. “A survey study on Internet of Things resource management”. In: *Journal of Network and Computer Applications* vol. 120 (2018).

[248] Kitchenham, B. and Charters, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Tech. rep. Keele University and Durham University Joint Report, 2007.

[249] Vogler, M., Schleicher, J. M., Inzinger, C., and Dustdar, S. “A Scalable Framework for Provisioning Large-Scale IoT Deployments”. In: *ACM Transactions on Internet Technology* vol. 16, no. 2 (2016).

[250] Minh, T. C., Bellalta, B., and Oliver, M. “Dison: A self-organizing network management framework for wireless sensor networks”. In: *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.* 2013.

[251] Sahni, Y., Cao, J., Zhang, S., and Yang, L. “Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things”. In: *IEEE Access* vol. 5 (2017).

[252] Slabicki, M., Premsankar, G., and Di Francesco, M. “Adaptive configuration of lora networks for dense IoT deployments”. In: *IEEE/IFIP Network Operations and Management Symposium (NOMS).* 2018.

[253] Lan, L., Shi, R., Wang, B., and Zhang, L. “An IoT Unified Access Platform for Heterogeneity Sensing Devices Based on Edge Computing”. In: *IEEE Access* vol. 7 (2019).

[254] Da, K., Roose, P., Dalmau, M., Nevado, J., and Karchoud, R. “Kali2Much: a context middleware for autonomic adaptation-driven platform”. In: *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT (M4IOT).* 2014.

[255] Weyns, D., Iftikhar, M. U., Hughes, D., and Matthys, N. “Applying architecture-based adaptation to automate the management of internet-of-things”. In: *European Conference on Software Architecture (ECSA).* 2018.

[256] Megahed, A., Tata, S., and Nazeem, A. “Cognitive Determination of Policies for Data Management in IoT Systems”. In: *Service-Oriented Computing, ICSOC 2017 Workshops. Lecture Notes in Computer Science).* Vol. 10797. 2018.

[257] Lee, K. W. and Cha, S. H. “Ontology-based context-aware management for wireless sensor networks”. In: *Communications in Computer and Information Science.* Vol. 214. 2011.

[258] Al-Turjman, F. M. “Information-centric sensor networks for cognitive IoT: an overview”. In: *Annals of Telecommunications* vol. 72 (2017).

[259] Shah, V. S. “Multi-agent cognitive architecture-enabled IoT applications of mobile edge computing”. In: *Annals of Telecommunications* vol. 73, no. 7-8 (2018).

[260] Athreya, A. P., DeBruhl, B., and Tague, P. “Designing for self-configuration and self-adaptation in the Internet of Things”. In: *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing.* 2013.

[261] Shah, K., Di Francesco, M., Anastasi, G., and Kumar, M. “A framework for Resource-Aware Data Accumulation in sparse wireless sensor networks”. In: *Computer Communications* vol. 34, no. 17 (2011).

[262] Alam, G. R., Hassan, M. M., Uddin, Z., Almogren, A., and Fortino, G. “Autonomic computation offloading in mobile edge for IoT applications”. In: *Future Generation Computer Systems* vol. 90 (2019).

[263] Hilal, A. R. and Basir, O. "A collaborative energy-aware sensor management system using team theory". In: *ACM Transactions on Embedded Computing Systems* vol. 15, no. 3 (2016).

[264] Samie, F., Tsoutsouras, V., Xydis, S., Bauer, L., Soudris, D., and Henkel, J. "Distributed QoS management for Internet of Things under resource constraints". In: *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. 2016.

[265] Portocarrero, J. M. T., Delicato, F. C., Pires, P. F., and Batista, T. V. "Reference architecture for self-adaptive management in wireless sensor networks". In: *International Conference on Adaptive and Intelligent Systems*. 2014.

[266] Park, J.-h., Salim, M. M., Jo, J. H., Sicato, J. C. S., Rathore, S., and Park, J. H. "CIoT-Net: a scalable cognitive IoT based smart city network architecture". In: *Human-centric Computing and Information Sciences* vol. 9, no. 1 (2019).

[267] Wei, Q. and Jin, Z. "Service discovery for internet of things: a context-awareness perspective". In: *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*. 2012.

[268] Cristea, V., Dobre, C., and Pop, F. "Context-aware environments for the Internet of Things". In: *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence. Studies in Computational Intelligence*. 2013.

[269] Fleurey, F., Morin, B., Solberg, A., and Barais, O. "MDE to manage communications with and between resource-constrained systems". In: *Model Driven Engineering Languages and Systems. MODELS 2011. Lecture Notes in Computer Science*. Vol. 6981. 2011.

[270] Chen, T., Barbarossa, S., Wang, X., Giannakis, G. B., and Zhang, Z.-L. "Learning and Management for Internet of Things: Accounting for Adaptivity and Scalability". In: *Proceedings of the IEEE* vol. 107, no. 4 (2019).

[271] Gurgen, L. and Honiden, S. "Management of Networked Sensing Devices". In: *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. 2009.

[272] Meddeb, M., Alaya, M. B., Monteil, T., Dhraief, A., and Drira, K. "M2M platform with autonomic device management service". In: *Procedia Computer Science*. 2014.

[273] Sezer, O. B., Dogdu, E., and Ozbayoglu, A. M. "Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey". In: *IEEE Internet of Things Journal* vol. 5, no. 1 (2017).

[274] Celikyilmaz, A. and Turksen, I. B. *Modeling uncertainty with fuzzy logic*. 2009.

[275] Negnevitsky, M. *Artificial Intelligence: A Guide to Intelligent Systems*. 2005.

[276] Eddy, S. R. "What is dynamic programming?" In: *Nature Biotechnology* vol. 22, no. 7 (2004).

[277] Zarri, G. P. “High-Level Knowledge Representation and Reasoning in a Cognitive IoT/WoT Context”. In: *Cognitive Computing for Big Data Systems Over IoT. Lecture Notes on Data Engineering and Communications Technologies*. Vol. 14. 2017.

[278] Craw, S. and Aamodt, A. “Case Based Reasoning as a Model for Cognitive Artificial Intelligence”. In: *Case-Based Reasoning Research and Development. ICCBR 2018. Lecture Notes in Computer Science*. 2018.

[279] Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., and Zambonelli, F. “A survey of autonomic communications”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* vol. 1, no. 2 (Dec. 2006), pp. 223–259.

[280] Van den Abeele, F., Hoebeke, J., Teklemariam, G. K., Moerman, I., and Demeester, P. “Sensor Function Virtualization to Support Distributed Intelligence in the Internet of Things”. In: *Wireless Personal Communications* vol. 81, no. 4 (2015).

[281] Park, T. and Saad, W. “Distributed learning for low latency machine type communication in a massive internet of things”. In: *IEEE Internet of Things Journal* vol. 6, no. 3 (2019).

[282] Malo, P., Almeida, B., Melo, R., Kalaboukas, K., and Cousin, P. “Self-organised middleware architecture for the internet-of-things”. In: *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. 2013.

[283] Bianchini, R., Fontoura, M., Cortez, E., Bonde, A., Muzio, A., Constantin, A.-M., Moscibroda, T., Magalhaes, G., Bablani, G., and Russinovich, M. “Toward ML-centric cloud platforms”. In: *Communications of the ACM* vol. 63, no. 2 (2020).

[284] Mohanty, S. and Vyas, S. “Intelligence of Things = IoT + Cloud + AI”. In: *How to Compete in the Age of Artificial Intelligence*. 2018.

[285] Savaglio, C., Ganzha, M., Paprzycki, M., Badica, C., Ivanovic, M., and Fortino, G. “Agent-based Internet of Things: State-of-the-art and research challenges”. In: *Future Generation Computer Systems* vol. 102 (2020).

[286] Wei, Y., Zhang, Y., Huang, J., and Yang, Q. “Transfer Learning via Learning to Transfer”. In: *International Conference on Machine Learning*. 2018.

[287] Kraemer, F. A., Palma, D., Braten, A. E., and Ammar, D. “Operationalizing Solar Energy Predictions for Sustainable, Autonomous IoT Device Management”. In: *IEEE Internet of Things Journal* vol. 7 (2020), pp. 11803–11814.

Appendices

Fog Computing in Healthcare — A Review and Discussion

Frank Alexander Kraemer, Anders Eivind Braten, Nattachart Tamkittikhun, David Palma

Abstract

Fog computing is an architectural style in which network components between devices and the cloud execute application-specific logic. We present the first review on fog computing within healthcare informatics, and explore, classify and discuss different application use cases presented in literature. For that, we categorize applications into use case classes and list an inventory of application-specific tasks that can be handled by fog computing. We discuss on which level of the network such fog computing tasks can be executed, and provide tradeoffs with respect to requirements relevant to healthcare. Our review indicates that (1) there is a significant number of computing tasks in healthcare that require or can benefit from fog computing principles, (2) processing on higher network tiers is required due to constraints in wireless devices and the need to aggregate data, and (3) privacy concerns and dependability prevent computation tasks to be completely moved to the cloud. These findings substantiate the need for a coherent approach towards fog computing in healthcare, for which we present a list of recommended research and development actions.

H.1 Introduction

As Topol writes in *The Creative Destruction of Medicine* [288], healthcare stands before its most fundamental changes ever. One driver of these changes is wireless sensor technology. Besides giving access to an increasing number of biometric parameters, sensors are also getting smaller, so that they can be worn without obstructing everyday life. This is important when data needs to be collected continuously. The BioStamp [289], for instance, is a sensor the size of a band-aid that can measure various biometric signals and simply be attached to the skin. Further, Kang et. al [290] describe an optimized technique to print sensors directly onto adhesive film that can be attached to skin. Contact lenses also offer possibilities of sensing a number of biometrics [291]. Such advances promote a scenario in which patients are instrumented with dozens of sensors. In addition comes the abundance of fitness trackers. They foreshadow a future in which each human, regardless of health status, is continuously monitored.

Sensory data is only useful if we can derive insights from it. Such insights are provided by other drivers in healthcare, like big data and machine learning, the accuracy of which will soon exceed that of humans [292]. Apart from automatic or assisted analysis of medical images, big data analysis can be used to study the effectiveness of treatments, identify patients at risk for chronic diseases, ensure that patients adhere to treatment plans, optimize processes and personalize care [293].

To monitor patients at this scale, sensors need to be wearable and wireless. This constraints their size, and influences the amount of energy, memory and processing capacity that they can offer. In addition, data is only valuable in context and needs to be aggregated from several sensors. Sensors therefore send it to other, more capable computing devices for analysis, aggregation and storage. The wireless ECG monitoring system *IntelliVue* from Philips [294] for instance, requires its own installation of access points and network switches in order to seamlessly forward ECG data to central servers. However, such vertical approaches do not scale. When many patients should be instrumented, each requiring a high number of sensors, these cannot be supported by their own, dedicated infrastructure, as such individual infrastructures are expensive and hard to maintain.

The Internet of Things (IoT) offers an alternative approach. Sensor devices can use a common infrastructure to forward their data to more comprehensive applications, using standardized protocols such as 6LoWPAN [295] over IPv6 [296]. Connectivity is provided by border routers, that connect the wireless resource-constrained nodes to existing network infrastructures. This enables a device-to-cloud architecture in which the infrastructure between device and cloud is only used as a communication channel. Cloud computing frees the sensors from battery-draining computing tasks and provides virtually unlimited resources. The cloud is also one possible place where data from different sensors can be aggregated, enabling the large-scale data sets required by the analysis tasks mentioned above.

For many applications within health informatics, however, such a simplistic sensor-to-cloud architecture is not feasible. In some cases, regulations do not allow to store patient data outside the hospital. For some applications, relying entirely on remote data centers is also unacceptable because of patient safety in case of network and data center failures.

One possible solution to bridge the gap between sensors and analytics in health informatics is *fog computing*. This is an architectural style for distributed systems in which application-specific logic resides not only in data centers (the cloud) or the devices closest to the users, but also in the infrastructure components between them. Examples of such infrastructure components are gateways, routers and access points. This added flexibility of computation opens new possibilities for solving healthcare challenges. Better patient mobility and increased integration will enable uninterrupted monitoring as introduced above, and also enable entirely new applications, as discussed later.

We observe an increasing number of publications on fog computing principles in general, including applications within healthcare. In most cases, however, little effort is spent to discuss where computation tasks should be placed, or the tradeoffs between different requirements. To advance the application of fog computing in healthcare, it is important to understand such tradeoffs holistically, taking into account the diverse requirements of several, interacting applications and the vision of future medicine as outlined above. This raises three questions:

- Which computational tasks in health informatics can be processed by fog computing?
- Which are potential locations in the Internet of Things where these tasks can be executed?
- Which are the tradeoffs to consider when placing computational tasks in the system?

To find answers to these questions, we performed a systematic review of pervasive health applications relevant for fog computing. We conducted a broad search within international journals, conferences and workshops, using the sources listed in Table H.13. We looked for papers addressing personal sensor network applications in general, and wireless healthcare applications in particular. To identify relevant publications, we set up three groups of search terms, summarized in Table H.14. The

Table H.13: Sources used in the search of relevant publications

Primary source	URL	Use cases found
IEEE Xplore	ieeexplore.ieee.org	17
ACM Digital Library	dl.acm.org	2
ScienceDirect	sciencedirect.com	2
SpringerLink	link.springer.com	3

Table H.14: Terms used in search for relevant publications

Network topology	Architecture		Healthcare	
Wireless sensor network	Fog computing	Pervasive computing	Mobile healthcare	Ubiquitous healthcare
Ubiquitous sensor networks	Edge computing	Wearable computing	IoT Healthcare	Health telemonitoring
Body area network / BAN	Mobile edge computing	Cyber-physical systems	Home healthcare	Critical care monitoring
Personal area network / PAN	Mobile cloud computing	Monitoring systems	Pervasive healthcare	Non-invasive monitoring
Local area networks / LAN	Ubiquitous computing	Ubiquitous mobility systems	Telehealth	
Wearable wireless sensors	Wearable computing	Wireless health		

first two groups encompass the terms that the authors use to describe the network topology and the architecture, respectively. They set the technical boundaries for the study. The third group of terms addresses the different phrases that are used to describe healthcare in a wireless or mobile setting. Whenever we found a paper that used a new term relevant to our study, we added that term to the corresponding group, and conducted a new search to find other publications using the same phrase.

The search resulted in 163 papers, published between 2005 and 2016, that we found relevant to our study after reading the abstracts. Out of this pool, we discarded 73 after conducting a full-text review. From the papers left, we identified the network topologies and requirements of the solutions, and extracted 24 relevant use cases for further analysis. The sources of these use cases are listed in Table H.13.

Previous reviews have addressed the thematic of healthcare related to wireless sensor networks [297] and body area networks [298], the Internet of Things [299], ubiquitous and pervasive computing [300] and mobile computing [301]. However, to the best of our knowledge, there has not yet been a survey of fog computing within healthcare.

Our review and discussion contributes the following:

- An overview of benefits and challenges of fog computing.
- A review of healthcare applications and the computing tasks that are relevant for fog computing.
- An overview of network and device types in different deployment scenarios.
- A review of where fog computing tasks are placed.
- A discussion of the tradeoffs when placing fog computing tasks, with respect to requirements in healthcare.
- A list of recommended research and development actions.

The paper is structured as follows. In Sect. H.2, we will present the concept of fog computing and list the main characteristics and benefits discussed in literature.

In Sect. H.3, we present some of the trends and challenges in healthcare, and provide an overview of medical sensors and actuators and their technical requirements. In Sect. H.4, we survey healthcare applications, categorized based on deployment scenario and use case class, and provide an inventory of computation tasks that are suitable for fog computing. In Sect. H.5 we provide an overview of the most relevant technologies for wireless health. We then analyze the architecture of applications in literature, and find out in which hierarchy levels of the network fog computing tasks are executed. In Sect. H.6 we discuss the benefits and challenges of the applications and architectures we have reviewed, and discuss selected tradeoffs. We conclude with an overview of the current state of research, and outline further research and development demands for applying fog computing within healthcare.

H.2 Fog Computing

The term *fog computing* was initially coined by industry [302] as a metaphor for the main architectural idea behind it: fog is somewhere between the cloud (data centers) and the ground, where the users' devices are located. A term often used synonymously is *edge computing*, describing tasks that are placed at the edge of the network in contrast to the cloud. Note that the term *edge* can refer to different tiers of the architecture. In an industrial setting, *edge* often refers to nodes in a production plant and resides on premisses with the user, for instance as part of a machine controller or a network gateway [303]. ETSI's terminology [304] takes the perspective of internet service providers, referring to edge as the border of the operator's network, like for instance an LTE base station. Our understanding of fog covers both of these perspectives.

The main characteristic of fog computing is its topology, i.e., the geographically distributed nodes that perform computation and offer storage and network services. Fog computing resources can be integrated into access points, routers and network gateways alongside the generic network functions. There may also be dedicated fog computing nodes, like the mobile edge computing (MEC) servers deployed at LTE base stations and access points described by ETSI [304]. Other devices can be dedicated gateways deployed at home, like home automation hubs. The specific types of tasks that fog computing performs depend on the specific application and domain. In general, tasks contain filtering, aggregating, analyzing and temporarily storing data.

Fog computing can be performed on a single fog computing node or on several nodes jointly. This can improve scalability and provide redundancy and elasticity, adding more fog nodes when more computing power is needed. Mechanisms like virtualization and sand-boxing can be used to execute applications, which is why fog computing shares many of the principles of cloud computing. Central to fog computing is the concept of computation *offloading*, which has been treated in research for instance by cloudlets [305], and can also be found in what is called *mobile cloud* [306]. Similarly, crowd computing focuses on the utilization of distributed computation power provided by, for instance, mobile devices [307].

There is a consensus in literature that fog computing is not intended to replace cloud computing, but rather view it as an *perfect ally* [308] or an *extension* [302] of it. [309] also points out how many of the technologies and properties like elasticity used for cloud computing also apply for fog computing.

In the following, we explain and exemplify the benefits of fog computing mentioned in literature. We discuss and evaluate these benefits with respect to healthcare later.

Reduced Latency: Compared to a device-to-cloud architecture, placing processing closer to the devices can reduce the latency since the physical distance is shorter and potential response time in a data center can be removed. Compared to a device-only architecture, latency can be reduced since computation-intensive tasks that take a long time on resource-constrained sensor devices can be moved to more capable fog computing nodes. The motivation can also be to keep the latency predictable [310].

Privacy: Compared to the device-to-cloud architecture, fog computing can reduce the propagation of data, for instance by analyzing sensitive data on a local gateway instead of a data center outside of the control of the user. This can improve the privacy of user data [311].

Energy Efficiency: There are several ways how fog computing can improve energy efficiency within sensor devices. First, gateways can serve as communication proxies, so that devices can increase the length of their sleep cycles. During the sleep mode, the gateway takes care of any requests or updates, which are then processed when the sensor device wakes up. Second, energy-intensive computations and other services can be offloaded from the battery-driven nodes [310].

Bandwidth: In comparison to a device-to-cloud architecture, fog computing can reduce the volume of data to be sent into data centers. This can happen in several ways: Raw data can be filtered, analyzed, pre-processed or compressed so that only a reduced amount of data needs to be forwarded [312, 313]. Local nodes can also answer requests from devices based on locally cached data, so that communication with data centers is not necessary at all [314].

Scalability: Fog computing can improve the scalability of a system. Local computation can reduce the load from more centralized resources, and be expanded as needed. Vaquero [311] refers to this as "mini-clouds."

Dependability: Fog computing can increase system dependability in two ways. It can be a means to realize redundancy, by letting several nodes in the network provide the same functionality. It can also execute computation closer to the sensor nodes, so that they are less dependent on the availability of a network connection to more centralized resources [308].

Context: In some cases, a fog computing node is the first node in a network that has enough overview to reason about a situation and the context of data. An example is a system that induces the current activity of medical staff from the location and activity of several devices [315].

H.3 Wireless Health Informatics

We briefly review the current challenges in healthcare, give an overview of the variety of sensors and their requirements.

H.3.1 Challenges for Healthcare

Healthcare systems in most countries face enormous challenges that will increase due to aging population and the rise of chronic diseases. Many countries also experience a growing nursing staff shortage. At the same time, there is a demand to reduce costs while maintaining high-quality care to patients [316]. As a consequence, healthcare industry promotes an information-centric healthcare delivery model [317]. Part of this delivery model enables remote monitoring of patients, which leads to increased accessibility, quality, efficiency, and continuity of healthcare to patients, and also reduces the overall cost of healthcare [318].

Today, much time is wasted in hospitals by manually measuring biometric parameters and transferring the data between systems, often involving pen and paper. Remote monitoring will free time for caretakers. Other improvements include automated supervision that can replace manual supervision. Bertini et al. [319] report benefits of remote monitoring compared to in-hospital follow-ups, including even a positive impact on survival. Another area is the improvement of processes within the hospital. Many processes are planned manually, and therefore done sequentially, instead of using resources more effectively. In addition, sensors will make it simpler to gain correct information about the current status and location of equipment, caretakers and patients. Sensors will also provide a more precise picture of patients, as they can capture data continuously and allow an insight into increasing variety of biometric parameters. This will revolutionize diagnostics and treatment. Topol [288] calls this "digitizing humans." Once this new picture of patients is matched with analytical techniques, new insights will transform early detections, diagnostics, medication and treatment of diseases. One precondition for this is that data is not treated in isolated silos, but that it is combined with other sources and seen in context.

Another trend is the departure from reactive treatment, where patients are treated in a hospital only after an incident, towards a more preventive medicine [320]. This starts by monitoring healthy people, to keep them out of hospital for as long as possible. Additionally, increasing the possibilities to monitor patients at home facilitates releasing them earlier from the hospital. In general, this means that the borders between hospital, home, and other points of care get increasingly blurred: healthcare happens continuously and everywhere.

H.3.2 Medical Devices: Wireless Sensors and Actuators

There is a wide variety of sensors, in different stages of technology readiness. Tanaka et al. [321], for instance, developed an incontinence sensor integrated in diapers. The sensor uses urine as an electrolyte between two electrodes, which allows it to send an ID signal with a range of 5 meters once coming into contact with urine. A similar principle is used for drug prescription. A digestible microchip the size of a sand particle is integrated into a pill that generates a signal once in contact with digestive juices [322]. This signal is detected by a skin patch, which relays it further to a mobile phone. Examples for actuators are hearing aids, medication dispensers (both intra- and extra-body) or pace makers. The iPill from Philips [323], for instance, is a small device swallowed by a patient, which senses the acidity of its surroundings, in order to release drugs via a pump at the right place in the gut.

H.3.3 Requirements of Healthcare Applications

All of the potential benefits of fog computing listed in Sect. H.2 are relevant for healthcare. We now exemplify the corresponding requirements and, where appropriate, quantify them.

Bandwidth: The bitrates of different physiological signals depend on the number of leads, the quantization step-size of the analog-to-digital converter (ADC) in bits, and the sampling frequency [324]. Body temperature, for instance, requires only a low sampling frequency of 0.2 Hz. With a 12-bit ADC, this results in a bitrate of 2.4 bit/s [325]. Blood pressure sampled at 120 Hz with 12-bit ADC yields 1.44 kbit/s [325]. Pulse oximetry needs to be sampled at 600 Hz and requires 7.2 kbit/s [325]. Electrocardiograms (ECG) usually require more than one lead. For clinical applications, a 5-lead ECG needs between 36 to 216 kbit/s, depending on the sampling rate and step size [324, 326]. Electromyograms (EMG) represent electrical signals generated by muscles and can be used in several applications such as food chewing recognition [327] and prosthetic finger control improvement [328]. These use cases require a bandwidth of at least 20.48 kbit/s and 96 kbit/s, respectively. Electroencephalogram (EEG) measures electrical activities from the brain and requires a lot of leads. A 192-lead EEG can demand 921.6 kbit/s bandwidth. This shows that the bitrates of physiological signals vary considerably.

Latency: With regard to latency, the requirements also vary considerably with the intended use for the data. For ECG, Alesanco and Garcia [326] found through experiments with cardiologists that latencies of up to 2 to 4 seconds in real-time monitoring are acceptable. These are relatively lax requirements from a technical point of view. Stricter requirements are necessary for applications within the realm of the Tactile Internet [329], for instance for the control of exoskeletons which allow paralyzed patients to walk. Other examples with latency constraints come from telehealth applications operating in rural areas, where the network infrastructure itself is often restricted [330].

Energy-Efficiency: Energy-efficiency is a major concern, because replacing batteries impedes the use of sensors. While some in-body sensors rely on energy-harvesting, either by heat or kinetic energy [298], some sensors may require an operation of the patient when a battery needs replacement.

Dependability: Depending on what data is used for, system failures have different consequences, from minor inconvenience to serious threat to the patients' lives. Thus, dependability is one of the most important requirements to consider, tightly interconnected with resilience against security threats.

Security: Because of the sensitivity of patient data and the potentially severe consequences of tampered or manipulated devices and systems, the security requirements in healthcare are high. With respect to remote monitoring, increased connectivity of devices results in larger attack surfaces. This requires procedures for detection and fixes of security vulnerabilities that are complex. Requirements go beyond technologies implemented in the devices and surrounding systems, but also require routines that need to be in place in organizations, regulators and manufacturers. See, for instance, [331] for an overview.

Interoperability: Systems, even when provided by different vendors, should be interoperable with each other. This is often not the case. Cardiology patients, for instance, who should be transported between hospitals and who require close monitoring via ECG, need to be attached to different equipment during the transfer due to incompatibilities [324].

H.3.4 The Vision of Fog Computing in Healthcare

The apparent match between healthcare challenges, the resulting requirements, and the benefits of fog computing as presented in literature suggests a potential for fog computing as a driver for pervasive, ubiquitous computing in healthcare:

Flexibility of computation locus: Where scalability, privacy and dependability issues prevent a cloud-only solution, fog computing can offer the needed computational resources within the network to meet both regulatory and technical requirements. For such approaches to be effective, it is not only important to have computational resources between sensors and cloud, but also to optimally manage them. This includes transparency of execution for application, as well as a flexibility regarding *where* computation can be executed. With fog computing, the location can be dynamic and depend on the current context, environment and application requirements.

Integration: In the current landscape, the introduction of new sensor devices often requires the simultaneous introduction of a support infrastructure. An example is the heart rate monitoring system mentioned in the introduction, which requires dedicated infrastructure. This is a considerable burden when introducing new, innovative devices. Within a fog computing architecture, new sensors can be added to the existing infrastructure. Fog computing can also serve as a compatibility layer to translate between various standards.

Patient Mobility: Application-specific infrastructure also limits the area where patients can be monitored. This is especially relevant when patients are about to leave the highly instrumented infrastructure of a hospital. Current use cases often do not cover this transition, which can effectively prolong a patient's stay at the hospital. With fog computing resources in place, the transitions between different environments can be managed more gradually.

New applications: Fog computing will also enable entirely new applications: By adding higher levels of autonomy and intelligence at the edge, fog computing will provide latency and response time improvements, as well as energy savings for wearable and low-cost devices, while performing complex tasks such as fall detection [332]. The next generation of healthcare devices will replace costly and complex devices, without resorting to simple algorithms with limited accuracy. These devices will be enabled by fog computing, ultimately leading to the "Internet of Healthcare Things."

H.4 Health Applications

In this section, we start with a description of deployment scenarios, give concrete examples of each type of scenario, and categorize healthcare applications into different use case classes. We then present an inventory of computation tasks that are candidates for fog computing.

H.4.1 Deployment Scenarios

From the reviewed papers, we extracted five deployment scenarios, illustrated in Fig. H.49. The scenarios differ in terms of involved users and stakeholders, devices and connectivity:

- **Mobile:** In this scenario, the mobile phones of users act as hub between sensor devices and cloud.
- **Home Treatment:** When at home, connectivity is often provided through the patient's internet access. This has influence on device ownership, required usability and maintainability, and how disturbances can be mitigated.
- **Hospital:** Within a hospital, devices are often proprietary, and are usually owned and maintained by the hospital itself. The systems are considerably more complex, which in turn requires the users of the applications to be qualified professionals.
- **Non-Hospital Premises:** Like hospitals, this scenario covers professional points-of-care, but with less staff and infrastructure. Examples are clinics, doctor's offices or nursing homes. Core devices are owned and maintained by the clinic, but patients are sometimes required to connect personal equipment to the network.
- **Transport:** This scenario covers connectivity in an ambulance or helicopter. It is similar to the non-hospital deployment scenario, but with the added complexity that the infrastructure needs to be mobile, for instance using a cellular connection.

H.4.2 Example Use Cases for Deployment Scenarios

In the following, we present example use cases that are typical for their respective deployment scenario. They are also illustrated in Fig. H.50.

- **Mobile:** An example for the mobile deployment scenario is the monitoring system for chronic obstructive pulmonary disease (COPD) patients in [333]. A mobile phone acts as mobile base unit and collects data from several sensing devices, processes it and sends it to a back end server. The purpose of placing fog computing on the mobile device is to increase battery life of the wearable sensor device.
- **Home Treatment:** The Parkinson speech analysis solution in [334] is an example for the home deployment scenario. A fog node is placed on the LAN-level in the network hierarchy. Like in the mobile scenario, fog computing is used to collect, store and process raw data, before sending it to the cloud for permanent storage. The main motivation for fog computing is to reduce network traffic and latency. Another example of a home deployment scenario is described in [335], where data from patient- and environmental sensors are used to detect if a patient falls, and raise alerts about gas leaks and fires.

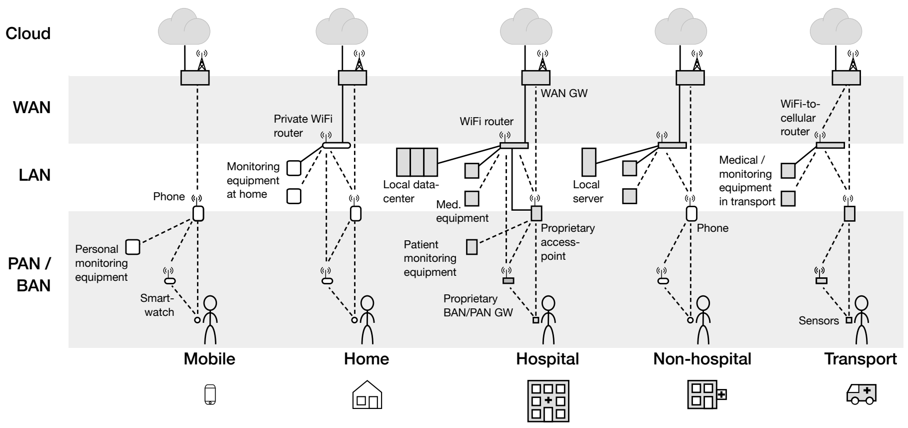

Figure H.49: Deployment scenarios in healthcare. Devices are used in different network layers. Examples are sensors and actuators, gateways, routers, access points, servers and data centers. We distinguish between devices and infrastructure owned or controlled by the health institutions (grey), and devices and infrastructure owned or controlled by the patient (white).

- **Hospital:** In [336] we see a typical example of a setup used in the hospital deployment scenario. Smart shirts, coupled with beacons, are used to monitor physiological data and the location of patients. Fog computing is distributed among several nodes. The data acquisition and processing board (DAPB) collects, processes and merges data from the sensors, and sends them to the wireless transmission board (WTB). The WTB collects data from the beacon points (BPs), merge them with the data from the DAPB and sends them in a single packet to the management subsystem, located at LAN level. The management subsystem uses the data from the DAPB and BPs to monitor the medical parameters of the patients, locates the patient within the hospital and verifies if an alarm has been activated.

- **Non-Hospital Premises:** The real-time epileptic seizure detection system [337] is an example of the non-hospital deployment scenario. A three-tier architecture is proposed, where filtering, preprocessing, feature extraction, feature selection and classification of EEG patterns are performed on the mobile device cloud (MDC), which is placed in the middle tier. Two advantages of using fog computing are mentioned: Providing sub-second real-time responses with minimal communication overhead, and reducing traffic between the local area network and the seizure detection system located in a cloud center.

- **Transport:** The transport deployment scenario is used in the ubiquitous e-health information interchange solution, described in [338]. The authors describe how physiological and contextual data can be collected in an emergency situation from a patient wearing a medical device, and how this information can be duplicated and shared between different devices on-site, in the ambulance and in the hospital. Fog computing is only concerned with

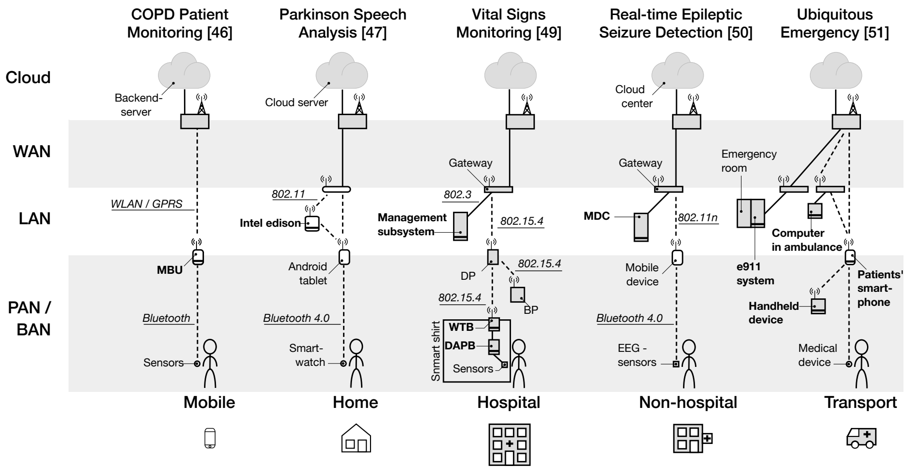

Figure H.50: Examples of actual deployment in heathcare. Mobile scenario: COPD Patient Monitoring [333]. Home scenario: Parkinson Speech Analysis [334]. Hospital scenario: Vital Signs Monitoring [336]. Non-hospital scenario: Real-time Epileptic Seizure Detection [337]. Transport scenario: Ubiquitous Emergency [338]. Fog computing nodes are marked with a black line at the bottom.

the collecting and sending of data, the complexity lies in the distribution of data among many fog nodes.

H.4.3 Use Case Classes

To facilitate our discussion, we have synthesized five use case classes, summarized in Tab. H.15. The table's columns show whether the use case class requires significant, application-specific *computing*, short response times (*real-time*), the *criticality* for the patient's health, and ability to provide *feedback* to control medical devices.

- **Data Collection.** This class of use cases only deals with the collection of data, which is then further examined by a doctor when needed. Examples are the logging of training activity, weight or body posture. The criticality of such data is low. If the system fails to log some data points, the patient is still safe.
- **Data Analysis.** This class extends the data collection with some automatic analysis of the data to gain further insights. An example is the speech analysis for Parkinson's patients [334]. Similarly to the data collection, the criticality of the data is low. This use case, however, requires considerable computation of data.
- **Critical Analysis.** These use cases analyze data for critical conditions. Examples are cardiac monitoring via ECG with automatic alarms once critical situations are detected [314]. The criticality also implies a certain maximum response time, i.e., real-time properties.

Table H.15: Use case classes with their properties.

Use Case Class	Computing	Real-time	Critical	Feedback
Data Collection	-	-	-	-
Data Analysis	●	-	-	-
Critical Analysis	●	●	●	-
Critical Control	●	●	●	●
Context Management	●	(●)	-	-

- **Critical Control.** In this class of use cases, detected events are not only used to alert personnel, but also to control devices. An example is a device that regulates the amount of oxygen provided to a patient [339].

- **Context Management.** This class of use cases is different from the ones above. It merely observes patients, devices, or employees to figure out their context and help by improving planning or taking proper decisions. This usually requires data analysis, but no or only lax real-time constraints. Examples are systems to figure out the context of healthcare workers [315].

H.4.4 Computing Tasks

Table H.16 lists the healthcare applications we examined, grouped by use case classes. If an application has several functions, we show it only once under the use case class with the most critical requirements. The third column describes which computing tasks are to be executed and subject to fog computing. Column four shows to which deployment scenario an application belongs. The final column summarizes at which levels of the network fog computation happens, explained later in Sect. H.5.4.

We will now illustrate some of the computation tasks listed in Table H.16. The *data collection* use case class is exemplified by the ubiquitous emergency scenario presented in [338], in which data is aggregated and exchanged among involved parties only in emergency cases. As a side effect, energy consumption due to the transmission of data is also reduced, increasing the effective operating time of the device. The *data analysis* class is exemplified by the speech tele-treatment system for Parkinson's patients described in [334]. The speech analysis, performed on a local gateway, reduces processing time and traffic to the cloud, while remote doctors can still retrieve the analyzed data from the cloud. The *critical analysis* class can be seen in [337], which describes a solution for real-time epileptic seizure detection. During a seizure, patients are usually unable to press a button, but automatic detection ensures that healthcare personnel is alerted and the treatment can be started quickly. The analysis for the seizure detection is done on local servers for low latency, while big data analysis is offloaded to the cloud. A *critical control* use case is found in [339], where an automatic oxygen-controlling system for COPD patients is proposed. An example of the *context management* class is [315], which determines the activity of staffs based on their location and devices being used.

Table H.16: Reviewed healthcare applications, grouped by use case classes.

REF.	APPLICATION	COMPUTING TASKS	DEP. SCENARIO	FOG COMP.
Data Collection *Data gathering from devices*				
[321]	Urinary Incontinence Detection	Detect signal from sensor, forward information into the system.	Non-hospital	PAN
[340]	Priority-Based Health Data Aggregation	Temporarily cache sensor data, analyze data to classify its priority, select which data to forward.	–	PAN + BAN
[338]	Ubiquitous Emergency Scenario	Forwarding of emergency calls and aggregation of patient data.	Transport	PAN
Data Analysis *Data analysis of the collected data*				
[307]	Activity Monitoring	Analysis of movement data in context of the location. Forwarding of relevant data into the system.	Mobile	PAN
[313]	ECG Data Compression	Encoding of ECG data to compress it and save transmission energy, decoding at the receiver.	–	BAN + PAN + LAN
[334]	Parkinson Speech Analysis	Caching of audio files and local feature extraction on audio files to analyze data. Forward analysis results, not raw data.	Home	PAN + LAN
[341]	Patient State Monitoring	Local analysis of video and audio data to figure out if a patient is in pain. No forwarding of raw data, only processed results.	–	–
[342]	UV Radiance Measurement	Analysis of camera data to measure UV level, aggregation of data from several phones to determine UV level in a specific area.	Mobile	–
[343]	Speech Recognition and ECG Monitoring	Analysis of speech data as in [334]. Analysis of ECG data to detect arrhythmic beats, compression and forwarding of relevant data.	Mobile	PAN + LAN
[344]	Image-Based Healthcare Analysis	Local processing of image data to assess wounds, detect skin cancer and detect heart rate.	Mobile	–
Critical Analysis *Data analysis for critical conditions with alarms when critical situations are detected*				
[312]	Fall Detection	Local analysis of accelerometer data for falls, with filtering of false positives, based on training data received from the cloud.	Mobile	PAN
[314]	ECG Monitoring	Feature extraction of ECG data based on wavelet analysis. Real-time notification and enrichment of data with location.	Home	BAN + LAN
[333]	COPD Patient Monitoring	Analysis of ECG data, local analysis and enrichment with GPS and activity, forwarding of relevant data in compressed form.	Mobile	PAN
[335]	Dementia and COPD Patient Monitoring	Real-time analysis of environmental and patient sensors to detect and alert users about fires and gas leaks. Monitoring a range of patient behavior, including fall detection.	Home	LAN
[345]	Arterial Blood Pressure Monitoring	Low-pass filtering to reduce noise.	Hospital	–
[346]	ECG Monitoring	Feature extraction of ECG data. Classification and detection of anomalies with local alarms. Filtering, transmission of results into the system.	Transport	PAN
[347]	Vital Signs and Environment Monitoring	Capturing and encoding of various types of biometrical and environmental sensor data. Authentication and encryption of data before transmission.	Hospital	PAN + LAN
[336]	Vital Signs Monitoring	Preprocessing and merging of data from a smart shirt, to add activity and location as context.	Hospital	BAN + LAN
[348]	ECG Monitoring	Applying low-pass and high-pass filters on ECG data to remove noise and baseline wandering.	–	BAN + LAN
[337]	Real-time Epileptic Seizure Detection	Analysis and preprocessing of EEG data, local analysis based on wavelet transformation, classification based on machine learning and notification of local staff.	Non-hospital	PAN + LAN
Critical Control *Control of actuators that are critical for patients*				
[339]	Oxygen Level Control	Analysis of oxygen level and patient activity to adjust the appropriate oxygen dose for the patient in real-time, also taking location and environmental data into account.	Mobile	–
[349]	Pacemaker Monitoring and Configuration	Monitoring and visualization of current pacemaker parameters. Local support to remotely update of pacemaker parameters.	Hospital	LAN
Context Management *Observation to deduce the context of a patient or healthcare personnel*				
[315]	Activity-Awareness of Medical Staff	Analysis of location, equipment usage and time to derive the current activity and availability status and improve planning and collaboration.	Hospital	PAN
[237]	Activity Monitoring	Local analysis of heart rate, acceleration and altitude to classify activity, like driving, resting or different types of walking.	Mobile	PAN + LAN

H.5 Fog Computing Architectures

Before we discuss the placement of fog computing tasks, we discuss the types of networks and devices typically found in healthcare.

H.5.1 Network Types

The reviewed pervasive health use cases employ combinations of four network types to bridge the gap between medical devices and the cloud: wireless personal area networks (WPANs), wireless body area networks (WBANs), local area networks (LANs), and wide area networks (WANs). The hierarchy of these networks is shown on the vertical axis of Fig. H.49. Some sensor devices are directly connected to the WLAN via Wi-Fi [312, 333]. Especially in the mobile deployment scenario, devices are directly connected to a WAN via cellular connections [307, 346].

Another way to connect sensors is by WPAN technology, as provided by Bluetooth, IEEE 802.15.4, or ZigBee. These typically have a lower range than

Wi-Fi or cellular connections, but are also more energy efficient. However, WPAN technologies have limitations. For some applications, they do not offer the necessary bitrates for the biomedical signals, such as EEG or ECG (cf. Sect. H.3), especially if patients wear several sensors. Furthermore, electromagnetic signal transmissions are blocked by the body in some postures [298]. This either reduces the quality of the link or makes communications with in-body devices impossible.

To mitigate the challenges with WPANs, a specific standard for wireless body area networks (WBAN) was introduced with IEEE 802.15.6 [350]. It uses a one- or two-hop star topology with only one hub as gateway to other networks [298]. In addition, IEEE 802.15.6 proposes three different physical layers that can be chosen for different applications [350]. The *narrow band physical layer* provides longer communication range, with slightly lower data rates than some WPAN technologies [351]. The narrow band utilizes existing frequency bands such as 402–405 MHz medical device radiocommunications band (MICS) and 2.4–2.45 GHz industrial, scientific and medical band (ISM). The *ultra wide band physical layer* offers higher data rates than the narrow band with low transmission power. This layer can also be designed to achieve better energy consumption per bit than the narrow band [352]. The *human body communication layer* utilizes the galvanic coupling on the surface of the human body for data transmissions. This eliminates antennas and signal propagation problems. Additionally, it is considered to be the most energy-efficient physical layer for high-data-rate requirements [353].

Devices compliant with IEEE 802.15.6 devices are, to the best of our knowledge, still under development [353, 354]. Existing systems referring to WBANs therefore usually utilize WPAN standards, e.g. Bluetooth or IEEE 802.15.4, which may be sufficient for some applications that do not require high data rates or communications with in-body devices.

H.5.2 Device Types in Healthcare

Depending on the deployment scenarios of Fig. H.49, different devices and network nodes are involved. In the *mobile deployment* scenario, mobile phones act as WPAN gateways that connect directly to the WAN through cellular networks. WBAN gateways, such as smartwatches, can be used as intermediate nodes. Off-path nodes, for instance environmental sensing equipment, are connected at the WPAN-level. In the *home deployment* scenario, wireless routers act as gateways from Wi-Fi to WAN. The sensing devices communicate via a gateway on BAN- or PAN-level. This can be a specialized device, for instance mounted in a belt or another item of clothing, or it can be a mobile phone. Off-path computation nodes, like fall detection devices, are placed at LAN-level. In the *hospital deployment* scenario, local data centers are often available. On both LAN- and PAN-level there are other off-path computation nodes like localization devices and stationary equipment in labs or operation rooms. Patients wear proprietary devices which connect to specialized gateways connecting WBAN or WPAN to the LAN. In the *non-hospital deployment* scenario, e.g., a doctor's office or a nursing home, we typically see small local servers. Lab equipment or environmental sensing devices act as off-path computation nodes. Patients wear the same kind of non-intrusive sensing equipment as identified in the home scenario. A patient in the *transport deployment* scenario wears the same kind of proprietary sensing equipment as in the hospital scenario. This connects to a gateway on a WPAN network that acts as a bridge to a WLAN router in the vehicle. The WLAN router connects to the WAN through a cellular network while in transit. Medical equipment and wired monitoring devices are connected on LAN-level.

H.5.3 Development Platforms

For research and development, there is a wide variety of hardware development platforms with wireless communication with small form factors and low energy consumption. We list some of them that are frequently referred to in literature.

- The Arduino is a low-cost platform used in many application domains. It requires additional hardware modules for wireless communication [335]. nRF24L01 is a low-cost radio transceiver module that can be used with Arduino and other platforms. It is designed for the 2.4 GHz ISM band and optimized for low energy consumption [355].
- The MC13213 system is also based on an 8-bit processor, but has a higher clock rate of 4 MHz. It integrates a 2.4 GHz transceiver module on the chip that supports 802.15.4 and ZigBee [336].
- Intel Edison is another system-on-chip (SoC) with integrated Wi-Fi and Bluetooth 4 radio modules. With its 400 MHz processor it is suitable for computing power-demanding applications like audio processing [334].
- CSEM's Icycom is a platform with a 900 MHz ISM band transceiver unit and a 16/32-bit microprocessor. Its form factor is about 1×1 cm with low energy consumption [356].
- Another SoC suitable for WPAN applications is the nRF51822. This chip supports 2.4 GHz BLE and can communicate with nRF24L01 providing that a BLE stack is implemented for nRF24L01. Unlike nRF24L01, the nRF51822 has an integrated 32-bit processor, yet it still consumes low energy and comes in a tiny package comparable to the nRF24L01 [357].

H.5.4 Positioning of Fog Computing Tasks

We also examined at which level of the network and in which devices the reviewed papers place computation tasks. This is summarized in the last column of Table H.16. (A dash indicates authors did not reveal enough information.)

Numerous approaches ([307][312][315][321][335][338]) place their computation task on a single node at either PAN- or LAN-level. At this level, data is processed and forwarded to higher levels and eventually to the cloud. There is a wide variety of tasks. A typical use case is to collect and analyze time-critical data, in order to achieve critical monitoring, like fall detection [335]. Another example is [307], which describes a sensing platform where a global task scheduler in the cloud is offloading a computation strategy to a worker node in the fog. This instructs the worker node to collect and filter only the most important and relevant data.

Other approaches ([313][314][334][336][340][343][348][237][358]) utilize two or more fog-nodes on a direct path between the sensor device and an access point to the cloud. An example is described by López et al. [336]. They have developed shirts with embedded sensors that collect physiological data about the patient. The shirts include a wearable data acquisition device that also acts as a BAN-gateway. The device processes the data and sends it to a management system at LAN level, where the data is further processed and permanently stored. In addition, the management receives data from a separate off-path location system, that collects positioning data. In cases where several fog computing nodes are used, we observe that the node closest to the sensor device is typically used for pre-processing or filtering. In-depth analysis, contextualisation and local storage is usually done on a node located closer to the cloud, often at LAN level.

We also observed approaches ([333][347]) which use a gateway node at PAN or BAN-level for computation, and where the node is capable of connecting to either LAN via Wi-Fi or WAN via a cellular network connection. This is especially useful for applications which need a high degree of mobility, and where flexibility with regard to network connectivity is important. Wac et al. [333] describe a scenario where a patient is wearing one or more sensors along with a mobile base unit, connected in a BAN. The base unit collects, synchronizes, filters and processes the data, before sending it further to a back-end server for storage via either WLAN or a cellular network. Huang et al. [347] describe a wearable sensor system where physiological data about the patient are captured by on-body biomedical sensors, and then encrypted locally before the data is sent to a mobile computing device (MCD) on a higher network tier. A separate system of sensor motes sends environmental data. The combined data sets are then captured and analyzed by the MCD before the data is eventually sent to a back-end system for permanent storage. The MCDs are also able to communicate with each other via a cellular network. This case also shows that a fog node can use different types of networks, depending on the type of data it is sending.

H.6 Discussion

After performing our review and going through the use cases, we conclude that fog computing, despite its potential, is still in an early phase within healthcare, and only implemented partially, if at all. The main shortcoming of the collective literature is that many of the works focus on isolated use cases, and often only discuss infrastructures that are accordingly specialized. Most use cases also only cover a single deployment scenario. This leads to the lack of a unified view, one that is required by the grander vision of fog computing for healthcare as introduced earlier. We will come back to this shortcoming, after discussing the various aspects of fog computing in health care.

H.6.1 Locus of Computation

Our survey in the previous section shows that computing tasks occur at several levels of the network, from BAN to cloud. This suggests that the distribution of computational tasks should not simply be focused on a node's hierarchical level. The placement of offloading computing tasks within an infrastructure is rather non-trivial. The different roles and computing resources of available devices require a careful consideration of the tradeoffs and complementarity between possible options, bearing in mind target-levels of performance like computational performance, latency limitations, energy consumption and security, to name a few [310]. Even though for some operations it may seem obvious that a resource-powerful environment as provided by cloud computing is preferred to another with fewer capabilities, constraints such as privacy may limit the number of available options, and for instance block information from leaving the hospital premises.

We have also observed that the health-specific deployment scenarios have a significant impact on decisions related to the implementation of the fog concept, despite the generalized acceptance of fog computing anywhere between the cloud and a device (c.f. H.4.1). In health informatics, clear examples of this impact are noticeable when comparing computation of fog tasks within hospital premises, with other locations where healthcare activities are also provided but where less resources may be available (e.g., doctors' offices and nursing homes). Other examples include first-responders' interventions in areas where access to typical communication and computing infrastructures may be extremely limited, creating new challenges and opportunities for fog computing.

When considering the different levels at which offloading can be performed, from the device to the cloud, enforcing local processing (e.g., within a facility) may be of paramount importance when reliability is discussed (c.f. H.6.3). This local processing does not invalidate the cooperation between local nodes and outbound servers. In fact, they could overlap, but it does provide additional guarantees in case of connectivity loss to the exterior and may be a requirement for critical systems. Privacy and regulations that can be coupled to a given scenario, particularly in health informatics, may however raise stricter constraints and require offloading tasks to take place within certain restrictions (see H.6.4).

H.6.2 Latency and Throughput

Previous works shows that computation offloading offered by fog computing, in nodes in the vicinity of constrained devices, can reduce latency up to 2.88 times [359], when compared against offloading to the cloud. This result is strongly influenced by the existing local resources and the ones used in the cloud which, with the steady increase of available data bitrates and a wide coverage of 3/4th generation cellular networks, should only depend on the amount of used servers (i.e., access to the network infrastructure is almost negligible). Nonetheless, the increasing number of nodes and highly specialized sensors raises scalability concerns and latency-sensitive applications may require improved mechanisms to handle the delay between the sensors and the cloud [310].

Theoretically [360], using dedicated servers at the edge of the network (e.g., cloudlets [361]), performance improvements have been achieved but they disregard the pervasiveness of IoT devices and their distinct characteristics. Additionally, while it is intuitive that performing computing tasks locally should improve latency, throughput and even energy consumption [362], several technical challenges have to be considered (e.g. VM or container deployment time, resource management, among other aspects). In fact, these mechanisms may be responsible for adverse effects, becoming a burden to fog nodes and hindering the desired improvements [309]. In many cases, we see that the benefit of reduced latency is taken as granted, without a precise quantification of the specific requirements and an evaluation of different solutions.

Latency and throughput may be improved by fog computing through the reduction of the amount of data transmitted between source and destination, relieving the core of the network and the overall system [363]. This load reduction may also reduce the likelihood of transmission errors and can be achieved by performing computing tasks such as filtering, feature extraction or even prediction [314, 362, 364]. Applications related to face and speech recognition require large amount of data and it is shown that local computations can reduce latency [341]. However, the performance improvement of resource-constrained devices will also rely on devices being capable of bridging network technologies (e.g. 802.15.4/6 with 802.11). Fog computing must be able to leverage on the diversity of resource-constrained nodes and their capabilities, throughout the hierarchy of network infrastructures, in order to scale and provide faster response times [360].

Even though the cloud is typically seen as the endpoint for the data transmitted by a node, this data may actually begin a new life-cycle within the cloud. For instance, it often needs to be delivered to another node (e.g. an actuator or doctor's computer), after being appropriately processed. This process within the cloud is prone to additional latency, but fog computing can significantly increase the performance of bandwidth-intensive and latency-sensitive applications when compared against a pure "node to cloud to node" solution [314, 309]. Ultimately the impact of latency and related metrics (e.g. jitter), must be considered in the Quality of Experience (QoE) registered by doctors and patients in general.

H.6.3 Dependability

The dependability of health applications is crucial, especially for the use case classes of critical monitoring and critical control (Sect. H.4.3). Any single point of failure requires careful consideration. With regard to cloud-based solutions, the general availability of data centers is high, but outages are still a problem, even with redundancy in place [365].

The network towards data centers may also be subject to failures. Ultimately, any of the connections towards a central data center in the different deployment scenarios (Sect. H.4.1) can fail, although some are more exposed than others. Ambulances can drive through areas without cellular coverage, or patients at home may loose connection to their Wi-Fi routers. This raises the question to which degree cloud services can be used for critical use cases. In the description of many use cases we believe that these aspects are not sufficiently addressed.

Local computation can be used to either completely replace critical tasks done in data centers, or to use local processing when there are limitations in the cloud [366]. An example is feature extraction to analyze patient ECG data in real-time [314, 336]. If caretakers rely on this function to monitor the well-being of patients, the analysis must not be interrupted. When done on a nearby gateway, it can also be performed when the data center or the connection towards it are down. For the use case class of data collection, where it is only important that data *eventually* arrives at some data base, fog nodes may also buffer data locally until it can be transferred further.

Like the cloud, fog computing nodes are also subject to failure. However, consequences and nature of failure are different from that of cloud computing. Failures in the cloud or the network towards it can affect an entire hospital. In contrast, when resources of a lower network hierarchy fail, consequences affect a smaller area, like hospital sections or single wards. Such minor incidents are often easier to handle with respect to re-equipment or re-staffing. Also, fog computing can lead to architectures with built-in redundancy on a local level, with several fog computing nodes acting as fault tolerant sets [366], which increases dependability.

H.6.4 Security

Davies et al. [367] argue that privacy concerns due to "over-centralisation" of IoT systems are a critical obstacle to their growth. Even though data can be protected on its way into the cloud and within data centers, a suitable strategy to protect data is to avoid sending it off premisses in the first place, and process it closer to its source [311]. The proximity of fog devices, which can be placed within ones infrastructure, may introduce the required trust and enforce the necessary privacy mechanisms that threat cloud computing in critical scenarios. An example is an application to analyze speech from patients with Parkinson's disease [334]. Instead of sending audio recordings into a data center, analysis happens locally, and only result metrics are forwarded. Privacy, though, still remains an issue in more decentralized solutions such as fog computing. Trust and authentication need to be handled, particularly when considering multi-vendor equipment and purely wireless devices. The decoupling between nodes and access points, or gateways, opens the possibility for rogue or compromised fog nodes to hinder the benefits of locality [368, 369]. In order to achieve a decentralised network between fog nodes and mobile nodes or sensors, interoperable trust models must be established, as well as software and physical security mechanisms to protect the networks and their nodes. Another way to make offloading of computation work on untrusted fog nodes is verifiable computing [370], given that the computational tasks can be efficiently mapped to the operations available under these conditions.

Fog nodes can also contribute to security functions. As they often have more computational power than constrained sensor devices, they may assist with cryptographic operations [371]. A link between a sensor device and a BAN gateway may be protected by symmetric encryption, which is supported by many embedded sensor nodes. The patient data may be further secured by the BAN gateway using schemes as proposed in [372], before sent further into the network. Fog nodes may also host other security functions such as intrusion detection [373], or explicit control of which information may leave a location [367].

The pervasiveness of things and fog-capable technologies will also introduce a new era for Human-Computer Interaction (HCI) and its relationship with the security of users and their nodes. In addition to the wireless nature of devices, which limits the users' ability to identify the "next hop in the loop," the size or the lack of input/output peripherals in some of them, creates new challenges. These systems will require simple, yet robust, Authentication, Authorisation and Accounting (AAA) mechanisms that do not compromise the functionalities of devices and their mobility between different networks.

H.6.5 Autonomic Fog Computing

Fog computing adds flexibility regarding where computation can be placed. To mitigate the increased complexity coming with this, dynamically managed behaviours are expected in the IoT and fog computing paradigms [311, 309], availing existing resources and coordinating actions for overall improved performance. Context and scenario-specific requirements are fundamental for enabling efficient and autonomic management in fog computing, being aligned with IoT in order to fully exploit its potential [374]. The use of big data is particularly relevant as an enabler for context-aware management [375], taking into account nodes and their different roles in the infrastructure [376]. However, these considerations must handle the heterogeneity of devices and vendors without introducing unbearable overheads.

Overall, the consideration of multiple parameters for improving management in fog computing should scale with the dissemination IoT devices and their distinct uses. This requires nodes to take part in the decision-making process when flexible reconfigurations are needed, without compromising their own purpose in the system. Such process should promote node autonomy or self-awareness, together with dynamic procedures triggered by standard pre-defined protocols [237]. These standard mechanisms are important for guaranteeing interoperability between devices [374, 377] and “cross the chasm” of the Internet of Things [367].

H.6.6 Energy Efficiency

Besides the energy spent for the actual sensing procedure, the main energy consumers on sensor devices are computation and the transmission of data. Fog computing facilitates energy-efficient sensor devices by offloading expensive computation. Hu et al. [361] show how offloading functionalities to cloudlets improves the energy-consumption of mobile devices significantly. These results are for mobile applications. For BANs, the cost for sending may be different, so that the energy gains through less processing can be reduced by increased cost for sending. Usually, fog computing nodes are energy-rich, which is why they are suitable for offloading in the first place. However, if fog nodes are mobile, like a mobile phone in the mobile deployment scenario, there is also a tradeoff between the energy consumption of the mobile phone and the sensor device. Tradeoffs like these may require autonomic reasoning in the device to determine in which situation which strategy is most efficient.

H.6.7 Insights and Future Directions

Despite some of the identified flaws and shortcomings in literature, fog computing emerges as a necessary architectural ingredient for ubiquitous computing. Due to the wide range of applications and use cases that can be considered, the extent to what offloading computing tasks can benefit health informatics has still not been fully explored. But fog computing has already proven its effectiveness in terms of bandwidth utilization and latency, for example when considering ECG feature extraction [314]. This is also backed by previous work showing improvements in latency and energy consumption resulting from offloading tasks from constrained devices to more powerful nodes, using common networking solutions such as Wi-Fi and 3/4G [354, 355]. The other main driver for fog computing is dependability. No matter how connectivity improves, outages can ultimately only be covered by computing and storage closer to the sensors, which corresponds to fog computing. Employing fog nodes enables the usage of smarter and autonomous decisions at the fog layer, regardless of cloud availability. This, however, presupposes interoperability between heterogeneous devices and systems.

The possibility of introducing local data processing, adaptation and storage, enabled by fog computing, has also an impact beyond security, latency and interoperability. It also creates new possibilities for actuation, autonomous reconfiguration, devices discovery, mobility and even energy efficiency [378]. Examples of these new prospects include robotic prescription dispensing and medication delivery [379], which must consider medical data collection, formatting, analyzing and storing, as well as the administration of medication according to patients' medical records, as we have seen for instance with the COPD treatment system [339].

The successful dissemination of fog computing in healthcare will not only be influenced by its advantages. An additional driver are restrictions such as regulations imposed by, for example, the Organization for Economic Cooperation and Development (OECD). Fog computing may help users and service-providers to overcome these restrictions. Fog nodes may be used for providing a layer between the end-users, service providers and the cloud, confining private or sensitive health information within trusted devices [380].

We have seen the demand for computation between sensor and cloud in virtually all use cases related to ubiquitous healthcare. However, our review also revealed the lack of a unified strategy or overall architecture for fog computing in healthcare, and pervasive healthcare applications in general. This lack of cohesion undermines the potential of IoT and fog computing. Systems are often seen in isolation, since creators of a specific system focus on isolated use cases, deployment scenarios or sensor technology. Based on these insights and identified shortcomings, we see demand in the following areas for research and development:

Standardization Within Healthcare. The challenge with most of the use cases we reviewed is that they span across several devices, systems and deployment domains, and therefore lack a single, well-defined stakeholder. Even hospitals, which cover many use cases, may not be sufficient since much of healthcare will also happen outside of their scope. The Continua Alliance [381] is one example for such standardization with special focus on personal health devices. In this context, it should be explored whether and how fog computing can be utilized to increase interoperability through its flexibility to offer computation, i.e., by enabling a heterogeneous, service-based architecture in which computation can take care of interoperability tasks.

Standardization of Fog Computing Mechanisms. The effort above will be facilitated with the availability of standards and protocols for advertising and discovering computing resources within fog environments, as well as offloading computation. The OpenFog consortium [382], for instance, though not a standardizing organization itself, works towards this goal.

Autonomic Fog Management. One challenge of the presented use cases is their complexity regarding the system structure and components involved. To be successful, such complexity must not lead to high maintenance costs or come at the expense of usability. Instead, solutions must be able to manage themselves, which implies a degree of autonomy. Similar to the issue of interoperability, fog computing can be both the subject of autonomic management, and also contribute with solutions, for instance by hosting the computation processes necessary for autonomy. This represents both opportunities and challenges for the area of autonomic computing.

Connectivity. The heterogeneity of devices and their communication technologies raise several challenges regarding connectivity. This should be seamless between different solutions, coping not only with mobility but also with existing bitrate and delay constraints. Additionally, networks should be non-intrusive, requiring for instance the sharing of networking resources or infrastructures.

Security and Trust. Fog computing leads to more complex relationships among the system nodes, especially sensor devices and fog computing nodes. Associations between nodes are dynamic. Apart from all security questions relating to privacy of data and safety of patients, this requires some form of trust management between these devices. Though trust models have been applied in various areas, these also need to work with the given complexity and dynamics of fog computing in healthcare.

To fully exploit the fog computing concepts and provide better integrated health applications and their specific requirements, the points above must be considered, both across use case classes and across different deployment scenarios.

H.7 Conclusions

Our review shows that there is a considerable number of computing tasks, across different deployment scenarios and application use cases, that can benefit from fog computing. In fact, our review shows that computation is a necessary element in almost all pervasive healthcare applications, and that these tasks often need to be executed somewhere between the sensors and the cloud. We provided an inventory of such computing tasks, and have shown in which nodes within a network they can be executed. The reviewed papers also show that there is potential for computation at all network levels.

We have further discussed tradeoffs when placing computation tasks in the network, and discussed benefits and challenges of fog computing related to pervasive health applications. Sensor devices are often not powerful enough to do such computation on their own, which is why they need to offload computing tasks. On the other hand, cloud computation is often not a suitable solution for such offloading due to restrictions regarding dependability, privacy concerns or regulations. Fog computing, with its flexibility to add computation as part of a network infrastructure, appears therefore as a suitable concept to meet the requirements of healthcare. Fog computing tasks can filter data, to help preserve privacy or reduce load on the network. The locus of execution can be adjusted to the current deployment scenario, regulations and other requirements. Fog computing tasks can also act as interoperability components, adapting specific sensor needs to standardized and harmonized interfaces. In addition, with their ability to act closely to the users, fog computing tasks add an important component to make systems more dependable. To make these benefits effective, however, it is necessary to lift focus from the individual use cases towards more comprehensive architectures, as discussed above. This review and discussion is a signpost into this direction, summarizing the wide span of deployment scenarios, variety of requirements in future healthcare and the variety of fog computing tasks.

Acknowledgement

David Palma is funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement No. 699924, SINet, Software-defined Intermittent Networking project.

Example of citations

References

[237] Preden, J. S., Tammemae, K., Jantsch, A., Leier, M., Riid, A., and Calis, E. "The Benefits of Self-Awareness and Attention in Fog and Mist Computing". In: *Computer* vol. 48, no. 7 (2015), pp. 37–45.

[288] Topol, E. *The Creative Destruction of Medicine: How the Digital Revolution Will Create Better Health Care*. 2013.

[289] *BioStamp*. Nov. 2016.

[290] Kang, D., Kim, Y.-S., Ornelas, G., Sinha, M., Naidu, K., and Coleman, T. "Scalable Microfabrication Procedures for Adhesive-Integrated Flexible and Stretchable Electronic Sensors". In: *Sensors* vol. 15, no. 9 (Sept. 2015), pp. 23459–23476.

[291] Farandos, N. M., Yetisen, A. K., Monteiro, M. J., Lowe, C. R., and Yun, S. H. "Contact Lens Sensors in Ocular Diagnostics". In: *Advanced Healthcare Materials* vol. 4, no. 6 (2015), pp. 792–810.

[292] Obermeyer, Z. and Emanuel, E. J. "Predicting the Future — Big Data, Machine Learning, and Clinical Medicine". In: *New England Journal of Medicine* vol. 375, no. 13 (Sept. 2016), pp. 1216–1219.

[293] *Bigger Data for Better Healthcare*. Tech. rep. Oct. 2016.

[294] *IntelliVue Telemetry System Infrastructure Installation and Service Guide*. June 2007.

[295] Montenegro, G. E., Kushalnagar, N., Hui, J. W., and Culler, D. E. "Transmission of IPv6 Packets over IEEE 802.15.4 Networks". In: *RFC* vol. 4944 (2007), pp. 1–30.

[296] Stephen, E. *Internet Protocol, Version6 (IPv6) Specification*. 1998.

[297] Alemdar, H. and Ersoy, C. "Wireless sensor networks for healthcare: A survey". In: *Computer Networks* vol. 54, no. 15 (Oct. 2010), pp. 2688–2710.

[298] Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. "Wireless Body Area Networks: A Survey". In: *IEEE Communications Surveys & Tutorials* vol. 16, no. 3 (2014), pp. 1658–1686.

[299] YIN, Y., Zeng, Y., Chen, X., and Fan, Y. "The internet of things in healthcare: An overview". In: *Journal of Industrial Information Integration* vol. 1 (Mar. 2016), pp. 3–13.

[300] Orwat, C., Graefe, A., and Faulwasser, T. "Towards pervasive computing in health care – A literature review". In: *BMC Medical Informatics and Decision Making* vol. 8, no. 1 (2008).

[301] Silva, B. M. C., Rodrigues, J. J. P. C., Torre Diez, I. de la, Lopez-Coronado, M., and Saleem, K. "Mobile-health: A review of current state in 2015". In: *Journal of Biomedical Informatics* vol. 56, no. C (Aug. 2015), pp. 265–272.

[302] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing.* MCC '12. Helsinki, Finland, 2012, pp. 13–16.

[303] *Predix Architecture and Services, Technical Whitepaper.* Tech. rep. General Electric, Sept. 2015.

[304] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. *Mobile Edge Computing - a key technology towards 5G.* Tech. rep. ETSI, Sept. 2016.

[305] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. "The Case for VM-Based Cloudlets in Mobile Computing". In: *IEEE Pervasive Computing* vol. 8, no. 4 (Oct. 2009), pp. 14–23.

[306] Fernando, N., Loke, S. W., and Rahayu, W. "Mobile cloud computing: A survey". In: *Future Generation Computer Systems* vol. 29, no. 1 (Jan. 2013), pp. 84–106.

[307] Perera, C., Talagala, D. S., Liu, C. H., and Estrella, J. C. "Energy-Efficient Location and Activity-Aware On-Demand Mobile Distributed Sensing Platform for Sensing as a Service in IoT Clouds". In: *IEEE Transactions on Computational Social Systems* vol. 2 (2015), pp. 171–181.

[308] Yannuzzi, M., Milito, R., Serral-Gracia, R., Montero, D., and Nemirovsky, M. "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing". In: *19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD).* 2014, pp. 325–329.

[309] Yi, S., Hao, Z., Qin, Z., and Li, Q. "Fog Computing: Platform and Applications". In: *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)* (2015), pp. 73–78.

[310] Deng, R., Lu, R., Lai, C., and Luan, T. H. "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing". In: *IEEE International Conference on Communications (ICC)* (2015), pp. 3909–3914.

[311] Vaquero, L. M. and Rodero-Merino, L. "Finding your Way in the Fog". In: *ACM SIGCOMM Computer Communication Review* vol. 44, no. 5 (2014), pp. 27–32.

[312] Cao, Y., Chen, S., Hou, P., and Brown, D. “FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation”. In: *Networking* (2015), pp. 2–11.

[313] Xu, K., Li, Y., and Ren, F. “An energy-efficient compressive sensing framework incorporating online dictionary learning for long-term wireless health monitoring”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP - Proceedings.* Vol. 2016-May. May 2016, pp. 804–808.

[314] Gia, T. N., Jiang, M., Rahmani, A.-M., Westerlund, T., Liljeberg, P., and Tenhunen, H. “Fog Computing in Healthcare Internet of Things - A Case Study on ECG Feature Extraction.” In: *2015 IEEE Int. Conf. on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)* (2015), pp. 356–363.

[315] Tentori, M. and Favela, J. “Activity-aware computing in mobile collaborative working environments”. In: *CRIWG’07: Proceedings of the 13th international conference on Groupware: design implementation, and use.* Sept. 2007, pp. 337–353.

[316] Catarinucci, L., Donno, D. de, Mainetti, L., Palano, L., Patrono, L., Stefanizzi, M. L., and Tarricone, L. “An IoT-Aware Architecture for Smart Healthcare Systems”. In: *IEEE Internet of Things Journal* vol. 2, no. 6 (May 2016), pp. 515–526.

[317] Needham, M. *Impact of cloud computing on healthcare, Reference architecture, Version 1.0.* May 2016.

[318] Mandellos, G. J., Koutelakis, G. V., Panagiotakopoulos, T. C., Koukias, M. N., and Lymberopoulos, D. K. “Requirements and Solutions for Advanced Telemedicine Applications”. In: *Biomedical Engineering.* Oct. 2009.

[319] Bertini, M., Marcantoni, L., Toselli, T., and Ferrari, R. “Remote monitoring of implantable devices: Should we continue to ignore it?” In: *International Journal of Cardiology* vol. 202 (Jan. 2016), pp. 368–377.

[320] MacIntosh, E., Rajakulendran, N., Khayat, Z., and Wise, A. *Transforming health: Shifting from reactive to proactive and predictive care.* Mar. 2016.

[321] Tanaka, A., Utsunomiya, F., and Douseki, T. “Wearable Self-Powered Diaper-Shaped Urinary-Incontinence Sensor Suppressing Response-Time Variation With 0.3 V Start-Up Converter”. In: *IEEE Sensors Journal* vol. 16, no. 10 (2016), pp. 3472–3479.

[322] *Digital pills make their way to market.* July 2012.

[323] *Philips’ intelligent pill targets drug development and treatment for digestive tract diseases.* Nov. 2008.

[324] Eide, R. “Low energy wireless ECG: An exploration of wireless ECG and the utilization of low energy sensors for clinical ambulatory patient monitoring”. PhD thesis. NTNU – Norwegian University of Science and Technology, June 2016.

[325] Paksuniemi, M., Sorvoja, H., Alasaarela, E., and Myllyla, R. “Wireless sensor and data transmission needs and technologies for patient monitoring in the operating room and intensive care unit”. In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. Jan. 2005, pp. 5182–5185.

[326] Alesanco, A. and Garcia, J. “Clinical Assessment of Wireless ECG Transmission in Real-Time Cardiac Telemonitoring”. In: *IEEE Transactions on Information Technology in Biomedicine* vol. 14, no. 5 (2010), pp. 1144–1152.

[327] Zhang, R., Bernhart, S., and Amft, O. “Diet eyeglasses: Recognising food chewing using EMG and smart eyeglasses”. In: *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. June 2016, pp. 7–12.

[328] Khushaba, R. N., Kodagoda, S., Takruri, M., and Dissanayake, G. “Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals”. In: *Expert Systems with Applications* vol. 39, no. 12 (2012), pp. 10731–10738.

[329] Fettweis, G. P. “The Tactile Internet: Applications and Challenges”. In: *IEEE Vehicular Technology Magazine* vol. 9, no. 1 (Oct. 2016), pp. 64–70.

[330] Steele, R. and Lo, A. “Telehealth and ubiquitous computing for bandwidth-constrained rural and remote areas”. In: *Personal and Ubiquitous Computing* vol. 17, no. 3 (2013), pp. 533–543.

[331] Sametinger, J., Rozenblit, J., Lysecky, R., and Ott, P. “Security challenges for medical devices”. In: *Communications of the ACM* vol. 58, no. 4 (Mar. 2015), pp. 74–82.

[332] Cao, Y., Chen, S., Hou, P., and Brown, D. “FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation”. In: *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*. Aug. 2015, pp. 2–11.

[333] Wac, K., Bargh, M. S., F. Van Beijnum, B. j., Bults, R. G. A., Pawar, P., and Peddemors, A. “Power- and delay-awareness of health telemonitoring services: the mobihealth system case study”. In: *IEEE Journal on Selected Areas in Communications* vol. 27, no. 4 (May 2009), pp. 525–536.

[334] Monteiro, A., Dubey, H., Mahler, L., Yang, Q., and Mankodiya, K. “Fit: A Fog Computing Device for Speech Tele-Treatments”. In: *2016 IEEE International Conference on Smart Computing (SMARTCOMP*. 2016, pp. 1–3.

[335] Craciunescu, R., Mihovska, A., Mihaylov, M., Kyriazakos, S., Prasad, R., and Halunga, S. “Implementation of Fog computing for reliable E-health applications”. In: *2015 49th Asilomar Conference on Signals, Systems and Computers*. 2015, pp. 459–463.

[336] Lopez, G., Custodio, V., and Moreno, J. I. “LOBIN: E-Textile and Wireless-Sensor-Network-Based Platform for Healthcare Monitoring in Future Hospital Environments”. In: *IEEE Transactions on Information Technology in Biomedicine* vol. 14, no. 6 (May 2016), pp. 1446–1458.

[337] Hosseini, M. P., Hajisami, A., and Pompili, D. “Real-Time Epileptic Seizure Detection from EEG Signals via Random Subspace Ensemble Learning”. In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. July 2016, pp. 209–218.

[338] Oladimeji, E. A., Chung, L., Jung, H. T., and Kim, J. “Managing Security and Privacy in Ubiquitous eHealth Information Interchange”. In: *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. ICUIMC ’11. Seoul, Korea, 2011, 26:1–26:10.

[339] Masip-Bruin, X., MarÃn-Tordera, E., Alonso, A., and Garcia, J. “Fog-to-cloud Computing (F2C): The key technology enabler for dependable e-health services deployment”. In: *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. June 2016, pp. 1–5.

[340] Zhang, K., Liang, X., Baura, M., Lu, R., and Shen, X. S. “PHDA: A priority based health data aggregation with privacy preservation for cloud assisted WBANs”. In: *Information Sciences* vol. 284, no. C (Nov. 2014), pp. 130–141.

[341] Hossain, M. S. and Muhammad, G. “Cloud-Assisted Speech and Face Recognition Framework for Health Monitoring”. In: *Mobile Networks and Applications* vol. 20 (2015), pp. 391–399.

[342] Mei, B., Cheng, W., and Cheng, X. “Fog Computing Based Ultraviolet Radiation Measurement via Smartphones”. In: *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. Nov. 2015, pp. 79–84.

[343] Dubey, H., Yang, J., Constant, N., Amiri, A. M., Yang, Q., and Makodiya, K. “Fog Data: Enhancing Telehealth Big Data Through Fog Computing”. In: *ASE BD&SI ’15: Proceedings of the ASE BigData & SocialInformatics 2015*. University of Rhode Island. Oct. 2015.

[344] Nejati, H., Pomponiu, V., Do, T. T., Zhou, Y., Iravani, S., and Cheung, N. M. “Smartphone and Mobile Image Processing for Assisted Living: Health-monitoring apps powered by advanced mobile imaging algorithms”. In: *IEEE Signal Processing Magazine* vol. 33, no. 4 (July 2016), pp. 30–48.

[345] Øyri, K., Balasingham, I., Samset, E., Høgetveit, J. O., and Fosse, E. “Wireless continuous arterial blood pressure monitoring during surgery: A pilot study”. In: *Anesthesia & Analgesia* vol. 102, no. 2 (2006), pp. 478–483.

[346] Chen, H. and Liu, H. “A remote electrocardiogram monitoring system with good swiftness and high reliablility”. In: *Computers and Electrical Engineering* (May 2016), pp. 1–12.

[347] Huang, Y. M., Hsieh, M. Y., Chao, H. C., Hung, S. H., and Park, J. H. “Pervasive, Secure Access to a Hierarchical Sensor-based Healthcare Monitoring Architecture in Wireless Heterogeneous Networks”. In: *IEEE J.Sel. A. Commun.* vol. 27, no. 4 (May 2009), pp. 400–411.

[348] Granados, J., Rahmani, A.-M., Nikander, P., Liljeberg, P., and Tenhunen, H. “Towards Energy-Efficient HealthCare: an Internet-of-Things Architecture Using Intelligent Gateways”. In: *Proceedings of the 4th Int. Conference on Wireless Mobile Communication and Healthcare*. 2014.

[349] Rotariu, C., Manta, V., and Costin, H. "Wireless remote monitoring system for patients with cardiac pacemakers". In: *2012 International Conference and Exposition on Electrical and Power Engineering*. Oct. 2012, pp. 845–848.

[350] *IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks*. IEEE Computer Society, Feb. 2012.

[351] Vauche, R., Bourdel, S., Gaubert, J., Dehaese, N., and Barthelemy, H. "Emitters and Receivers for Impulse Radio Ultra-Wideband and Their Healthcare Applications". In: *2015 IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*. Oct. 2015, pp. 1–5.

[352] Dokania, R. K., Wang, X. Y., Tallur, S. G., and Apsel, A. B. "A Low Power Impulse Radio Design for Body-Area-Networks". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* vol. 58, no. 7 (July 2011), pp. 1458–1469.

[353] Lee, H., Cho, H., and Yoo, H. J. "A 33 μW/node Duty Cycle Controlled HBC Transceiver system for medical BAN with 64 sensor nodes". In: *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*. Sept. 2014, pp. 1–8.

[354] Manchi, P. K., Paily, R., and Gogoi, A. K. "Design and Implementation of Low-Power Digital Baseband Transceivers for IEEE802.15.6 Standard". In: *29th Int. Conference on VLSI Design and 15th Int. Conference on Embedded Systems (VLSID)*. Jan. 2016, pp. 581–582.

[355] Nair, K., Kulkarni, J., Warde, M., Dave, Z., Rawalgaonkar, V., Gore, G., and Joshi, J. "Optimizing power consumption in iot based wireless sensor networks using Bluetooth Low Energy". In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. 2015, pp. 589–593.

[356] Mijovic, S., Cavallari, R., and Buratti, C. "Experimental characterisation of energy consumption in Body Area Networks". In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 514–519.

[357] Chen, C. C., Liu, Y. J., Wen, S. M., Yang, C. C., Chue, J. J., Wu, C. M., and Huang, C. M. "Low-cost electronic dose counter for pressurized metered dose inhaler". In: *2015 IEEE International Conference on Consumer Electronics - Taiwan*. June 2015, pp. 400–401.

[358] Øyri, K. "Feasibility of short-range wireless monitoring in critical care environments". PhD thesis. Aug. 2015.

[359] Gordon, M. S., Jamshidi, D. A., Mahlke, S. A., Mao, Z. M., and Chen, X. "COMET - Code Offload by Migrating Execution Transparently". In: *OSDI* (2012).

[360] Xu, Y., Mahendran, V., and Radhakrishnan, S. "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery". In: *2016 8th International Conference on Communication Systems and Networks (COMSNETS)* (2016), pp. 1–6.

[361] Hu, W., Gao, Y., Ha, K., Wang, J., Amos, B., Chen, Z., Pillai, P., and Satyanarayanan, M. "Quantifying the Impact of Edge Computing on Mobile Applications". In: *the 7th ACM SIGOPS Asia-Pacific Workshop*. 2016, pp. 1–8.

[362] Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H., and Yang, Q. “A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities”. In: *Proceedings of the ASE BigData & SocialInformatics 2015*. ASE BD&SI ’15. 2015, 28:1–28:6.

[363] Aazam, M. and Huh, E.-N. “Fog Computing and Smart Gateway Based Communication for Cloud of Things”. In: *2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud)*. May 2016, pp. 464–470.

[364] Gomes, A. S., Sousa, B., Palma, D., Fonseca, V., Zhao, Z., Monteiro, E., Braun, T., Simoes, P., and Cordeiro, L. “Edge caching with mobility prediction in virtualized LTE mobile networks”. In: *Future Generation Computer Systems* (2016).

[365] Gunawi, H. S., Hao, M., Suminto, R. O., Laksono, A., Satria, A. D., Adityatama, J., and Eliazar, K. J. “Why Does the Cloud Stop Computing?” In: *the Seventh ACM Symposium*. 2016, pp. 1–16.

[366] Byers, C. C. and Wetterwald, P. “Fog Computing: Distributing Data and Intelligence for Resiliency and Scale Necessary for IoT”. In: *Ubiquity* vol. 2015, no. November (Nov. 2015), pp. 1–12.

[367] Davies, N., Taft, N., Satyanarayanan, M., Clinch, S., and Amos, B. “Privacy Mediators: Helping IoT Cross the Chasm”. In: *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications* (2016), pp. 39–44.

[368] Yi, S., Qin, Z., and Li, Q. “Security and Privacy Issues of Fog Computing - A Survey.” In: *WASA* (2015).

[369] Stojmenovic, I. and Wen, S. “The Fog Computing Paradigm: Scenarios and Security Issues”. In: *2014 Federated Conference on Computer Science and Information Systems*. Sept. 2014, pp. 1–8.

[370] Gennaro, R., Gentry, C., and Parno, B. “Non-interactive Verifiable Computing - Outsourcing Computation to Untrusted Workers.” In: *CRYPTO* vol. 6223 (2010), pp. 465–482.

[371] Hummen, R., Shafagh, H., Raza, S., Voig, T., and Wehrle, K. “Delegation-based authentication and authorization for the IP-based Internet of Things”. In: *Eleventh Annual IEEE Int. Conference on Sensing, Communication, and Networking (SECON)*. 2014, pp. 284–292.

[372] Rousseau, F., Duda, A., Damon, L., Guizzetti, R., and Vucinic, M. “OSCAR: Object security architecture for the Internet of Things”. In: vol. 32, no. C (Sept. 2015), pp. 3–16.

[373] Shi, Y., Abhilash, S., and Hwang, K. “Cloudlet Mesh for Securing Mobile Clouds from Intrusions and Network Attacks”. In: *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on*. 2015, pp. 109–118.

[374] Cerf, V. and Senges, M. “Taking the Internet to the Next Physical Level”. In: *Computer* vol. 49, no. 2 (2016), pp. 80–86.

[375] Salman, O., Elhajj, I., Kayssi, A., and Chehab, A. “Edge computing enabling the Internet of Things”. In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 603–608.

[376] Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N., and Dutta, P. “The Internet of Things Has a Gateway Problem”. In: *the 16th International Workshop*. New York, New York, USA, 2015, pp. 27–32.

[377] Zhanikeev, M. “A cloud visitation platform to facilitate cloud federation and fog computing”. In: vol. 48, no. 5 (2015), pp. 80–83.

[378] Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P. “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach”. In: *Future Generation Computer Systems* (2017).

[379] Bibani, O., Mouradian, C., Yangui, S., Glitho, R. H., Gaaloul, W., Hadj-Alouane, N. B., Morrow, M., and Polakos, P. “A Demo of IoT Healthcare Application Provisioning in Hybrid Cloud/Fog Environment”. In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Dec. 2016, pp. 472–475.

[380] Elmisery, A. M., Rho, S., and Botvich, D. “A Fog Based Middleware for Automated Compliance With OECD Privacy Principles in Internet of Healthcare Things”. In: *IEEE Access* vol. 4 (2016), pp. 8418–8441.

[381] *About Continua*. http://www.continuaalliance.org/about-continua. Accessed: 2016-08-30.

[382] *OpenFog Consortium*. http://www.openfogconsortium.org/. Accessed: 2016-11-24.

ISBN 978-82-326-8288-1 (printed ver.)
ISBN 978-82-326-8287-4 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology