Titre: Title:	Designing and Deploying Internet of Things Applications in the Industry: An Empirical Investigation
Auteur: Author:	Mohab Aly
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Aly, M. (2020). Designing and Deploying Internet of Things Applications in the Industry: An Empirical Investigation [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5242/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Designing and Deploying Internet of Things Applications in the Industry: An Empirical Investigation

MOHAB ALY
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie Industriel

Avril 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Designing and Deploying Internet of Things Applications in the Industry: An Empirical Investigation

présentée par **Mohab ALY**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Samuel PIERRE, président
Soumaya YACOUT, membre et directrice de recherche
Foutse KHOMH, membre et codirecteur de recherche
Samuel-Jean BASSETTO, membre
Roch GLITHO, membre externe

DEDICATION

To my parents and my sister
For their endless love, support and encouragement...

ACKNOWLEDGEMENTS

First, I'd like to thank my supervisors, Soumaya Yacout and Foutse Khomh, for their endless encouragement and for pushing further my confidence to finish my Ph.D. I am really thankful for everything I learned from them, for all the opportunities they gave me or pointed me to, for their guidance with their extensive knowledge when I was confused and lost directions, for their patience when I needed them to engage with me, listen, for being honest, with grace, when they had doubts about ideas and–or methodologies, for being not so honest when answering [I do not know] with their aim that I figure it out myself.

Furthermore, I'd like to thank the members of my Ph.D. committee who enthusiastically accepted and dedicated some time to review my thesis. A big thank you for all my friends and colleagues, current and past members of the SWAT, Ptidej, Soccerlab and MCIS teams, for sharing ideas, productive discussions, joy and happiness, and most importantly providing their feedback to have my work improved.

I am really grateful and thankful to my whole family, to my mother, Dr. Ebtesam, who always pushes me hard to pursue my journey with its ups and downs and showing me the right directions even if she is far away from me; to my father, Professor Dr. Hassan, who is ever an exceptional listener, advisor even though it is not his speciality or domain, and always very happy for my success; to my sister, Dr. Menna-T-Alla, who is always there for me no matter what and whenever needed – you are not only a strong woman, you are the strongest tie to the best years of my life; to my brother-in-law, Lieutenant Commander Mohamed Ezz El-din, who continuously advises me to be a better version of myself – thank you for loving my sister and becoming the big brother I have always wanted. I love you all so much!

I also thank my little nephews, Tamim and Rasheed, who were born on January 8th, 2015 and April 14th, 2020, respectively – during my Ph.D. – and gave me a tremendous source of love and energy by their endless smiles and spontaneity. Tamim and Rasheed, you bring me and all of us happiness, joy, and laughter... We all love you little handsome boys.

Last but certainly not least, a big thank you for all who were behind the scenes, who helped me through my entire trip, either academically or professionally, for their unconditional support and valued time, for their patience, and understanding. I'll always remember your kind act of help and I will always remember to pay it forward, like you did with me.

RÉSUMÉ

L'Internet des objets (IdO) a pour objectif de permettre la connectivité à presque tous les objets trouvés dans l'espace physique. Il étend la connectivité aux objets de tous les jours et offre la possibilité de surveiller, de suivre, de se connecter et d'intéragir plus efficacement avec les actifs industriels. Dans l'industrie de nos jours, les réseaux de capteurs connectés surveillent les mouvements logistiques, fabriquent des machines et aident les organisations à améliorer leur efficacité et à réduire les coûts. Cependant, la conception et l'implémentation d'un réseau IdO restent, aujourd'hui, une tâche particulièrement difficile. Nous constatons un haut niveau de fragmentation dans le paysage de l'IdO, les développeurs se complaignent régulièrement de la difficulté à intégrer diverses technologies avec des divers objets trouvés dans les systèmes IdO et l'absence des directives et/ou des pratiques claires pour le développement et le déploiement d'application IdO sûres et efficaces. Par conséquent, analyser et comprendre les problèmes liés au développement et au déploiement de l'IdO sont primordiaux pour permettre à l'industrie d'exploiter son plein potentiel.

Dans cette thèse, nous examinons les interactions des spécialistes de l'IdO sur le sites Web populaire, Stack Overflow et Stack Exchange, afin de comprendre les défis et les problèmes auxquels ils sont confrontés lors du développement et du déploiement de différentes applications de l'IdO. Ensuite, nous examinons le manque d'interopérabilité entre les techniques développées pour l'IdO, nous étudions les défis que leur intégration pose et nous fournissons des directives aux praticiens intéressés par la connexion des réseaux et des dispositifs de l'IdO pour développer divers services et applications. D'autre part, la sécurité étant essentielle au succès de cette technologie, nous étudions les différentes menaces et défis de sécurité sur les différentes couches de l'architecture des systèmes de l'IdO et nous proposons des contre-mesures.

Enfin, nous menons une série d'expériences qui vise à comprendre les avantages et les inconvénients des déploiements 'serverful' et 'serverless' des applications de l'IdO afin de fournir aux praticiens des directives et des recommandations fondées sur des éléments probants relatifs à de tels déploiements. Les résultats présentés représentent une étape très importante vers une profonde compréhension de ces technologies très prometteuses. Nous estimons que nos recommandations et nos suggestions aideront les praticiens et les bâtisseurs technologiques à améliorer la qualité des logiciels et des systèmes de l'IdO. Nous espérons que nos résultats pourront aider les communautés et les consortiums de l'IdO à établir des normes et des directives pour le développement, la maintenance, et l'évolution des logiciels de l'IdO.

ABSTRACT

Internet of Things (IoT) aims to bring connectivity to almost every object found in the physical space. It extends connectivity to everyday things, opens up the possibility to monitor, track, connect, and interact with industrial assets more efficiently. In the industry nowadays, we can see connected sensor networks monitor logistics movements, manufacturing machines, and help organizations improve their efficiency and reduce costs as well. However, designing and implementing an IoT network today is still a very challenging task. We are witnessing a high level of fragmentation in the IoT landscape and developers regularly complain about the difficulty to integrate diverse technologies of various objects found in IoT systems, and the lack of clear guidelines and–or practices for developing and deploying safe and efficient IoT applications. Therefore, analyzing and understanding issues related to the development and deployment of the Internet of Things is utterly important to allow the industry to utilize its fullest potential.

In this thesis, we examine IoT practitioners' discussions on the popular Q&A websites, Stack Overflow and Stack Exchange, to understand the challenges and issues that they face when developing and deploying different IoT applications. Next, we examine the lack of interoperability among technologies developed for IoT and study the challenges that their integration poses and provide guidelines for practitioners interested in connecting IoT networks and devices to develop various services and applications. Since security issues are center to the success of this technology, we also investigate different security threats and challenges across different layers of the architecture of IoT systems and propose countermeasures.

Finally, we conduct a series of experiments to understand the advantages and trade-offs of serverful and serverless deployments of IoT applications in order to provide practitioners with evidence-based guidelines and recommendations on such deployments. The results presented in this thesis represent a first important step towards a deep understanding of these very promising technologies. We believe that our recommendations and suggestions will help practitioners and technology builders improve the quality of IoT software and systems. We also hope that our results can help IoT communities and consortia establish standards and guidelines for the development, maintenance, and evolution of IoT software and systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Internet of Things (IoT) has become prevalent and promises to reshape the manufacturing industry. Its focus is on creating an intelligent network of physical assets, processing through the deployment of information and communication technologies, and exploiting new technologies to serve different sectors.

The current trend of collaborating, distributed teams through the Internet, mobile communications, and autonomous entities, is the first phase of the IoT to develop and deliver diverse services and applications. However, such collaborations are not only threatened by the fragmentation that we are witnessing in the industry nowadays, but also by the conventional security issues for the existing networking technologies as well as the challenges for designing and deploying applications in the presence of the limited resources available on the "things". Simulation and analytical modeling have been recently used to analyze the impact of such threats. Nonetheless, they cannot provide a full "work-as-intended" behavior and–or good results especially when limitations and–or failures exist.

1.1 Basic Concepts

This section presents the main concepts and definitions that are used throughout this document. We begin with the definitions that are seen as the anchor of this document, subsequently, we define the rest of the other concepts to fulfill the reader with all the necessary information that is presented inside.

1.1.1 Internet of Things

The Internet of Things (IoT) refers to a type of network that is able to connect anything with the Internet-based on stipulated standards and protocols, via information sensing equipment, to conduct information exchange and communications to achieve smart recognition, positioning, tracking, monitoring, and administration. Internet of Things (IoT), can be contained in three categories: (1) People-to-People, (2) People-to-Machine/Things, (3) Things/Machine-to-Things/Machine, interacting via the Internet.

The vision of the Internet of Things (IoT) could be seen as a concept and a paradigm that considers pervasive presence in the environment of a number of objects/things that via wireless and–or wired connections, and unique addressing schemes, are able to interact with each other and cooperate with other objects/things to create new applications/services and

reach common goals. In such a context, the research and development challenges to create a smart world are enormous. A world where the real, digital and the virtual are converging to create smart environments that make energy, transport, cities and many other areas more intelligent [1, 2].

1.1.2 Interoperability in Internet of Things

In the last few years, many smart objects/things in the physical world are interconnected and also communicate via the existing internet infrastructure that creates a global network infrastructure called the Internet of Things (IoT). With the substantial solution development for a wide range of devices and IoT platforms, still, each solution provides its own IoT infrastructure, devices, APIs, and data formats leading to interoperability issues within the paradigm. Such issues are the consequence of many serious issues, such as vendor lock-in, impossibility to develop IoT application exposing cross-platform, and–or cross-domain, difficulty in plugging non-interoperable IoT devices into different IoT platforms. Hence, to enable seamless resource sharing between different IoT vendors, efforts should emerge to help IoT interoperability, i.e., the ability for multiple IoT platforms from different vendors to work together [3].

1.1.3 Security in Internet of Things

The Internet of Things (IoT) has an impact on the security and privacy of the involved sectors and stakeholders, i.e., hospitals, cities, grids, organizations, and buildings. Now, various security issues are considered a major problem for a full-fledged IoT environment. There exist lots of security challenges with the number of proposed architectures and technologies that make the backbone of the IoT. Some efficient and promising security mechanisms have been developed to secure the IoT environment, however, there is still a lot to do. The challenges are ever-increasing and the solutions have to be ever-improving. Measures ensuring the architecture's resilience to attacks, data authentication, access control, and client privacy need to be established well in order to address such challenges [4].

1.1.4 Application Development/Deployment in Internet of Things

Application development and–or deployment in the Internet of Things (IoT) is challenging because it involves dealing with a wide range of related issues, such as lack of separation of concerns, and lack of high-level abstractions to address both the large scale and heterogeneity. Furthermore, stakeholders involved in different application activities have to address issues

that can be attributed to different life-cycles phases. When developing/deploying applications, the application logic has to be analyzed and then separated into a set of distributed tasks for an underlying network. Then, the tasks have to be implemented/deployed for the specific hardware device. Apart from handling such issues, they have to deal with other aspects of life-cycle, such as changes in application requirements and deployed devices. A number of approaches have been suggested in the closely related fields of wireless sensor networks, ubiquitous and pervasive computing, and software engineering in general to address the above-mentioned challenges. However, existing approaches only cover limited subsets of the above challenges [5].

1.2 Research Problem Statement and Research Questions

Internet of Things (IoT) is an important emerging technology in a broad range of domains. It is defined as the connection of physical objects and places via the Internet [6], [7]. It defines a technological revolution where physical and virtual objects would be connected to other objects and to the current Internet infrastructure. Consequently, IoT technologies have attracted increasing interest from the research/software development communities. IoT-related discussions have become increasingly prevalent in various domains' question and answer (Q&A) websites, such as Stack Exchange communities. Such Q&A websites moderate thousands of posts each month from IoT practitioners, including developers, with a variety of backgrounds. The ability to analyze and understand such knowledge repositories could provide major insights into the topics of interest and their evaluation to IoT practitioners. Prior works have conducted a wide range of empirical studies on the knowledge in IoT different domains, including programming Q&A websites [8], [9], [10], [11], [12], [13], [14]. Such prior studies provided insights into taxonomies, categories, topics, and trends in various programming Q&A websites, and have uncovered different challenges and concepts thanks to the shared knowledge on these websites.

Most of the previously studied topics and–or technologies via Stack Exchange communities often have long existence before Stack Exchange emerged as a reliable platform for practitioner communication and knowledge sharing. In contrary to those technologies, IoT-related technologies are nascent, thus, we have a promising opportunity for the first time to study how a development community grows its knowledge in fast-paced domains. We also can study how a domain evolves in response to a variety of concerns as captured on the Stack Exchange communities. The ability to identify such discussion topics will help pinpoint the major challenges that IoT practitioners face today.

To address the above challenges, we use the publicly available Stack Overflow data to investigate the most common issues that are/were encountered by IoT practitioners. Knowing what issues the practitioners are facing will help: 1) knowing what are the common issues others have encountered so they can set countermeasures and–or alternative plans for resolutions accordingly, 2) research/software development communities determine avenues for future research (i.e., help us know what real issues that exist are). We apply LDA-based topic models, using Machine Learning for Language Toolkit (MALLET), on the IoT-related Stack Overflow posts to determine what are the practitioners asking for.

To foster the above-mentioned matter, we formulate a number of research questions, inside Chapter 3, and answer them with a comprehensive set of analyses to provide valuable set of information. For instance, we can list the following:

- What topics are covered by IoT-related questions asked on Stack Exchange communities?, and,
- Which IoT-related topics are the most difficult to answer?

We use topic modeling to investigate topics related to the IoT and categorize them to make their comprehension easier. We, also, measure their difficulties in order to capture the attention of the practitioners in their regard, and enhance them to dedicate more time towards their resolutions.

Furthermore, in the past few years, an abundance of IoT initiatives, that include a number of different devices and platforms, has been integrated into a wide range of various applications, such as healthcare, agriculture, utilities, transportation, industrial control, and buildings, etc. Many studies forecast the substantial development of the IoT in the coming years. Those studies are encouraging since they suggest a tremendous impact of the IoT over the coming years. However, a new McKinsey analysis [15] points out a substantial threat and–or challenge to the predicted economic value: *missing interoperability*. In particular, authors state that around 40% of the potential benefits of IoT can be obtained with the interoperability between different IoT systems, i.e., two or more dissimilar systems are able to work together.

The current IoT market is fragmented due to the extreme degree of heterogeneity in terms of device protocols, controllers, network connectivity methods, application protocols, standards, data formats and so on. The absence of interoperability in IoT is due to a lack of standardization [16] [17]. Vendors are intentionally defining various IoT platforms, proprietary protocols, and interfaces that are incompatible with other solutions and–or services. Hence, those vendors create different verticals and mostly closed ecosystems, which are sometimes called silos.

Precisely, components in one silo cannot talk to components in another silo. For instance, before customers can access different IoT things, they generally need dedicated applications for particular things pre-loaded on their smartphones. By this, customers will have a number of devices, each with their own applications, that work independently of each other.

Also, there are other data interoperability issues when developers want to create innovative IoT applications exploiting resources from different IoT applications and–or services in heterogeneous domains, e.g., smart health and smart home. Those issues ultimately lead to vendor lock-in of end-users. Considering the importance of interoperability in IoT, we need to understand interoperability and the existing solutions to analyze what is needed and identify different platforms that are ahead to assist increase the number of interoperable IoT products.

In addition, we discuss a number of integration challenges, inside Chapter 4, and suggest a list of countermeasures for their resolutions. We can list the following parts to have a nature of the discussed issues:

- The integration and interoperability challenges in IoT,
- Interoperability countermeasures to improve the intergration and interoperability in IoT.

We describe the potential interoperability issues that can affect the IoT paradigm, as well as we investigate the solutions and countermeasures to improve such challenges.

On the other hand, security issues such as privacy, access control, secure communication and secure storage of data are becoming significant challenges in IoT environments [18]. Every single device that was created, every new sensor that was deployed, and every single byte that is synchronized within an IoT environment, may at some point, come under security in the course of an investigation.

The fast growth of IoT devices, products, and services has led to the deployment of too many vulnerable and insecure nodes inside the created clusters [19]. Moreover, conventional user-driven security architectures are of little use in object-driven IoT networks [20]. Existing solutions are often not integrated into the entire system, and sometimes they violate the criteria that designers have taken into consideration from the beginning. These are subtle points that are not addressed by designers who tend to focus mainly on functionality and by companies that tend to focus on short term profits. All these reveal the importance of fundamental security solutions and the need for applied security. Hence, we are in need of specialized tools, techniques, and procedures for securing IoT networks and collecting,

preserving, and analyzing residual evidence(s) of IoT environments. We investigate new and unpublished works in the domain of IoT security and forensics.

Therefore, we formulate a number of research questions and challenges, inside Chapter 5, and also propose a list of countermeasures for to address them. We can list the following as a part of issue's comprehension:

- What are the main issues pertaining to IoT security?,
- What solutions are proposed to address such issues?

The aim here is to identify the main security issues faced by IoT ecosystem, and provide a comprehensive list of countermeasures, including their benefits and limitations, to help mitigate them.

Moreover, the advent of Internet of Things (IoT) is changing the way of conceiving information and communication systems. In general, we talk about IoT Cloud to indicate a new type of distributed system consisting of a set of smart devices, e.g., single board computers running lightweight operating systems, interconnected with a remote Cloud infrastructure, platform, or software through the Internet and able to provide IoT-as-a-Service (IoTaaS). In such a context, container-based virtualization is a lightweight alternative to the hypervisor-based approach that can be adopted on new IoT platforms, for enhancingthe IoT Cloud service and–or application provisioning. In particular, considering different IoT application scenarios, container-based virtualization allows IoT Cloud providers to deploy and customize in a flexible fashion pieces of software on the paradigm. However, with the increasingly usage of the IoT domain nowadays, applications started to influence our lives, nonetheless, their deployments became more time-consuming, error-prone, and costly.

The variety of available deployment automation systems also increases the complexity of selecting the most appropriate technology. As a result of the numerous technologies found, the deployment of the IoT applications became a serious challenge that requires immense technical expertise, for instance, devices need to be installed and connected, scripts to be deployed, and backend software has to be provisioned. Due to such a complexity, not knowing the proper deployment approaches for the IoT applications leads to daunting deployment processes. Hence, the container-based virtualization in the perspective of an IoT Cloud scenario should be explored so that the utilized resources and performance costs are analyzed for effective deployments. Particularly, evaluating and exploring the benefits to adopt virtualization techniques in IoT scenarios both in terms of Cloud service management and industrial-business opportunities, and picking up the proper deployment strategy will be of

a benefit for the IoT practitioners, including developers. In addition, overhead introduced by deploying different kinds of applications on top of container virtualization technologies should be highlighted and investigated to select the best technology that suits IoT applications' configurations and deployments.

To cope with the above limitations and selecting the proper way of application deployment, we carry out a number of different experiments with multiple deployment strategies to test and analyze the performance metrics of deploying various types of applications, including serverful and serverless apps, on top of a number of container setups and look deeper into their behaviors. We, finally, provide guidelines to practitioners so that they are good equipped when it comes to choosing the right technology for their IoT applications.

1.3 Research Goals

Scoping the current state of the proposed technologies to address the different challenges that appeared on the horizon recently, such as smart devices integration and interoperability, and enforcing security in modernized IoT systems. Analyzing and understanding those states would provide insights about such topics of interest to the practitioners and help better understand the needs and issues pertaining them as they work in this domain. Besides, drawing a map of the current limitations that both industry and practitioners, including developers, have to pay attention to so that they are able to pursue their strategies with such a technology effectively. Nonetheless, evaluating the effectiveness of different solutions to build efficient applications to provide guidelines to developers on how to select proper deployment scenarios for their several application types. Therefore, in order to achieve such goals, we propose the following objectives and methodologies.

1.4 Research Objectives

The main objective of this thesis is to empirically investigate the design and deployment strategies of the Internet of Things (IoT) applications in the Industry. This will be achieved via investigating the challenges facing IoT practitioners, through a detailed analysis of questions and answers exchanged by practitioners on the popular Stack Exchange communities. Studying the impact of fragmentation, interoperability and security issues pertaining IoT frameworks. Also assessing the performance implications of serverful and serverless IoT application deployments on top of a number of container setups, see figure 1.1. More specifically, the main objective are fulfilled through the following sub-objectives, including:

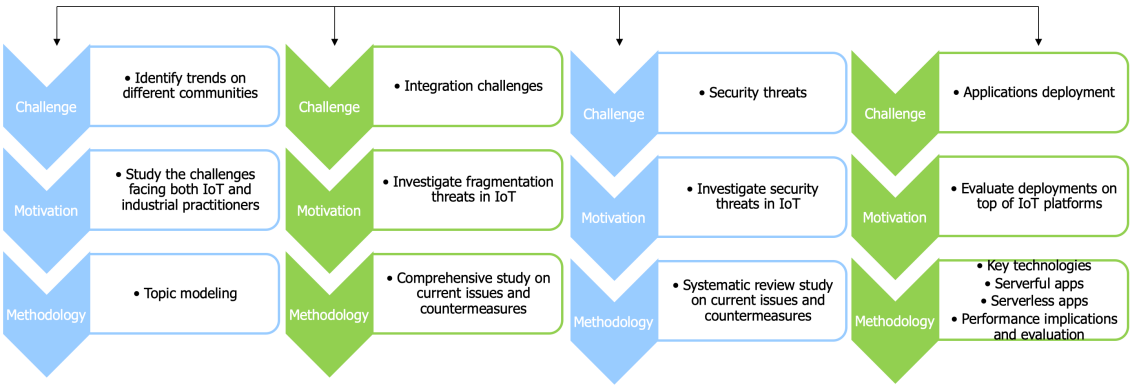


Figure 1.1 *Challenges to be addressed.*

- **What IoT and IIoT questions do practitioners ask *(Chapter 3)*?**

  We undertake an empirical study to find different trends in IoT- and Industrial IoT (IIoT)- related topics. We examine the topics collected to provide insights about their trends and evolution of interest, with the intention to help practitioners and different research/software development communities better understand the needs and challenges of different aspects as they work in the domain.

  Our findings aim to deepen our understanding of issues faced by practitioners when dealing with IoT and IIoT related technologies. The knowledge gained through this study can be leveraged by technology builders and researchers to improve the quality of service of IoT applications and systems.

- **Fragmentation, integration and interoperability issues *(Chapter 4)***

  We examine fragmentation, integration and interoperability issues in IoT technologies and outline some guidelines for researchers and practitioners interested in connecting IoT networks and devices.

  We report different trends, including frameworks and technologies, discuss the integration and interoperability challenges across the different layers of this technology while shedding light on the current IoT state-of-practice. We also discuss some future research directions for the community.

- **Security in Internet of Things systems *(Chapter 5)***

  We describe different security threats and challenges across the different layers of the IoT system's architecture. We also examine some countermeasures proposed in the literature to provide guidelines for researchers and practitioners interested in understanding different IoT security issues.

- **Serverful deployment of IoT applications (*Chapter 6)***

  We conduct an empirical study to evaluate the performance costs of deploying open-source IoT serverful applications. Our results show that there exists discrepancies between performance metrics while considering three different modes:
  (a.) Bare-metal deployments of the open-source IoT platform, (b.) Scaling out the IoT platform and incorporating a messaging-as-a-service infrastructure on top of it, (c.) Adding business applications to the platform. Using these results, developers working to deploy IoT applications can understand the performance implications of their design choices, and select the best deployment strategy for their context.

- **Serverless deployment of IoT applications *(Chapter 7)***

  We conduct an empirical study to examine the performance implications of deploying IoT applications using a serverless infrastructure (i.e., deploying applications as Function-as-a-Service "FaaS") in comparison to deployments performed on an origin server. We investigate whether the deployment of serverless apps, on top of open-sourced IoT frameworks, cause latency issues and–or come with a performance cost. Development teams can leverage our results to select the best deployment strategy for their IoT software systems.

1.5 Proposed Methodologies

First, we conduct a large-scale study on IoT related questions on Stack Exchange forums as we believe that practitioners, including developers, are the best resources to report about different sets of issues due to their hands-on nature of such a framework. Second, we apply methods advocated by the Evidence-based Software Engineering paradigm (EBSE) to systematically and objectively generate and gather evidence about issues related to interoperability and security in IoT, and capture the state-of-practice of IoT technologies in the industry. We study the challenges related to the IoT that have significantly shifted the landscape of Internet-based collaborative services and applications nowadays, nonetheless, we also propose countermeasures to provide guidelines on how such challenges could be tackled. Last but not least, we pick the key technologies used to deploy various kinds of IoT applications and investigate their characteristics (using performance metrics), and limitations, in order to provide guidelines for better application deployments.

1.6 Thesis Overview

In this thesis, we conduct empirical studies on Internet of Things (IoT) different aspects. Our contributions are summarized as follows:

- We analyze Stack Exchange community forums' data to identify what IoT practitioners ask about and what questions tend to be most problematic for them. We aim, with our findings, to assist researchers and practitioners identify such issues by analyzing IoT-related issues on such forums. We are able to naturally and objectively discover the real issues that are being asked, viewed, and faced by a large number of software and research communities; we also establish an approach for finding what people ask on Q&A forums that can be extended to see what is being asked for any other topic and can be useful for researchers to help them get data and information related to their industry of interest.

- Comprehensive analyses of the current capabilities and limitations of the interoperability and security issues and–or challenges of the IoT platform, when handling large data updates from devices using different scaling strategies in the Cloud.We study the history and background of IoT and in regards to objects interconnection and security-based analyses of IoT architecture. We also delve into the associated important technologies related to such an architecture. In addition, we provide taxonomies of various countermeasures and defense mechanisms to provide better handling of the challenges and issues being found; we provide a number of solution spaces and future research directions for the IoT systems.

- We evaluate the challenges, presented by the steps involved in IoT application development and deployment. We delve into a number of different methodologies for IoT application handling, based on techniques presented in the domains of container technologies, such as Kubernetes, OpenShift, and–or Docker Swarms, and model-driven development/deployments. Our contribution is to propose various approaches to reduce time and costs in software development by guiding practitioners into proper deployment scenarios and strategies.

1.7 Related Contributions

The following is a list of our publications related to this dissertation.

Articles in journal

1. **Mohab Aly**, Foutse Khomh, Soumaya Yacout. *What Do Practitioners Discuss about IoT? Characterization and Identification of IoT Categories in Stack Overflow Discussions.* Under revision for submission to Elsevier Special Issue on Machine Learning for Security, Privacy and Trust in IoT.

2. **Mohab Aly**, Foutse Khomh, Soumaya Yacout. *On the Performance Implications of Deploying IoT Apps as FaaS.* Submitted to the Special Issue on Cloud Computing in the IoT Revolution, Internet of Things Journal (IoTJ), Under review as of August 2019, Elsevier.

3. **Mohab Aly**, Foutse Khomh, Mohamed Haoues, Alejandro Quintero, Soumaya Yacout. *Enforcing security in Internet of Things frameworks: A Systematic Literature Review.* Journal: Internet of Things: Engineering Cyber Physical Human Systems, March 2019, Elsevier. DOI:10.1016/j.iot.2019.100050

4. **Mohab Aly**, Foutse Khomh, Yann-Gaël Guéhéneuc, Hironori Washizaki, Soumaya Yacout. *Is Fragmentation a Threat to the Success of the Internet of Things?.* Internet of Things Journal (IoTJ), February 2019, IEEE. DOI:10.1109/JIOT.2018.2863180

Conferences articles

1. **Aly, Mohab**, Foutse Khomh, Soumaya Yacout. "*Kubernetes or OpenShift? Which Technology Best Suits Eclipse Hono IoT Deployments*". Paper presented at the 11th Conference on Service-Oriented Computing and Applications (SOCA), November 2018, IEEE. DOI:10.1109/SOCA.2018.00024

The following publications are not directly related to the material in this dissertation, but were produced in parallel to the research contained for this dissertation.

2. **Aly, Mohab**, Soumaya Yacout, Yasser Shaban. "*Analysis of massive industrial data using MapReduce framework for parallel processing*". Paper presented at Annual Reliability and Maintainability Symposium (RAMS), 2017, IEEE. DOI:10.1109/RAM.2017.7889681

3. Shaban, Yasser, Soumaya Yacout, **Mohab Aly**. "*Condition-based reliability prediction based on logical analysis of survival data*". Paper presented at Annual Reliability and Maintainability Symposium (RAMS), 2017, IEEE. DOI:10.1109/RAM.2017.7889739

Posters and talks

1. **Mohab Aly**, Foutse Khomh, Soumaya Yacout, (2018) "*Kubernetes or OpenShift? Which Technology Best Suits Eclipse Hono IoT Deployments?*", Consortium for Software Engineering Research (CSER), November, 2018, Markham, Canada.

2. **Mohab Aly**, Foutse Khomh, Soumaya Yacout, (2018) "*Kubernetes or OpenShift? Which Technology Best Suits Eclipse Hono IoT Deployments?*", IBM Centers for Advanced Studies Conference (CASCON), November, 2018, Markham, Canada.

1.8 Organization of the Thesis

The remainder of this dissertation is organised as follows. **Chapter 2** provides background information on the different concepts discussed in the thesis, it also surveys related work on IoT- and IIoT- related discussions, studies leveraging topic modeling, study on Stack Overflow & Stack Exchange, interoperability & fragmentation in the IoT frameworks, security affecting the IoT paradgim, serverless deployments and their performance metrics.

Chapter 3 reports about our analysis of IoT- and IIoT- related issues discussed on Stack Overflow & Stack Exchange forums and platforms to capture the current topics of interests to the practitioners as well as research communities. **Chapter 4** discusses the fragmentation issues in the landscape of IoT technologies and the possible proposed solutions to address them. **Chapter 5** reports our analysis of security challenges in IoT systems, it also discusses the possible countermeasures for such challenges. **Chapters 6** and **7** examine a number of deployment solutions for IoT applications over various container technologies and setups. Finally, **Chapter 8** draws conclusions of this dissertation, limitations of the work, and also outlines avenues for future works.

CHAPTER 2 BACKGROUND AND RELATED WORK

2.1 Chapter Overview

This chapter develops the context in which this thesis takes place. It provides the necessary background information and description of the IoT and its architectures, topic modeling, container orchestrators, Eclipse IoT, and serverless computing. In addition, it describes the state-of-the-art which sustains the work in this thesis. This chapter is aimed at readers who are unfamiliar with aforementioned concepts.

2.1.1 Internet of Things

There is no exact definition of IoT yet since it is still in the forming process and is subject to the perspectives taken [21–23]. It was first introduced to the community as a "dynamic global network infrastructure with self-configuring capabilities based on standards and interoperable communication protocols". In IoT, physical and virtual things have identities and attributes that are capable of using intelligent interfaces and being integrated as an information network [24–26]. This concept can be seen as a superset of connecting devices that are identifiable in a unique way by the help of existing near field communications (NFC) techniques [27,28]. When the two words "Internet" and "Things" combine together, this implies an interconnected world-wide network based on infrastructure, communication, networking and information processing technologies [29].

The sensory capabilities of devices have been extended significantly via the emerging wirelessly sensory technology, thus the original concept of IoT is therefore extending to the ambient intelligent and autonomous control. Multiple technologies are involved in the IoT paradigm, such as wireless sensor networks (WSNs), Cloud Computing, RFID, NFC, low energy wireless communications, etc. [30–35]. With the evolution of these technologies, new technologies are brought to the IoT framework [36–43]. Physical things could be identified, accessed and operated via the Internet technology. Depending on various mechanisms for the implementation, the definition of IoT and its context could vary.

In the industrial sectors nowadays, machines are equipped with sensors to help monitor their health and communicate important information about their status to specialized teams distributed around the world, allowing the integration of such information in real time communication to make decisions about the working systems. The next step for those industrial machines is to be integrated in a virtual environment; this enables not only realistic repre-

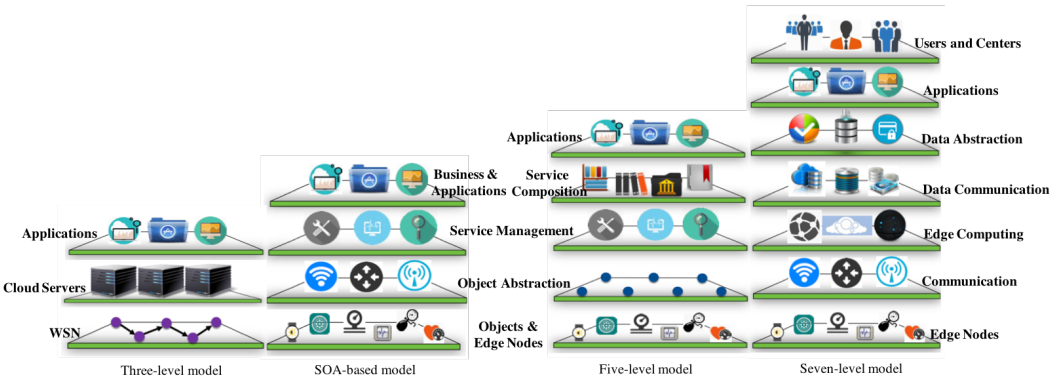


Figure 2.1 *Three, SOA-based, Five and Seven-level Internet of Things reference models*

sentation of the past and present state of the participating machines, but also forecast of possible future scenarios. End-to-end interoperability should be considered to ensure the proper delivery of services regardless of the specifications of the machines used.

Allowing devices for data interaction and cross-platform interoperable communications is an important step towards device collaboration in the digital world. However, there are multi-fold meanings for things communications about hierarchical architecture of IoT-ized systems, including discovery and connectivity, messaging systems and mechanisms, and semantic interoperability. The later factor of things communications is achieved via seamless integration of the underlying protocols and standards.

Architectures for the Internet of Things

Several IoT architectural models have been advanced in recent years, Figure 2.1[1], each of them is focusing on some specific formulations or abstractions of the IoT ecosystem [47]. The three-level model [7] is among the proposed architectures; it depicts IoT as an extended version of wireless sensor networks (WSNs) and models it as a combination of Cloud servers and WSNs which offers different services to users. The service-oriented architecture (SoA) model architecture ensures the interoperability among heterogeneous smart devices in multiple ways [48–50]; it has been suggested to add more abstraction to the IoT architecture [51–53]. The five-level model [6,54], an alternative to the three and SOA-based ones, has been proposed to facilitate the interactions between different sections of an enterprise by decomposing complex systems/services into simplified applications consisting of an ecosystem of simpler and well-defined components.

Recently, in 2014 and 2017 respectively, CISCO and the Open System IoT Reference Model

[1]adopted and customized from: [46]

Table 2.1 *Seven-level Internet of Things model*

	Description
Collaboration and processes	- highest level of IoT model where users reside; users make use of the applications and the analytical data therein.
Applications	- this level provides "information interpretation" where software cooperates with data accumulation and data abstraction levels. IoT applications are numerous and might vary across markets and industrial needs.
Data abstraction	- the opportunity to render and store data is provisioned in this layer; hence further processing becomes easier and more efficient. Common tasks of entities at this level include normalization, denormalization, indexing, consolidating data into one place and providing access to multiple datastores.
Data accumulation	- not all the applications need instant processing; this level enables conversion of data in motion to data at rest, i.e., it allows the storage of data for future analysis or to share with high-level computing servers. Converting the format from network packets to database tables, reducing data via filtering and selective storing and determining whether the data are of interest to higher levels are of the main tasks of this level.
Edge Computing	- this is a level in the model where simple data processing is initiated; this is essential for reducing the computational load in higher levels and providing a fast response as well. Nowadays, most real-time applications need to perform computations as close as possible to the edge of the network; the amount of processing in this level depends on the computing power of each of the servers, computing nodes as well as service providers. Typically, simple signal processing and learning algorithms are utilized in-here. (AKA Fog Computing)
Connectivity	- consists of all the parts and components that enable the transmission of information and commands: (a) communication between devices; (b) communication between the components, and; (c) transmission of the information between the edge devices and computing (edge computing levels).
Edge devices and controllers	- *consists of computing nodes, such as RFID readers, sensors, smart controllers and different versions of RFID tags. Both data confidentiality and integrity must be taken into account from this level upwards.*

(OSiRM) suggested a comprehensive extension to the traditional three, four and five-levels models by introducing their seven-level models that have the potential to be standardized, hence, creating a widely accepted reference model for the IoT paradigm [55]. In such models, data flow is bidirectional in nature, however, the dominant direction of the flow of data depends on the application being worked on. For example, in a control system, data and commands travel from the top of the model (the applications level) to the bottom (edge-node level), whereas in a monitoring scenario, the flow is vice versa (from bottom to top). Tables 2.1 and 2.2 briefly describe the different layers of the two models.

2.1.2 Industry 4.0

Industry 4.0 is a strategic initiative to support the development of the industrial sectors. The main idea is to exploit the potentials of new technologies and concepts, such as availability and use of the Internet and IoT, integration of technical processes and business processes in the companies, digital mapping and virtualization of the real world, and "smart" factory including "smart" means of industrial production and "smart" products, to affect entire industries by transforming the way goods are designed, manufactured, delivered and paid.

Table 2.2 *OSiRM seven-level model*

	Description
Layer 1	- Includes the things that can be subjected to the automation offered by the IoT paradigm. This includes smartphones, surveillance camera, vehicles such as cars and ambulances, as well as utility grid elements. This list is unlimited in its scope.
Layer 2	- Is responsible for data acquisition: it encompasses sensors (appropriate to the thing and the higher-level layer “application"), sensor hubs, embedded devices, etc. Information and data collected include localization data, voice and video (multimedia) and data parameters.
Layer 3	- Fog computing is supported in this layer. Fog networking that is localized (neighborhood or site-specific) is optimized to the IoT users’ operating environments that allow them to use specialized protocols to process their data. It could be wire linked (as in a factory LAN in a robotic application) or wireless (as in wireless LAN, such as infrared links “Li-Fi" or it could be optionally included as well).
Layer 4	- Data aggregation function is supported. Protocol conversion (mapping from a low complexity protocol due to the low power of the sensor to more standard networking protocol) or data summarization (in-network processing [44]) and the edge networking capabilities as well may be entailed. The data aggregation function is carried by a gateway device, whereas edge networking identifies the outer tier of a traditional network, employing renowned networking protocols as well as representing the access tier of the network.
Layer 5	- Supports data centralization function: this corresponds to the traditional core networking functions of modern networks. This layer includes core networks, public/private/hybrid Cloud-oriented connectivity, industry-specific extra-nets, and Internet tunnels. These networks, comprised of infrastructure, entail wired and wireless links and carrier-provided connectivity.
Layer 6	- Encompasses data analytics and storage functionalities.
Layer 7	- Encompasses a vast array of applications or domains, also described as use cases. Here, the list of applications is unlimited in their scope: they include smart grid, building, cities, intelligent transport, surveillance, and sensing including crowd-sensing [45].

This fourth industrial evolution can be best described as a shift in the manufacturing logic towards an increasingly decentralized, self-regulating approach of value creation, enabled by concepts and technologies such as Cyber Physics Systems (CPS), IoT, Cloud Computing or additive manufacturing and smart factories, so as to help organizations meet future production requirements. The comprehensive nature of this definition requires organizations to define what Industry 4.0 means to them.

2.1.3 Topic Modeling

Topic modeling is a statistical technique that can discover topics from a given corpus automatically, without the need for criteria, like tags, training data and–or predefined taxonomies [56, 57]. It uses both word and co-occurrence frequencies in documents to build up a model of related words and texts [10]. In other description, it is that all text is created from words contained in jars representing a specific topic. Hence, we can mathematically discover from which jar a piece of text was assembled. However, the model has no semantic knowledge. By manually examining the keywords in a topic or a jar, we can derive its meaning.

Topic modeling has shown success while being used in other tasks to automatically analyze millions of (un)structured documents, including analyzing news articles [58], identifying topics in the source code [59], and extracting topics from bug reports [60]. The Latent Dirichlet Allocation (LDA) is a topic modeling technique that has been previously used for automated clustering and text mining of Stack Exchange posts [10, 11, 14], by enabling the quantitative analysis to gain more insights into the characteristics and trends in the given dataset. A recent work [61] leveraged LDA as a text mining approach to retrieve the trends in some software research. The technique represents topics as probability distributions over the words in the corpus and documents as probability distributions over the discovered topics. It discovers topics based on how the words sets tend to co-occur in the corpus documents frequently. The words in a discovered topic are semantically related and give meaning to the topics as a whole. Each and every document of the corpus has a mixture of topics. Topics exist across multiple documents, making it possible for the LDA models to discover granular themes that represent the corpus as a whole. Another prior study [62] showed that LDA is biased towards corpus with a larger size. When an approach feeds the LDA multiple corpora at once, the new resulting topics are derived from the larger corpus, since larger corpus dominates the corpora in sheer size.

2.1.4 Latent Dirichlet Allocation

A topic model views a document to be a probability distribution of topics, although a topic is a probability distribution of words. In the configured settings, a document is the text in a post (i.e., body and title), and a topic is a higher-level concept that corresponds to a distribution of words in the text. For instance, we can have a topic "Injection in SQL" when the text has a list of words such as "inject", "sql", "statement", "query", "mysql", etc.

In theory, Latent Dirichlet Allocation (LDA) is a generative probabilistic model that assumes data (i.e., a collection of documents) is generated based on a certain statistical process. Specifically, LDA contains the following three steps.

- Step 1: A topic distribution vector ***theta*** and a term distribution vector ***phi*** are generated by the LDA, based on two Dirichlet distributions [63], respectively.
- Step 2: LDA generates a topic assignment vector $\boldsymbol{z}$ to assign each term in a document to a certain topic according to the topic distribution vector of the document ***theta***.
- Step 3: Each term in a document with the topic distribution vector ***phi*** and the topic assignment vector $\boldsymbol{z}$ is generated by the LDA.

With the repetition of step 1 K times, K topics are generated. Similarly, with the repetition of step 2 and step 3 N times, a document having N terms is generated. With the repetition of steps 1-3 D times, a collection of D documents is generated. Practically, LDA takes a document-by-term (DxN) matrix A as an input, and outputs two matrices B and C, i.e., document-by-topic (DxK) matrix and topic-by-term (KxN) matrix. The document-by-term matrix A can be a term frequency matrix, where A_{ij} represents the number of times that the j-th term appears in the i-th document. In the document-by-topic matrix B, B_{ij} represents the probability of the i-th document that belongs to the j-th topic. In general, a document is viewed as belonging to the topic with the highest probability. In the topic-by-term matrix C, C_{ij} represents the probability that the j-th term belongs to the i-th topic. Likewise, we assign a term to the topic with the highest probability and then we can infer what a topic is talking about by looking up the terms it contains.

To a good extent, the LDA can be viewed as a clustering algorithm. By assigning a specific topic for each and every document using document-by-topic matrix, a cluster of documents can be completed. Specifically, documents assigned to the same topic are grouped altogether. There are a number of off-the-shelf LDA implementations. In this work context, we use a Java package named MALLET[2], which is an implementation based on collapsed Gibbs sampling.

[2]http://mallet.cs.umass.edu/topics.php

2.1.5 Container Orchestrators

Containers are an important building block of application agility. As Cloud applications have evolved from bare metal to virtualized machines, containers, towards serverless computing, the efficiency gains have enabled a wide range of new applications. Organizations have used containers to run long-running services, batch processing at scale, control planes, IoT, and artificial intelligence workloads.

The concept of microservices has arisen in response to the challenges of building large applications that must scale so that they can manage massive workloads, be incrementally upgraded, and remain running on platforms that withstand periodic abnormal behaviors. A microservice-based application is one in which the core functionality has been decomposed into many small, nearly stateless units that communicate with each other via messages and–or events. These atomic units of work are the microservices and they are typically packaged as containers.

Docker

Is a container technology that allows developers to package their application and its dependencies into a bundle. This bundle is, sometimes, also referred to as a container. This container is largely self-sufficient and can be deployed on different platforms. Essentially, the bundle should run in a similar manner wherever it is deployed. This alleviates many problems around implementing new software into production. Often, it could be noticed that developers complain that what has worked fine in development or staging environment, has not worked in production. Many times, the reason for such issues is missing dependencies. Hence, Docker aims at solving such issue.

Kubernetes

Is an open-source platform designed to automate deploying, scaling, and operating application containers. It uses Docker images as a basis to deploy applications into the containers. With Kubernetes, the containers are easily scaled up, destroyed and remade. In comparison with normal virtual machines, they are deployed faster, more efficiently and reliably. Docker creates the image, which is used in Kubernetes. In the world of growing virtualization and the IoT, applications and services need to be deployed as quick and efficient as possible. This is where Kubernetes comes in.

OpenShift

Provides the best experience for developers who are developing, deploying, and running applications. It is a layer on top of Docker and Kubernetes that makes it accessible and easy for the developers to create applications and a platform that is a dream for operators to deploy containers on for both development and production workloads.

2.1.6 Other Orchestration Paradigms

Although this thesis focuses on the above-mentioned container technologies, we want to discuss other types, such as traditional hypervisors and emerging uni-kernels as well to get familiar with the different environments that could be used to manage all the resources for different IoT systems.

Traditional hypervisors: is the part of the private Cloud that has the capability to manage the virtual machines, i.e., it is the part (program) that enable multiple operating systems to share the same hardware. Each operating system could use all the hardware capabilities, e.g., processor, memory, etc. if no other operating system is on. That is the maximum hardware available to one operating system in the Cloud. Nevertheless, the hypervisor is what controls and allocates portions of hardware resources that each operating system should get so that every one of them gets what they need and not to disrupt each other while in operation. There are two types of hypervisors, see figure 2.2[3]:

- Type 1 hypervisor: hypervisors run directly on the system hardware – A "bare metal" embedded hypervisor, such as VMware ESX/ESXi and Oracle VM,
- Type 2 hypervisor: hypervisors run on a host operating system that provides virtualization services, such as I/O device support and memory management, e.g., VMware Server, KVM, and Oracle VM VirtualBox.

Uni-kernels, on the other hand, merge the application with the operating system kernel. Language runtimes, e.g., Python, Node, could be merged with the kernel to have something akin to a container environment to handle application deployments. However, the tooling that is needed to carry on such a task is not there yet. Currently, uni-kernels can add value in creating these small, fast and ultra-scalable services that operate further down the stack compared to a container. So services like load-balancing, firewalling, etc., are considered to be good candidates; a uni-kernel firewall could be deployed with a small-sized memory and outperforming Linux by a noticeable margin.

[3]Image Source: www.ibm.com

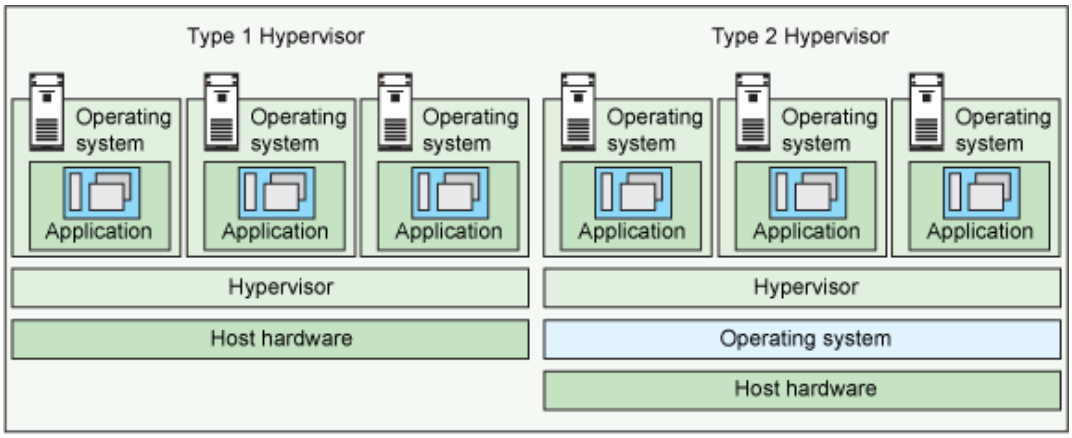


Figure 2.2 *How type 1 and type 2 hypervisors differ.*

2.1.7 Eclipse Hono

Hono provides a platform for scalable messaging in the IoT industry. It introduces a middleware layer between the back-end "micro" services and the devices that are registered within the ecosystem. Thus, communication and networking with back-end services happen via the Advanced Message Queuing Protocol (`AMQP`). If the participating devices speak this protocol, then they can connect directly to the middleware in a very transparent way. Otherwise, Hono, the framework, provides the so-called "protocol adapters" that help to translate the messages from the device's protocol to the `AMQP`. Hence, Hono's core services are decoupled from the protocols that specific applications are using. Through the `AMQP` 1.0 endpoints, the framework provides APIs that depicts two common communications scenarios for devices in the IoT system: (i) Telemetry and Event, (ii) Command & Control, and Registration.

Via Hono's Telemetry and Event Application Programming Interface (`API`), data flows downstream from devices to the back-end, and to a consumer such as Business Application or Device Management component that usually consists of a small set of discrete values, e.g., sensor readings or status property values. Messages are "one-way" directed where devices send such kind of data and usually do not expect a reply from the back-end. Whereas, Hono's Command & Control API allows the sending of commands "requests messages" to devices and expect a reply reception by the back-end component to such commands from devices asynchronously in a reliable way; messages flowing upstream from back-end components like Business Application often represent invocations of services provided by connected devices, e.g., instructions to download and–or apply a firmware update, setting configuration parameters or querying the current reading sensor. Finally, Hono provides APIs for provisioning and managing both the identities and credentials of connected devices.

The platform has different building blocks: 1.) the protocol adapters that are required to connect smart devices that do not have the ability to speak the AMQP natively. Hono comes with two protocol adapters: HTTP-based Representational State Transfer (REST) messages and Message Queuing Telemetry Transport (MQTT). 2) A dispatch router that handles the proper routing of the AMQP messages, between producing and consuming endpoints, within Hono itself. Such router is based on the Apache Qpid project and is designed with scalability in mind so that it can possibly handle connections from millions of devices. As such, it does not take any ownership of the messages being flowing, but rather, passes AMQP packets between different endpoints. This allows a horizontal scaling to achieve reliability and responsiveness. 3) The event and commands messages, that are in a need for a delivery guarantee, can be transmitted and routed through a broker queue; the broker (based on Apache ActiveMQ project) dispatches the messages that need delivery assurance. Typically, such messages originate from the Command & Control API. While devices are being connected to the Hono server components, back-end services are connected via subscribing to certain topics at the Qpid server [64].

To enforce the security of the routed messages, Hono possesses a device registry that is responsible for the registration, activation of devices, provisioning of credentials and an Auth Server to handle the authentication and authorization of the devices. By using an InfluxDB and a Grafana dashboard "Cloud front-end visualization tool", the platform comes also with some monitoring infrastructure to visualize data via a variety of charts and diagrams configured by the users, i.e., in form of time, series, histogram, bar/line charts, stacks, and further customization possibilities. Due to the modular nature of its design, other AMQP 1.0-compatible message brokers than the Apache ActiveMQ Artemis can be used.

2.1.8 Scaling out of Eclipse Hono: EnMasse

EnMasse is an open-sourced "messaging as a service" platform that helps simplify the deployment of a messaging infrastructure on-premise and–or in the Cloud. It provides the scalability and elasticity needed to support the messaging requirements for various IoT use cases. Furthermore, it supports common messaging patterns, that includes (request/reply, competing consumers and publish/subscribe) in addition to the two main protocols: AMQP 1.0 and MQTT. Nevertheless, HTTP support is coming along the road map of the services to be included.

Moreover, this framework provides the possibility of multi-tenancy, in other words, the same infrastructure can be used and shared between various tenants, but are also isolated from each other for security purposes. It enforces security with respect to securing connections

using Transport Layer Security (TLS) as well as authenticating clients using "Keycloak" as the Identity Management System (IMS). EnMasse is completely containerized and runs on key container orchestration setups such as Kubernetes and OpenShift. This aspect makes it appealing to be used with Eclipse Hono. It is considered to be an excellent complement to Hono's microservices architecture and deployment models. It offers all the features needed to be its messaging infrastructure.

2.1.9 Eclipse Che

Eclipse Che is an open-source Java-based developer work-space server and Cloud Integrated Development Environment (IDE), that provides a remote development platform for multi-users purposes. Typically, it can be either utilized by its own IDE browser or directly by connecting to the respective work-spaces that are realized as customized `Docker` containers that bring their complete run-time environment, e.g., an Ubuntu-based installation with Java, Maven and–or a C/C++ toolchain. Different than typical IDEs, the concept of having work-spaces alongside with run-time stacks allows skipping the setup times for end-developers by sharing the proper configurations, e.g., with preloaded example projects and tutorials. In Hono, Eclipse Che has proven to be of a valuable asset for designing and developing various applications and projects, such as IoT workload simulators for different requests targeted towards the Hono platform.

2.1.10 Serverful Computing

Serverful application is a software application that is composed of modules that are not independent of the application to which they belong. Since the modules depend on shared resources, they are not independently executable. This makes it difficult to naturally distribute without the use of specific frameworks or ad-hoc solutions [65, 66]. In other words, the complete distributed application has to be deployed all at once in the case of updates or new service releases. This kind leads to situations where complete applications are simply packaged as on a large virtual machine image. However, depending on the application size, this would involve down-times of the application for end-users and limits the capability to scale the application in the case of increasing or decreasing workloads.

2.1.11 Serverless Computing

A typical usage scenario on a serverless platform consists in writing a Cloud function in a high-level programming language, specifying event(s) that should trigger the running of

the function, e.g., pulling information about the platform being used (i.e., the configuration specs), or loading an image into cloud storage. The serverless system handles everything else from instance selection, scaling, deployment – to fault tolerance, monitoring, logging, security patches, etc. The so-called serverless architecture replaces server administration and operation mainly by using Function-as-a-Service (FaaS) concepts [67] and integrating third-party backend services. FaaS platforms apply time-sharing principles and increase the utilization factor of computing infrastructures in order to avoid expensive always-on components. There are three noticeable distinctions of serverless computing, that can be summarized as follows:

a. Decoupled computation and storage. The computation and storage scale separately. They are provisioned and priced independently. Generally, the computation is stateless and the storage is provided by a separate Cloud service.

b. Executing code without managing resources allocation. Instead of requesting resources, the user just provides a piece of code and the Cloud provisions the resources automatically to execute that code. And,

c. Paying in proportion to resources used instead of for resources allocated. Billing is always associated with the execution, using indicators such as execution time, rather than indicators such as the number and size of the VMs allocated (as it is the case in serverfull computing).

2.2 Literature Review

In the following sections, we survey the most relevant research work related to this dissertation, which includes Internet of Things (IoT) and Industrial Internet of Things (IIoT) related discussions on Stack Exchange communities, issues pertaining the fragmentation, interoperability, security in IoT frameworks, Industrial applications' deployment in the IoT systems, and their performance implications.

2.3 Internet of Things Related Discussions

IoT-related discussions have increasingly become prevalent in various question and answer (Q&A) websites. Such Q&A websites moderate thousands of posts each month from IoT practitioners, including developers, with a variety of backgrounds. The ability to analyze and understand such knowledge repositories could provide major insights into the topics of interest of IoT practitioners.

Prior works have conducted a wide range of empirical studies on the knowledge in different domains, including programming [8–14]. Such prior studies provided insights into taxonomies, categories, topics, and trends in various programming Q&A websites, and have uncovered different challenges and concepts thanks to the shared knowledge on these websites.

Most of the previously studied topics and–or technologies via Stack Exchange communities often have long existence before Stack Exchange emerged as a reliable platform for practitioner communication and knowledge sharing. In contrary to those technologies, IoT-related technologies are nascent, thus, we have a promising opportunity for the first time to study how a development community grows its knowledge in fast-paced domains.

2.3.1 Studies on Stack Exchange Communities

An extensive literature used Stack Exchange to facilitate software engineering tasks. Bajaj et al. [13] studied the common challenges and misconceptions amongst Web developers. Allamanis et al. [8] applied topic modeling on Stack Overflow questions and associated them with programming concepts and identifiers. Li et al. [68] carried out an empirical study with a number of developers, 24, to determine the needs and challenges of developers when performing development tasks. In another work, Nasehi et al. [69] investigated what makes an effective code example via qualitative analysis of Stack Overflow posts. Wang and Godfrey [70] analyzed Android and iOS developer questions on Stack Overflow with the aim to detect API usage obstacles. Furthermore, Azad et al. [9] created rules to be able to predict API calls usage by grouping API calls that are contained in positively voted answer posts on the Stack Overflow platform.

There are other studies that focus on the investigation of the characteristics of Stack Exchange discussions. Treude et al. [12] categorized the questions on Stack Overflow. Barua et al. [10] performed analyses on the textual contents and analyzed both the topics and trends on Stack Overflow. Another work conducted by Rosen and Shihab [11] studied mobile-related questions on Stack Overflow, whereas Yang et al. [14] studied security-related questions. Asaduzzaman et al. [71] analyzed unanswered questions on Stack Overflow and used a machine learning classifier to predict such questions. Other studies proposed the usage of Latent Dirichlet Allocation (LDA)-topic modeling technique to study Stack Exchange discussions, e.g., [10, 11].

There are some recent empirical studies on Stack Overflow [10, 11, 72–74]. Barua et al. conducted an empirical study on all the posts on Stack Overflow [10]. They used LDA to analyze the topics and trends of what developers talk about. Rosen and Shihab narrowed down the research scale by specifically studying mobile-related questions on Stack Overflow [11]. They

also applied LDA to the dataset to investigate the topics mobile developers are interested in. Linares-Vásquez et al. performed an exploratory analysis of mobile development issues using Stack Overflow [73]. They employed topic model to extract the main discussion topics from more than 400K mobile-development related questions. Beyer and Pinzger manually analyzed 450 Android-related posts on Stack Overflow and found that most common question types are "How" and "What" [72]. They also found the dependencies between question types and problem categories. Nadi et al. performed an empirical investigation into the obstacles Java developers face with cryptography API, through triangulating data including top 100 Java cryptography related questions on Stack Overflow [74]. They identified nine main topics related to cryptography and the results suggest that developers do face difficulties using cryptography.

There are many other studies which leverage data on Stack Overflow [13,68,75–83]. Nie et al. suggested a technique that leverages crowd knowledge from Stack Overflow to find the exact tutorial fragments explaining APIs [77]. Jiang et al. proposed a more accurate model, which also leverages crowd knowledge from Stack Overflow, to find exact tutorial fragments explaining APIs [78]. Xia et al. suggested a tool called TagCombine that automatically recommends tags for questions and answer sites, such as Stack Overflow [75]. They found TagCombine has better performance than the state-of-the-art technqiue TagRec. In later work, Wang et al. suggesed another tag recommendation tool named EnTagRec that leverages historical tag assignments [81]. They found EnTagRec imporves the performance of TagCombine further. Bajaj et al. conducted a study on web development related posts on Stack Overflow [13]. They concluded several points about common challenges and misconceptions among web developers. Our work is related to, but different from the above-mentioned studies. In this thesis context, we perform an empirical study on IoT-related posts on Stack Overflow and IIoT-related discussions on Stack Exchange, respectively.

LDA in Software Engineering

Latent Dirichlet Allocation (LDA) infers latent topics to describe text documents. In the LDA model, each document contains a mixture of topics. Topics are allowed to exist across multiple documents, making it possible for the LDA model to discover themes and ideas that represents the corpus as a whole. The number of topics that are found by the model is decided by the user. The larger the number of topics, the more detailed these topics become [84].

LDA has been used to analyze software engineering data. Hindle et al. [85] applied LDA to commit log messages in a version control system to determine topics being worked on by developers at any given time and examining development trends. Neuhaus et al. [86] applied

LDA on a vulnerability database to find the trends in particular security vulnerabilities over time. Garcia et al. [87] suggested a hierarchical clustering algorithm that uses the functionality of LDA to extract concerns from source code identifiers and comments. Advanced information retrieval methods show rather low performance once applied on source code, using parameters and configurations that were applicable and tested on natural language corpora [88]. Barua et al. [10] used LDA to discover main discussion topics on Stack Overflow posts and analyzed the trends over time. Nguyen et al. suggested approaches based on LDA for bug localization [89] and detection of duplicate bug report(s) [90]. Moreover, Rosen and Shihab [11] employed LDA-based topic models to summarize mobile-related questions on Stack Overflow. Yang et al. [14] used LDA-Genetic Algorithms (GA) to cluster security-related questions on Stack Overflow and investigated the popularity and difficulty of the discovered topics.

2.3.2 Studies Leveraging Topic Modeling

Panichella et al. introduced an approach named LDA-GA to address software engineering tasks properly [88]. They used Genetic Algorithms (GA) to search for a good configuration for LDA, which leads to better performance on various software engineering tasks and topics. There are also a number of software engineering conducted studies that have leveraged topic modeling [89–91] to acquire and achieve their functionality. For instance, Nguyen et al. suggested an automated approached named BugScout to help developers reduce buggy code locating efforts by narrowing down the search space of buggy files [89]. They were able to develop a specialized topic modeling to correlate bug reports and the corresponding buggy files through their shared topics. In a later study, Nguyen et al. introduced a novel approach called DBTM, that again leverages topic model to detect duplicate bug reports [90]. Their approach, that merges both information retrieval and topic modeling techniques, has taken the advantages of both IR-based features and topic-based features. Lukins et al. presented a static LDA-based technique for automatic bug localization [91]. Their study shows that the performance of the LDA-based technique is affected neither by the size of the software system nor by the stability of the source code base.

2.4 Interoperability & Fragmentation in Internet of Things Systems

Most popular technologies in IoT, e.g., sensor network technologies, have been the subject of multiple studies. Contributions in regards to IoT protocols standardization have been summarized by Sheng et al. [92]. However, this previous work only focuses on some specific technologies (e.g., the IEEE 802.15.4 standard as well as the details of 6LowPAN work) and

does not investigate interoperability issues.

The essential features and the key capabilities of wireless protocol stack for IoT, such as addressing vital requirements in reliability, efficiency, and connectivity to the Internet, are surveyed by Palattella et al. [93]. This work presents some key technologies for the IoT in the wireless domain, focusing on the IEEE 802.15.4 protocol stack and its pertaining features in support of reliability, efficiency and Internet connectivity. However, other key technologies of IoT, e.g., capabilities of each layer, etc., are not covered. Hence, more investigations are needed to broaden our understanding of IoT challenges. Another work by Gubbi et al. [7] examines potential applications of IoT, technological drivers, challenges, and future research directions, but overlooked key aspects, such as standardization and–or interoperability.

Al et al. [54] examine the technological pillars of IoT, primarily from the perspective of: (a) the technologies employed in wireless communications among a group of IoT nodes, and (b) the interoperable data exchange between the IoT and Internet nodes. Yet, the need for stronger horizontal integration at the IoT above layers is identified, but the contributions of relevant standards addressing the interoperability between nodes are not discussed.

2.5 Security in Internet of Things Systems

There has been a number of research efforts in recent years to cope with and address security threats within the IoT paradigm. Some of the approaches target security issues at a specific layer, whereas, other approaches aim to provide end-to-end security for IoT. A recent study by [20] categorizes security threats in terms of architecture, data, communication, and application. Such suggested taxonomy for security in the IoT is different from the conventional layered architecture. The threats on IoT are then discussed for hardware, network, and application components. Similarly, another survey by [94] discusses and analyses security threats for the protocols defined for the IoT.

The security analyses presented in [95–97] discuss and compare various key management systems and cryptographic algorithms. In addition, similarly, the authors in [98, 98, 99] perform a comparative evaluation of intrusion detection systems. An analytical study of security issues for Fog Computing is presented in [100, 101]. A survey conducted by Sicari et al. [102] discusses research works that provided security, confidentiality, integrity, privacy and access control solutions for IoT and middleware architectures. The authors discuss authentication, trust management, privacy issues, data, and network security as well as intrusion detection systems. For Edge Computing based frameworks including mobile Edge computing, mobile Cloud computing, and Fog computing, the identity and authentication,

access control, trust management, fault tolerance, network security and implementation of forensics are surveyed in [103].

A survey of privacy-preserving mechanisms for IoT is proposed in [104]. The authors describe the secure multi-party computations to be enforced for preserving the privacy of IoT users. Credit checking mechanisms and attribute-based access control are suggested for preserving privacy in IoT. Zhou et al. [105] discuss various security threats and their possible countermeasures for Cloud-based IoT mobile technologies. The authors describe identity and location privacy, node comprising, layer adding or removing, and key management threats for the IoT using Cloud. Another survey [106] discusses major IoT security threats in terms of authentication and authorization, privacy, unique identification of objects, the need for lightweight cryptographic procedures, software vulnerabilities, and malware. The Open Web Application Security Project (OWASP) [107] describes the top 10 vulnerabilities for the IoT architecture. Those vulnerabilities include insecure interfaces of entities of the IoT architecture, physical security, inappropriate security configuration, and insecure software and firmware.

However, to the best of our knowledge, not all aspects of IoT security were addressed in the above-mentioned research efforts. For example, [97] provides a review of network security and identity management, but fails to cover privacy, trust, and resilience. Kozlov et al. [108] addresses both privacy and trust, but not network security, identity management, nor resilience. The conducted study in [109] does not tackle the conventional security aspects as it only considers and focuses on identity management. In this thesis context, we attempt to fill these gaps found in the literature by systematically reviewing security and privacy issues that affect all layers of the IoT architecture. We identify and classify the security requirements of IoT systems and highlight open challenges associated with enforcing their security.

2.6 Industry 4.0 and Its Current State

2.6.1 Industry 4.0 emergence

The industrial sector plays an important role in Europe, serving as a key driver of its economic growth, e.g., job creation, and accounting for 75% of all exports and 80% of all innovations [110]. However, the European manufacturing landscape is twofold. While Eastern Europe alongside Germany shows a constantly growing industrial sector, a number of Western European countries, such as the United Kingdom and–or France have experienced shrinking market shares in the last two decades. While Europe has lost about 10% of its industry share over the past 20 years, many emerging countries managed to double their

share, accounting for 40% of global manufacturing. A few years ago, Germany started to think about some initiatives to maintain and even foster its role as a "forerunner" in the industrial sector. Eventually, the term Industry 4.0 was publicly announced at the Hanover Trade Fair in 2011, presented as part of Germany's high tech strategy so as to prepare and strengthen the industrial sector with regard to future production requirements [111].

While the IoT is assumed to take on a leading role in the Industry 4.0 paradigm, Hermannet et at. [112] [113] found that the Internet of Services (IoS) will find its way into factories, too. Cyber-Physical Systems (CPS), which are able to interact with their environment through sensors and actuators, constitute another element of the Industry 4.0, since they are expected to enable factories to control and organize themselves autonomously in a real-time as well as decentralized fashion [114].
With their capabilities, such factories are often referred to as "smart factories". Given all these concepts, the difficulty of finding a unique and concise definition for Industry 4.0 becomes apparent, and it is surprising that consensus among researchers and practitioners diverge. Furthermore, it is still uncertain how Industry 4.0 will manifest itself in practice and how much time that will take. Regarding a more precise understanding of the topic, we now try to clarify the core components of the Industry 4.0 paradigm.

2.6.2 Industry 4.0 key components

Hermann et al. [112] [113] identified four Industry 4.0 key components based on a surveying academic and business publications, using different a number of databases so that they ensure objectivity. These key components are now briefly described.

Cyber-Physical Systems (CPS): Industry 4.0 is characterized by an unprecedented connection through the internet or other distributed ledgers and so-called CPS, which can be viewed as systems that bring the physical and virtual world altogether []. More precisely, "cyber-physical systems are the integration of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice-versa", [115].
In the manufacturing context, this means that information related to physical and virtual computational spaces are highly synchronized [116]. This allows for a whole new degree of control, surveillance, transparency and efficiency in the production process phase. In regards to their structure, CPS have "two parallel network control, namely a physical network of interconnected components of the infrastructure and a cyber network comprised of intelligent controllers and communication links among them" [117]. CPS realize the integration of such

networks via the use of multiple sensors, actuators, control processing units and communication devices.

Internet of Services (IoS): Sometimes, it is said that we are living in a so-called "service society" nowadays [118]. With respect to that, there are strong indications that, similar to the IoT, Internet of Service (IoS) is emerging based on the idea that services are made easily available via web technologies, allowing enterprises and private users to combine, create and offer new kinds of value-added services [119]. It can be assumed that internet-based services' market-places will hold an essential part in some future industries. On the other hand, from a technological perspective, concepts such as service-oriented architecture (SOA), Software-as-a-Service (SaaS) or Business Process Outsourcing (BPO) are closely related to the IoS context. Barros and Oberle [120] suggest a wider definition of the term service, namely "a commercial transaction where one party grants temporary access to the resources of another party to perform a prescribed function and a related benefit. Resources may be human workforce and skills, technical systems, information, consumables, land, and others".

Smart Factory: Up until now, the IoT and IoS were introduced as core components for the Industry 4.0 paradigm. It is noted that these "concepts" are closely linked to each other, since CPS communicate over the IoT and IoS, hence, enabling the so-called "Smart Factory" which is built on the idea of a decentralized production systems in which "human beings, machines, and resources communicate with each other as naturally as in social network", [121]. The close linkage and communication between products, machinery, transport systems, and humans are expected to change the existing production logic. Therefore, smart factories can be considered another key feature of Industry 4.0. In the smart factory paradigm, products find their way independently via production processes and are easily identifiable and locatable at any time, pursuing the idea of cost-efficient, yet highly flexible and individualized mass production. Authors in [121] note that smart factories "will make the increasing complexity of manufacturing processes manageable for the people who work there and will ensure that production can be simultaneously attractive, sustainable in an urban environment and profitable". Therefore, the potentials that might be associated with smart factories are expected to be huge.

It is important to understand that not only production processes but also the roles of employees are expected to change in a noticeable way. Spath et al. [122] expect that employees to act as decision-makers and–or take on supervising tasks instead of driving forklifts, for

an instant. In the same context, some critics have recently pointed out that the automated and self-regulating nature of the smart factory might cause severe job description. However, hardly any reliable study supports that fear.

Alongside these key components, there is an increasing set of further Industry 4.0 technologies in a broader sense, such as wearable (in form of smart glasses, watches, or gloves), augmented reality applications, autonomous vehicles – including drones – distributed ledger systems, e.g., blockchain, or even big data analytics.

As a preliminary summary, we can define Industry 4.0 as follows:

- The value of networks are controlled decentralized while system elements, such as manufacturing facilities or transport vehicles, are making autonomous decisions (autonomous and decentralized decision making).
- Products and services are flexibly connected through out the internet or other network applications like the blockchain (consistent connectivity and computerization).
- The digital connectivity enables an automated and self-optimized production of goods and services including the delivering without human interventions (self-adapting production systems based on transparency and predictive power).

2.7 Internet of Things in Industrial Applications

IoT in industrial applications are still in its early stage [6, 40, 123]. However, the use of IoT is rapidly evolving and growing. A few applications are being developed and–or deployed in various industries including environmental monitoring, healthcare service, inventory and production management, food supply chain (FSC), transportation, workplace and home support, security and surveillance, etc. Atzori et al. [6] and Miorandi et al. [40] provide a general introduction to IoT in industrial applications in various domains. The design of industrial IoT applications needs to consider multiple goals. Depending on the intended industrial application, designers may have to make a trade-off among such goals to achieve a balance of cost and benefits [124]. Below are some IoT application in industries [16]:

1. Using IoT in the healthcare service industry [125]. IoT provides new opportunities to help improve healthcare [126]. Powered by IoT's ubiquitous identification, sending, and communication capabilities, all objects in the healthcare systems (people, equipment, medicine, etc.) can be tracked and monitored regularly [127]. Enabled by its global

connectivity, all the healthcare-related information (e.g., logistics, diagnostic, therapy, recovery, medication, management, finance) can be collected, managed, and shared in an efficient way. For example, a patient's heart rate can be collected by sensors from time to time and then sent to the doctor's office.

2. Using IoT in firefighting. IoT has been used in the firefighting safety field to help detect potential fire and provide early warning for possible fire disasters. Radio Frequency Identification (RFID) tags and–or bar codes are being attached to firefighting products to develop firefighting product information databases and management systems. Leveraging RFID tags, mobile RFID readers, intelligent video cameras, sensor networks, and wireless communication networks, the firefighting organizations could perform automatic diagnosis to realize real-time environment monitoring, early fire warning and emergency rescue as needed. Recently, Ji et al. [128] illustrate an infrastructure of IoT applications used for emergency management. Their IoT application infrastructure contains sensing, transmission, supporting, platform, and application layers. Their IoT infrastructure has been designed to integrate both local-based and sector-specific emergency systems. In general, establishing standards for implementing Fire IoT is a challenge now.

3. Using IoT for safer mining production. Mine safety is considered as a big concern for many countries due to the harsh working condition in the underground mines. To prevent and–or reduce accidents in the mining fields, there is a need to use IoT technologies to sense mine disaster signals to make early warning, disaster forecasting, and safety improvement of underground production possible [129]. By using RFID, WiFi, and other wireless communications technology and devices to enable effective communication between surface and underground, mining companies can track the location of underground miners and analyze safety data collected from sensors to enhance safety measures. A challenge is that wireless devices need power and could potentially detonate gas in the mine. Further research is needed regarding safety characteriestics of IoT devices used in the mining production.

2.8 Serverless Deployments of Internet of Things Applications

Performance is a key quality attribute of IoT systems [130]. Failing to meet performance requirements often results in reputational and monetary consequences. To mitigate this, practitioners – including developers – routinely conduct performance evaluations [130] with different workloads, e.g., mimicking users' behavior in the field, on software systems [131], and

monitoring relevant performance metrics. These metrics are used to gauge the performance of the systems and identify potential performance issues, such as memory leaks [132], memory allocated consumption and network throughput bottlenecks [133].

Moreover, practitioners using new systems need guidance on how to build such platforms and–or deploy their applications efficiently on top of them. They need to have the know-how to pick up the right configurations and frameworks since the participating devices are resources constrained; devices are not optimal in terms of resources utilization, i.e., CPU, memory, network, etc., and their misuse is likely to significantly degrade the Quality of Service (QoS) as well as User Experiences (UEs). Prior research has suggested a slew of techniques to analyze performance testing results, i.e., performance metrics. Those techniques, typically, examine the following aspects metrics: (a) single performance metric levels and (b) performance metrics relationships.

Analysis of performance metrics levels Malik et al. [134, 135] suggest approaches that cluster performance metrics using Principal Component Analysis (PCA) where each component generated is mapped to its performance metrics by a weight value. Such value measures how far a metric can contribute to the component. For every performance metric, a comparison is conducted on each component's weight value, in order to detect performance regressions.

Nguyen et al. [136] introduced the concept control charts [137] to detect performance regressions. These control charts use a predefined threshold to detect performance anomalies. However, such charts assume that the output follows a uni-model distribution, which may be considered to be an inappropriate assumption for performance. Authors in [136] suggested an approach to normalize performance metrics between heterogeneous environments and workloads to build robust control charts.

Heger et al. [138] presented an approach that uses software development history and unit tests to diagnose the root causes of the performance regressions phenomena. As the first step in their approach, they leveraged Analysis of Variance (ANOVA) to compare system's response time to detect performance regressions. Similarly, Jiang et al. [139] were able to extract response times from systems logs. Instead of conducting statistical tests, they visualized the trend of the response time during the performance tests to be able to identify performance issues.

Relationship between performance metrics Malik et al. [133] leveraged Spearman's rank correlation to capture the relationship between performance metrics. The correlations deviance is measured so that subsystems are pinpointed for maintenance. Furthermore, Foo et al. [140] suggested an approach that leverages association rules to address the limitations of

manually detecting performance regression in large scale software systems. Association rules capture the historical relationships among performance metrics and generate rules based on the results of the prior performance tests. Deviations in the association rules are considered to be signs of performance regressions. Moreover, Jiang et al. [141] used the normalized mutual information as a similarity measure to clusters' correlated performance metrics. Since metrics could be highly correlated in one cluster, hence, the uncertainty among metrics tend to be low. In addition, the authors leveraged the information theory entropy, a logarithmic measure of the rate of transfer of information of a particular message, to monitor the uncertainty of each cluster. A noticeable, i.e., significant, change in the entropy is considered to be a sign of performance degradation.

Analysis of virtual machines overhead Kraft et al. [142] discuss issues pertaining to disk I/O in a virtual environment; they examine disk request–response time performance degradation using a trace-driven approach. Through their analysis, they could identify latency issues in virtual machine I/O requests. Another work conducted by Huber et al. [143] presents a study on Cloud-like environments. They compared the performance of different virtual environments and identified performance degradation. They further categorize the factors that influence the overhead and use regression-based models to evaluate them, however, their modeling was only considering CPU and memory.

Serverless Predictable Performance

Although Cloud functions have a much lower startup latency than traditional VM-based instances, the delays that are incurred when starting new instances can be high for some applications. There are three criteria impacting this cold start latency: (1) the time it takes to start a cloud function; (2) the time it takes to initialize the software environment of the formed function, e.g., load node-information libraries, including underlying platform, CPU counts and Uptime; and (3) application-specific initialization in the code. The latter two can seem small or insignificant in comparison to the former. While it can take less than one second to start a Cloud function, it also might take tens of seconds to load all application and–or function libraries. Another obstacle to serverless predictable performance is the variability in the hardware resources that results from giving the container providers flexibility to pick up the underlying structure and–or architecture.

Many serverless application developers conducted a number of experiments to measure cold-start latency, function instance lifetime, the maximum idle time before shutdown, and CPU usage in Amazon Web Services (AWS) Lambda [144–150]. These experiments were ad-hoc and the results may be misleading because they did not control for the contention caused by

other instances. On the other hand, a few research papers report about measured performance in AWS. Authors in [151], measured the requested latency of AWS Lambda and found it had higher latency than AWS Elastic BeanStalk (a Platform-as-Service system). McGrath et al. [152] conducted preliminary measurements on four serverless platforms and found that AWS achieved better scalability, cold-start latency, and throughput than Google and Azure. Recently, Lloyed et al. [153] investigated the factors that affect application performance in AWS and Azure. They developed a heuristic function to identify the VM as a function that runs on AWS based on the VM uptime in `/proc/stat`. In this thesis context, we perform performance analysis in applications deployment setups, such as Kubernetes, OpenShift, and Docker Swarm, and inspect their resource utilization when having Eclipse IoT-Hono, serverful and serverless application(s) on top of each.

2.9 Chapter Summary

In this chapter, we stated some background information and state-of-the-art on IoT, including various architectures and platforms, container technologies and serverless computing. Furthermore, we investigated literature on different aspects, such as IoT-related discussions on Stack Exchange forums and communities, challenges pertaining interconnection in modern IoT systems as well as the potential security issues related to them and their countermeasures. Moreover, we reviewed a number of works in different IoT applications in the industry, including serverful, their related performance metrics and predictable performance. Finally, we introduced Industry 4.0 and brought on our own definition for it since there is no real consensus on what it means in the meantime.

CHAPTER 3 UNDERSTANDING DIFFERENT IOT AND INDUSTRIAL ISSUES[1]

3.1 Chapter Overview

In this chapter, we perform an empirical study to investigate the challenges facing both IoT and industrial practitioners and establish approaches for analyzing questions being asked on Q&A forums, such as Stack Overflow and Stack Exchange. IoT-related discussions have become increasingly prevalent such communities, analyzing and understanding those discussions could provide insights about the topics of interest to practitioners, including developers, and help the software development and research communities better understand the needs and challenges facing them as they tackle this domain.

3.2 Context

Nowadays, we are witnessing a proliferation of developers' and industrial online communities. Q&A forums like Stack Overflow[2] and Stack Exchange[3] have become prime communication channels for both developers and industrial individuals. They turn to these platforms to share their experiences with different technologies; asking questions about different technical issues experienced during their development activities and sharing their knowledge by answering to the questions of their peers. Through these series of questions and answers, knowledge is created, shared, and curated. Analysing the knowledge contained in these platforms can help understand the challenges and issues faced by practitioners working with different technologies, including IoT and Industry 4.0. Therefore, in this chapter, we mine information contained in such platforms and extract topics related to IoT and Industry 4.0 technologies. We list the main topics pertaining IoT and Industry 4.0, and investigate both their popularity and level of difficulty. Based on the results of our study, we provide a number of recommendations for researchers, educators and practitioners.

[1]Part of the content of this chapter "What Do Practitioners Discuss about IoT? Characterization and Identification of IoT Categories in Stack Overflow Discussions", Mohab Aly, Foutse Khomh, and Soumaya Yacout, is under revision for submission to Elsevier *Special Issue on Machine Learning for Security, Privacy and Trust in IoT*.

[2]http://www.stackoverflow.com/

[3]http://www.stackexchange.com/

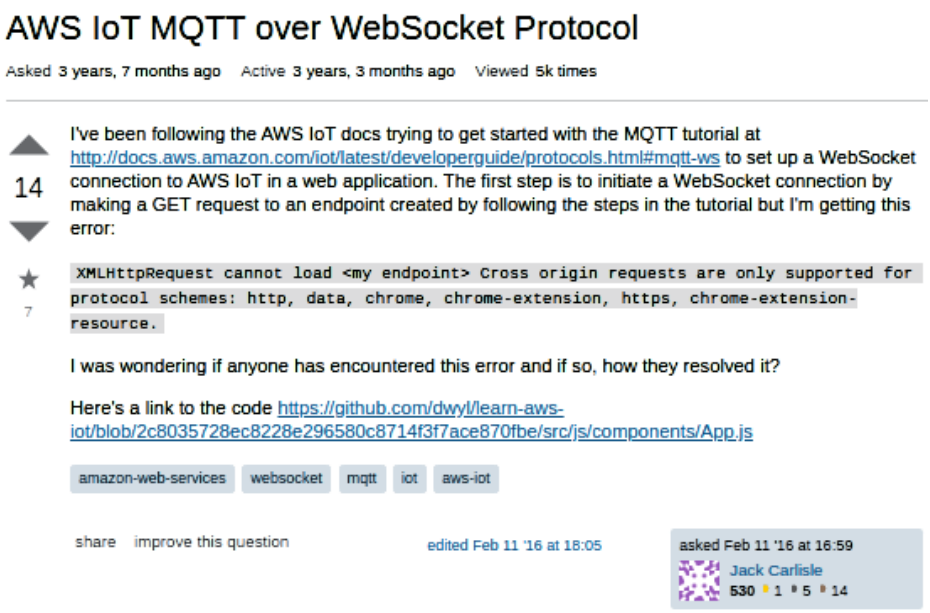


Figure 3.1 *IoT-related post on Stack Overflow*

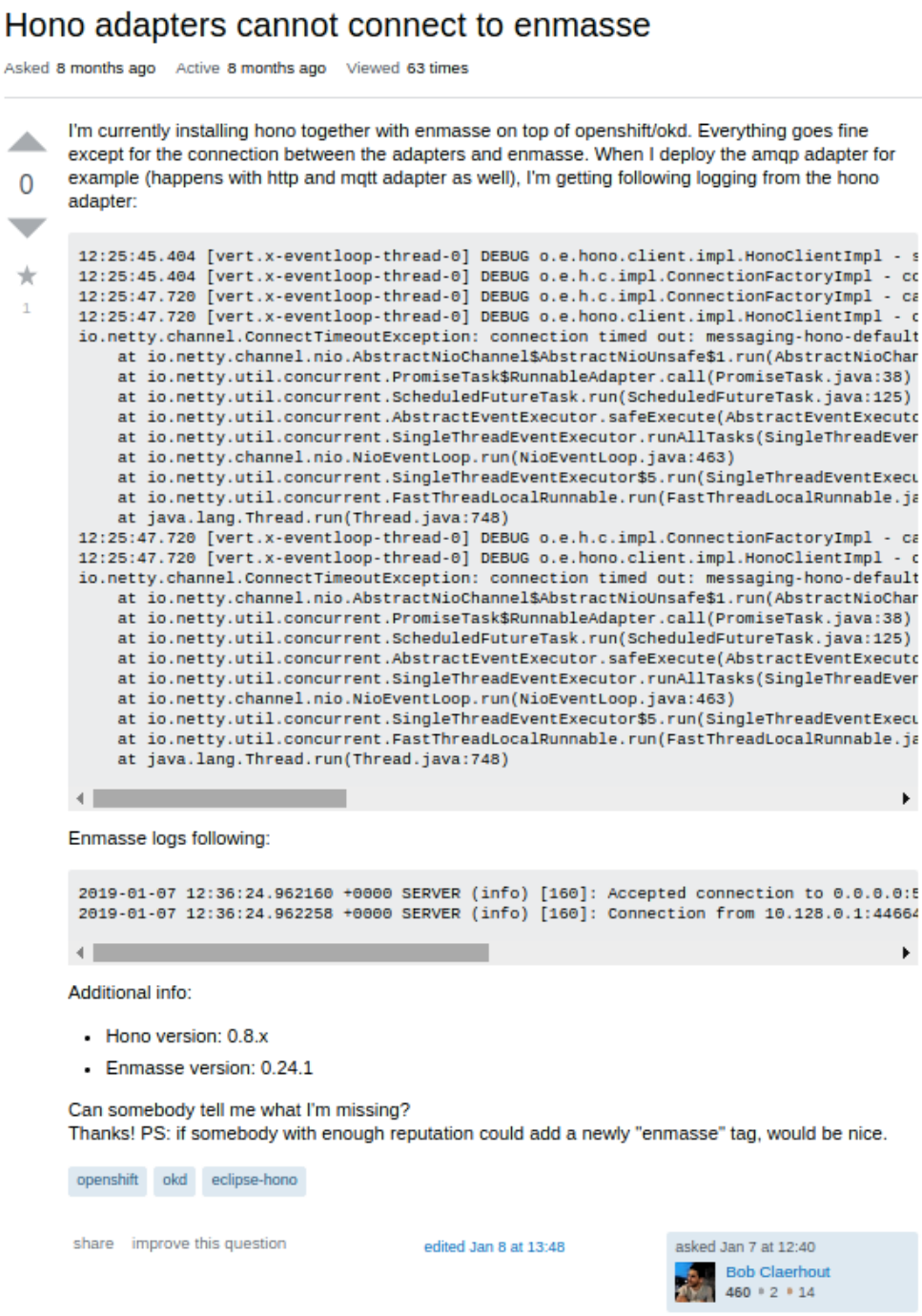
3.2.1 Research Problem and Contribution

Stack Overflow and Stack Exchange are two of the most popular software information sites where practitioners ask and answer technical questions about different kinds of developments, maintenance, and industrial issues. It contains millions of posts which cover a wide range of topics including IoT-related, industrial-related, programming-related, mobile–hardware, and security-related topics. We employ topic modeling techniques to investigate IoT and industrial related questions and determine what popular issues are the most difficult, explore IoT and industry specific issues, and investigate the types of questions that practitioners ask about. Our goal is to identify challenges facing practitioners, including developers and industrial individuals, that require more attention from the research & development, and industrial communities.

3.2.2 IoT-Related Posts

Stack Overflow, i.e., the largest and most trusted online community for developers, consists of millions of posts that cover a wide range of topics, such as mobile-related, programming-related, and IoT-related topics. Due to the importance of IoT nowadays, there is a significant proportion of IoT-related posts.

Figure 3.1 depicts a related IoT question on Stack Overflow. The title of the post is "AWS IoT MQTT over WebSocket Protocol", the tags associated with the post are "stack", "amazon-



Hono adapters cannot connect to enmasse

Asked 8 months ago Active 8 months ago Viewed 63 times

0

1

I'm currently installing hono together with enmasse on top of openshift/okd. Everything goes fine except for the connection between the adapters and enmasse. When I deploy the amqp adapter for example (happens with http and mqtt adapter as well), I'm getting following logging from the hono adapter:

```
12:25:45.404 [vert.x-eventloop-thread-0] DEBUG o.e.hono.client.impl.HonoClientImpl - s
12:25:45.404 [vert.x-eventloop-thread-0] DEBUG o.e.h.c.impl.ConnectionFactoryImpl - cc
12:25:47.720 [vert.x-eventloop-thread-0] DEBUG o.e.h.c.impl.ConnectionFactoryImpl - ca
12:25:47.720 [vert.x-eventloop-thread-0] DEBUG o.e.hono.client.impl.HonoClientImpl - c
io.netty.channel.ConnectTimeoutException: connection timed out: messaging-hono-default
    at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe$1.run(AbstractNioChar
    at io.netty.util.concurrent.PromiseTask$RunnableAdapter.call(PromiseTask.java:38)
    at io.netty.util.concurrent.ScheduledFutureTask.run(ScheduledFutureTask.java:125)
    at io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutc
    at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEver
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:463)
    at io.netty.util.concurrent.SingleThreadEventExecutor$5.run(SingleThreadEventExecu
    at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.ja
    at java.lang.Thread.run(Thread.java:748)
12:25:47.720 [vert.x-eventloop-thread-0] DEBUG o.e.h.c.impl.ConnectionFactoryImpl - ca
12:25:47.720 [vert.x-eventloop-thread-0] DEBUG o.e.hono.client.impl.HonoClientImpl - c
io.netty.channel.ConnectTimeoutException: connection timed out: messaging-hono-default
    at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe$1.run(AbstractNioChar
    at io.netty.util.concurrent.PromiseTask$RunnableAdapter.call(PromiseTask.java:38)
    at io.netty.util.concurrent.ScheduledFutureTask.run(ScheduledFutureTask.java:125)
    at io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutc
    at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEver
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:463)
    at io.netty.util.concurrent.SingleThreadEventExecutor$5.run(SingleThreadEventExecu
    at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.ja
    at java.lang.Thread.run(Thread.java:748)
```

Enmasse logs following:

```
2019-01-07 12:36:24.962160 +0000 SERVER (info) [160]: Accepted connection to 0.0.0.0:5
2019-01-07 12:36:24.962258 +0000 SERVER (info) [160]: Connection from 10.128.0.1:44664
```

Additional info:

- Hono version: 0.8.x
- Enmasse version: 0.24.1

Can somebody tell me what I'm missing?
Thanks! PS: if somebody with enough reputation could add a newly "enmasse" tag, would be nice.

openshift okd eclipse-hono

share improve this question

edited Jan 8 at 13:48

asked Jan 7 at 12:40
Bob Claerhout
460 • 2 • 14

Figure 3.2 *IoT-related post whose tags do not contain "iot"*

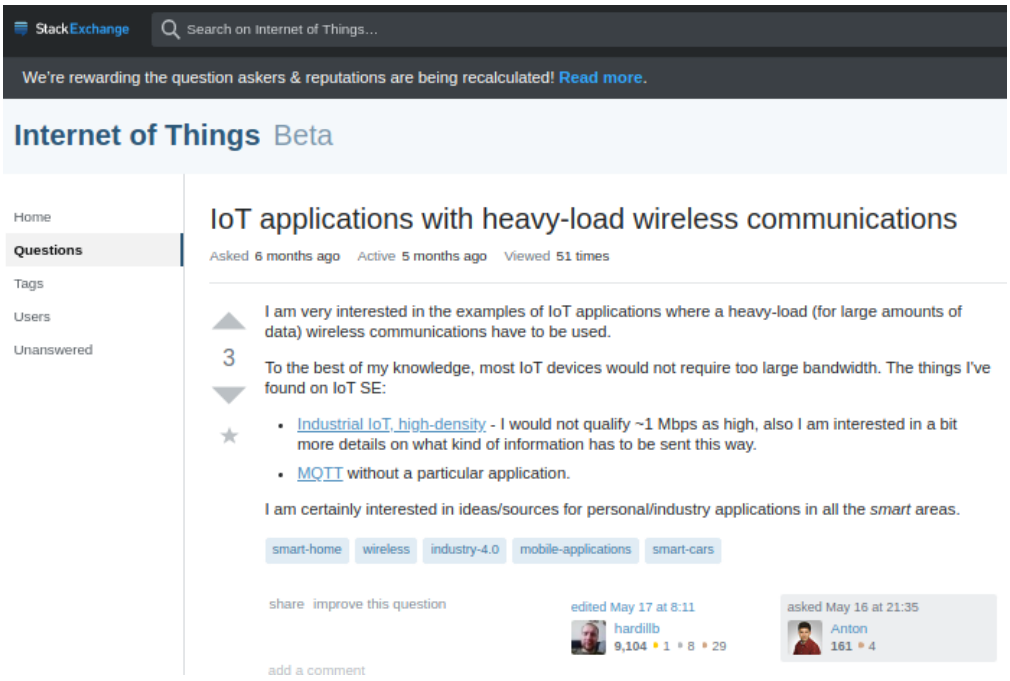


Figure 3.3 *Industry-4.0 related post whose tags contain "industry-4.0"*

web-services", "websocket", "mqtt", "iot", and "aws-iot". Between the title and the tags, the body of the posts exists, describing the question in details. Also, there are numerous information in the margin of the post, such as the number of comments, the edit date, etc.

Note that the tags of the above IoT-related posts contain "iot". However, not all IoT-related questions contain this tag. For example, figure 3.2 shows an IoT-related post whose tags do not contain "iot". Hence, we cannot determine IoT-related posts by simply checking whether the posts contain the tag of "iot", since the extracted posts will not be sufficient and satisfactory. To address this limitation, in this study, we design two heuristics (shown in Subsection 3.3) to extract IoT-related tags, then extract IoT-related posts according to the extracted tags.

3.2.3 Industry 4.0 Related Posts

Similar to Stack Overflow, Stack Exchange comprises 173 Q&A communities to learn and share knowledge on different topics, such as smart-home, smart-cars, definitions for technologies, and Industry 4.0 related topics. Due to the novelty of IIoT and Industry 4.0, it can be expected that a few number of related posts can be retrieved and extracted.

Figure 3.3 illustrates a related Industry 4.0 question on Stack Exchange. Title of the post is "IoT applications with heavy-loaded wireless communications", and the associated tags are "smart-home, wireless, industry-4.0, mobile-applications, and smart-cars".

3.2.4 Research Questions

This chapter answers the following research questions:

- **RQ4.1.: What topics are covered by IoT-related questions asked on Stack Overflow?**
  We use topic modeling to investigate IoT-related topics. IoT-related questions on Stack Overflow cover a wide range of topics. These topics mainly belong to five main categories, i.e., network management, software development, platform development, hardware management and system management. We then mapped them to the integration, security, and applications deployments issues explained in the previous chapter. Among them most questions are about software development, i.e., applications deployments, with $\approx 39\%$.

- **RQ4.2.: Which topics are the most popular among IoT-related questions?**
  We measure the popularity of IoT-related topics by one major metric (i.e., the average number of views), and three minor metrics (i.e., the average number of each of the comments, favourites, and score). The top four most popular IoT topics are "Software Bugs", "Development apps for devices", "Initializing and creating IoT projects", and "Design and implementation of IoT platforms", among which the first two topics are the most valuable since they receive the largest number of comments and favorites, and the highest scores.

- **RQ4.3.: Which IoT-related topics are the most difficult to answer?**
  We measure the difficulty of IoT-related topics by two metrics (i.e., the percentage of questions receiving satisfactory answers and the average time, in days, needed to get an accepted answer). We identified the top most difficult IoT-related topics by considering both percentage and average time needed. "Software bugs" and "design and implementation of IoT platforms", are the two topics that deserve the most attention since they are popular, difficult, and take longer time to be addressed at the satisfaction of practitioners.

- **RQ4.4.: What concerns do Industrial individuals discuss about the new Industry 4.0 trend?**
  Identifying the discussion topics can help practitioners pinpoint the major concerns that Industry 4.0 practitioners are currently experiencing. Industrial researchers could objectively discover real issues. Accordingly, future research efforts can focus on such relevant concerns and–or issues to help support and improve the quality of Industry 4.0 platforms.

- **RQ4.5.: What is the current State-of-Practice of Industrial Internet of Things (IIoT)?**
  Although many IIoT projects are still in the proof-of-concept or trial stage, there are clear signs of a widespread move towards full production deployments. Relevant technologies are available to help achieve good returns, but still we need to investigate more into that.

3.3 Study Design

We now describe the extraction and processing of Stack Overflow and Stack Exchange data for our experiment. We first present the details of the data collection in Subsection 3.3.1, and then we elaborate our experimental approach in subsequent Sections and Subsections.

3.3.1 Data Extraction and Processing

Data Collection

Stack Overflow and Exchange are similar to most Q&A sites allow users to post questions, post answers to questions, comment on posts, and vote on posts. When users create a post, they are allowed to tag them with their subject area to make it easier for others to find the post. For instance, a question asking about using custom tool in implementing applications is tagged with iot or industry-4.0, implementation, tool, and deployment. Each question is required to have at least one tag and may contain up to five different tags that represent their subject area.

To perform our study, we started by downloading and extracting Stack Overflow and Stack Exchange data dumps in CSV formats using the below steps:

- Step 1: Getting the questions with <iot> and <industry-4.0> tags.

- Step 2: Executing the following query-scripts to get the questions, and export the CSV file.

```
[Step 2:]  #standardSQL in Stack Overflow (<iot-tag>)
Select Id, ParentId, PostTypeId, Score, CreationDate, OwnerUserId, Body
from Posts where PostTypeId=1 and Tags like '%<iot>%' '
union
Select Id, ParentId, PostTypeId, Score, CreationDate, OwnerUserId, Body
from Posts where PostTypeId=2 and ParentId in (select Id from Posts
where PostTypeId=1 and Tags like '%<iot>%')
```

```
[Step 2:]  #standardSQL in Stack Exchange (<industry-4.0-tag>)
Select * from Posts where Posts.tags like '%industry-4.0%')
```

- Step 3: Executing the following query-script to get the comments and answers from the extracted questions, and export the CSV file.

```
[Step 3:]  #standardSQL in Stack Overflow (<iot-tag>)
Select Id, PostId, 3, Score, CreationDate, UserId, Text from Comments
where PostId in (Select Id from Posts where PostTypeId=1 and Tags like
'%<iot>%'
union
Select Id from Posts where PostTypeId=2 and ParentId in (Select Id from
Posts where PostTypeId=1 and Tags like '%<iot>%')))
```

```
[Step 3:]  #standardSQL in Stack Exchange (<industry-4.0-tag>)
Select * from Posts answer,Posts question where answer.PostTypeId=2 and
question.PostTypeId=1 and answer.ParentId=question.Id and question.tags
like '%industry-4.0%')))
```

- Step 4: Generating the corpus of the desired format.

The extracted data dump from Stack Overflow for IoT-related discussions contained 176,808 posts including different questions and answers, spanning from 2012 to 2019. Each discussion includes a question and zero or more answer posts and their metadata (e.g., body, creation date and number of votes). Whereas 11 discussions were extracted on Stack Exchange for Industry 4.0 discussions. A detailed information on a Stack- Overflow/Exchange post is presented in the following table 3.1. Questions are typically tagged with terms describing the categories under which these Q&A discussions are grouped. During the next processing step, we extract the relevant discussions for a study context. We focus on IoT- and Industry-4.0 related topics, and for each post, they include body and several metadata. Since a prior

work [69] suggests the use of highly voted answers, we focused on IoT and industry-4.0 related answers that have high number of votes by users.

We traverse the dataset to lookup for the question posts whose tag contains the term "iot", "iot-related", or "industry-4.0". However, those posts are far from sufficient; some posts may not have that tag even if they are talking about the same topic; since "iot" and–or "industry-4.0" are very general terms and the tags of a post may be in much finer granularity. Hence, we need to carefully extract some other tags that are related to it.

In the second step, we extract the tags from all of the posts extracted; we refer to such tags as candidate tags – and from them we extract tags related to "iot" and "industry-4.0". For each candidate tag t, we count three values, i.e., a, b and c. Specifically, a denotes the number of questions posts whose tags contain t within all the posts extracted in the first step (i.e., the number of all question posts whose tags contain t, "iot", and "industry-4.0"). b denotes the number of questions posts whose tags contain t within all the posts in the original dataset (i.e., the number of all question posts whose tags contain t). Based on a and b, we are able to calculate the first value of $H1$ = a/b, which indicates to what extent the tag t is exclusively related to "iot" and "industry-4.0". The value of $H1$ ranges from 0 to 1. The larger the value of H1 is, the more exclusive relation tag t has to be "iot" and–or "industry-4.0". If the value of $H1$ is equal to 1, it means that tag t only appears together with "iot"–"industry-4.0" in the tags of the posts (i.e., the most exclusive). We can filter the tags by setting a threshold $Thre1$. For instance, given Thre1 set to 0.1, a tag that appears together with "iot"–"industry-4.0" in less than 10% of all the questions whose tags contain the tag will be removed.

However, using the aforementioned heuristic $H1$ by itself to extract the tags would cause another problem. Suppose that a tag only appears in one post within the whole dataset and the post happens to be related to "iot" and–or "industry-4.0". In this case, the tag is so specific to an issue that is not representative to "iot"–"industry-4.0", although the value of the $H1$ is equal to 1. Hence, we also want to filter such kind of tags. We denote c as the total number of question posts extracted in the first step (i.e., the number of all question posts whose tags have the value "iot"). Based on a and c, we, then, can calculate the second value of $H2 = a/c$, that solves well the above issue if we set a second threshold $Thre2$ to filter the tags once more. For instance, given $Thre2$ set to 0.01, a tag that appears in less than 1% of all the questions whose tags contain "iot"–"industry-4.0" will be removed.

By adjusting $Thre1$ and $Thre2$, we can get several tags that are representative to "iot"–"industry-4.0". Table 3.2 depicts various tag sets related to "iot" and "industry-4.0" that we extracted using different threshold setups. In the following text, the threshold setup (0.1,

Table 3.1 *Detailed information about a Stack Overflow/Exchange post*

Name	Description
Id	- Id of the post
PostTypeId	- *Type of a post: 1 represents a question post, and 2 represents an answer of that post*
AcceptedAnswerId	- Id of the corresponding accepted answer post for the identified question post (optional, and appears only when PostTypeId==1)
ParentId	- *Id of the corresponding question post for the answer post (optional, and appears only when PostTypeId==2)*
CreationDate	- Creation date of the post
Score	- *Average score by the viewers of the post*
ViewContent	- Total number of views for the post (optional, and appears only when PostTypeId==1)
Body	- *Body of the post*
OwnerUserId	- Id of the owner of the post (optional)
OwnerDisplayName	- *Username of the owner of the post (optional)*
LastEditorUserId	- Id of the person who last edited the post
LastEditorDisplayName	- *Username of the person who last edited the post*
LastEditDate	- The date when the post was last edited
LastActivityDate	- *The date when the status of the post was last changed*
Title	- Title of the post (optional, and appears only when PostTypeId==1)
Tags	- *Tags of the post (optional, and appears only when PostTypeId==1*)
AnswerCount	- Number of answers for the post (optional, and appears only when PostTypeId==1)
CommnetCount	- *Number of comments for the post*
FavoriteCount	- Number of people who liked the post (optional, and appears only when PostTypeId==1)
CloseDate	- *The date when the post was closed (optional)*

0.01) is the default setup [14], and the corresponding result is the default tag set we use (i.e., the first row in table 3.2) to find the iot-related posts. In the last step, we traverse the dataset again to retrieve the questions posts whose tags include at least one of the tags in the tag set. We use such questions and their corresponding answer posts to make our analysis accordingly.

3.3.2 Data Analysis

We analyse the extracted data using feature extraction and topic modeling. In the feature extraction phase, we extract a number of features from the posts. Those features are selected as representative terms that are considered to be useful to build a good topic model. In our study, we opt to use the term frequency as features. Afterwards, we build a topic model with the extracted features using LDA tuned using MAchine Learning for LanguagE Toolkit (MALLET). MALLET is used to determine the optimal number of topics and LDA clusters various IoT posts into a number of different topics based on their corresponding topics.

Table 3.2 *Different tag sets "iot"/"industry-4.0" extracted by different threshold setups*

Threshold Setup (Thre1, Thre2)	Tag Set	Number of Tags
(0.1, 0.01)	iot	2,672
(0.1. 0.001)	windows-10-iot-core	754
(0.15, 0.001)	aws-iot	664
(0.15, 0.005)	azure-iot-edge	246
($\geq$ 0.2/0.25, 0.01/0.005)	watson-iot	185
($\geq$ 0.3, 0.01/0.005)	google-cloud-iot	111
(0.1, 0.01)	industry-4.0	11

Feature Extraction

As previously mentioned in Subsection 3.3.1, a question post includes title, body and several other metadata. To cluster the posts, we ought to build a corpus in which each row is a text for a post. In turn, for each post, we combine both title and body to form the final text, and then we pre-process the texts in four main steps.

- Step 1: we remove all the code snippets (that are enclosed in $< code >$ tag) in the text, since Barua et al. showed that code snippets do not help topic models [10].
- Step 2: we remove all the HTML tags such as $< p >$ and $< /p >$ since they do not have useful information for the topic model.
- Step 3: we remove the stop words, numbers, punctuation marks and other non-alphabetic characters since they add little value to the topic.
- Step 4: we use the Snowball stemmers [154] to transform the remaining terms to their root forms (e.g., “reading” and “reads” are reduced to “read”) in order to reduce the feature dimensions and unify similar words into a common representation.

After following the aforementioned steps, we compute the appearance frequency in all the posts for each stemmed term. To further reduce the noise, we sort all the stemmed terms based on their total term frequency and discard the terms that appear less than 10 times. The remaining 4,633 different "iot" and the 11 different "industry-4.0" stemmed terms are the final features that we extract[4]. We count the times of appearance for each term in each post and form a term frequency matrix $\mathbf{m}$. Specifically, w_{ij} denotes the number of the times the j-th term appears in the i-th post.

[4]Replication data package are shared online at osf.io scientific data repository: https://osf.io/yuzn4/

LDA Tuned using MALLET

We use LDA to group IoT posts into different topics. In LDA, the number of topics K is an undetermined, but important parameter. An overly large or overly small value of K may affect the performance of our approach in a very severe way. Hence, we use an adopted LDA technique, tuned using MALLET, to look for an optimized value of K. MALLET provides a simple way to analyze large volumes of unlabeled text. Within MALLET, a topic consists of a cluster of words that frequently occur together. Using contextual clues, it can connect words with similar meanings and distinguish between uses of words with multiple meanings. Further, MALLET package includes an extremely fast and highly scalable implementation of Gibbs sampling, efficient methods for document-topic hyperparameter optimization, and tools for inferring topics for new documents given trained models.

- Importing Documents: Importing files into MALLET's internal format. The following instructions assume that the documents to be used as input to the topic model are in separate files, in a directory that contains no other files.
  *bin/mallet import-dir -input /data/topic-input -output *
  topic-input.mallet -keep-sequence -remove-stopwords

- Building Topic Models: Once documents have been imported, we can use the `train-topic` functionality to build a topic model.
  *bin/mallet train-topics -input topic-input.mallet -num-topics 100 *
  -output-state topic-state.gz

  ––*num-topics [NUMBER]* The number of topics to use. The best number depends on what is being looked for in the model. The default (10) will provide a broad overview of the contents of the corpus. The number of topics should depend, to some degree, on the size of the collection.
  ––*num-iterations [NUMBER]* The number of sampling iterations should be a trade off between the time taken to complete sampling and the quality of the topic model.

- Hyperparameter Optimization
  ––*optimize-interval [NUMBER]* turns hyperparameter optimization on. Allows the model to better fit the data by allowing some topics to be more prominent than others.

- Model Output
  ––*output-model [FILENAME]* specifies a file to write a serialized MALLET topic trainer object. Such type of output is appropriate for pausing and restarting training,

but does not produce data that can easily be analyzed.
−−output-state [FILENAME] Similar to output-model, it outputs a compressed text file containing the words in the corpus with their topic assignments.
−−output-doc-topics [FILENAME] specifies a file to write the topic composition of documents. *−−output-topic-keys [FILENAME]* contains a "key" consisting of the top k words for each topic. This output can be useful for checking that the model is working as well as displaying results of the model. In addition, this file reports the Dirichlet parameter of each topic. If hyperparamter optimization is turned on, this number will be roughly proportional to the overall portion of the collection assigned to a given topic.

3.4 Study Results

This section presents, analyzes, and discusses the results of our research questions. For each question, we discuss each of the motivation, the approach designed to answer the question, and the findings.

- **RQ4.1.: What topics are covered by IoT-related questions asked on Stack Overflow?**

Motivation. The interest to use the IoT has grown much the past few years. We aim at understanding the topics that are covered by IoT-related questions asked on Stack Overflow. This research question is important to identify topics that are important to IoT practitioners, and to understand the challenges that IoT developers face.

Approach. Since there is a large number of IoT-related posts, we use topic modeling as a way to summarize them. Topic modeling is one technique that has been successfully applied in the past to summarize large corpora in many different fields including software engineering [10, 155]. We use an adapted topic model, LDA tuned using MALLET, to do the clustering of the IoT-related questions and to discover the issues that are being asked on Stack Overflow. In general, LDA needs a predefined number of topics K and it has different optimal values of K for different problems. The approach we are following can automatically determine a "near" optimal value of K based on the characteristics of a specific problem so that LDA can achieve better results. LDA is a statistical topic modeling technique, which means topics are represented as a probability distribution over the words in the corpus [10]. To better analyze the question posts in our corpus, we add their dominant topic as determined by the LDA topic model as an additional attribute.

Results. By using our approach, we group IoT-related questions into 100 topics. Tables 3.3

Table 3.3 *Topic names and related top 20 key terms for top 100 topics-(1)*

NO.	Topic Name	(After stemming) Key Terms
1	Message rate configurations, limits and throttling in Cloud platforms (network management)	time number seconds limit performance size maximum messages long single max minute rate interval amount count day window connections
2	SetRetryPolicy method issue (software development)	alljoyn module.js message double deviceid messagestring mac_address telemetrydata currenttemperature echo minhumidity minitemperature myweather raod,nextdouble raod,deviceclient currenthumidity messageid temperature
3	Java Publish and Subscribe API reference for realtime apps (software development)	pubnub channel sbidots object TRUE jun objects class callback chart static console single stalk asyncio msg published publish
4	Message routing configuration for Azure IoT-hub (network management/platform development)	hub event azure messages function iot service events bus queue endpoint trigger custom route send partition hubs routing
5	Mosquitto configuration file for network operation (network management)	mosquitto broker client bridge set connect mosquitto_pub listener TRUE file option ths mqtt clients support FALSE mosquitto_sub running options default
6	Telegram bot library for Arduinio devices (software development)	end processing function hta verbose noderect heap functions window list print bytes random ibm.iot android.iotstarter esp npm build
7	Modbus TCP/IP for real-time automation (network management/(automation))	ope class interface unit test factory command_model member modbus_plc methods servers tests commands tick printer type classes types
8	IoT based temperature and humidity monitoring over ThingSpeak using hardware devices (hardware management)	sensor temperature data sensors values humidity send temp read reading dht raspberry code thingspeak field email sending readings case channel
9	MQTT essentials: publish ,subscribe and unsubscribe (network management)	topic mqtt message messages publish broker subscribe client topics qos published publishing send receive sending payload subscribing subscribed received subscription
10	Sending commands to devices – configuring devices (hardware management)	devices iot connected send connect register connection status information commands type specific unique create api case command deviceid registered
11	Dealing wiht Watson IoT platform using Node-RED (platform development)	iot ibm node bluemix watson platform node-red service application foundation dashboard event comment device org events cloud flow command organization
12	Writing WebSocket servers (network management)	server client web send connection request http connect clients socket php side browser data running tcp requests page websocket message
13	Connecting Android devices to a wireless network (network and hardware management)	wifi network connection internet connect connected ssid android portal wi-fi profile password access networks wireless hotspot mobile point case val
14	Developing a smart contract with the IBM Blockchain platform (platform development)	driver blockchain usb func extension err lightning nil spring controller write state java chaincode kernel bsod drivers fabric transaction behavior
15	HTML script tag and methods (software development)	script html type div var function class page title body src namespace head usr javascript text/javascript href component style form
16	Creation of a self-signed SSL certificate (network management/(security))	certificate key certificates cert private openssl root ssl keys secure public tls signed connect certs rsa generate authentication security created
17	IoT-hub feedback issue – rejection – after calling CompleteAsync() function (system management)	method direct message serviceclient object invoke response methods deviceclient call null cloud invoking called feedback ledpinstate deviceid failed request moduleclient
18	Referencing .NET standard assemblies from both .NET core and .NET framework (software development)	net core project version windows arm package uwp library dll nuget asp.net win assembly reference build console framework runtime application
19	Synchornization of device-twins states from Azure IoT hub (platform development)	device twin update state properties reported desired property fff result twins received completed size change version notification device_twin_update_complete updating updates
20	Using AltBeacon library to detect the RadBeacon tag from Radius Networks (network management)	beacon oct gmt kura jst active debug beacons wed ibeacon events,length thu bundle pir installed motion osgi start java detect
21	Java heap size – memory – allocation issue (software development)	memory flash size space epc stack fff exception line dispose heap bytes usage plate allocated ram writing sending cpu uint
22	Identifying settings and parameters to create a gateway from GSM to ISDN with TeleService (network management)	sim command modem fiware agent commands gprs orion set context module ipv apn type gsm entity card sophr cellular iotagent
23	Void pointer – statements – functions in c/c++ (software development)	int char float include return void string unsigned null code cmd function sum const double buffer addr address define size
24	Testing Bluetooth low energy devices (hardware management/(testing))	bluetooth ble device devices connection pairing sensortag read gatt mode paired connect pair advertising write connected characteristics tag master peripheral
25	Distributed Hash Table (DHT) and Pub/sub client network issue (network management)	esp char void delay include serial println int loop client println const string define code arduino setup serial client wifi serial.print serial.begin
26	IoT client – continuously retrying connections with errors (hardware and network management)	weo org.eso error org.eclipse.paho.client.mqttv server info iot java caused jar iot-analytics broker carbon mqttexception
27	Intel PROSet for Windows – Device manager WMI Provider guidance (hardware management)	cpu volume protect actor namespace result uwf level platforms wef guid intel protection processor options wmi exceptionmessage cores actors write
28	ThingsBoard, using RPC capabilities and MQTT gateway API reference (network management)	thingsboard true rpc geospatial devices pressure payload deviceb defaultassertionconsumerserviceurl assertionconsumerserviceurl attributes issuer signresponse assertionconsumerserviceurls serviceprovider phenomenontime params datastream telemetry
29	InfluxDB connector schema exploration for various systems (system management)	database sqlite sql provider mysql dism info store tag failure pid table tid tags dism exe conn connection cursor iot core query
30	Issues playing video on media element on some device (hardware management)	audio await tmp play usb sound microphone mediaplayer storagefolder storagefile playing playback file recording mediaelement music treadonlylist player wav
31	Resoding AWS CloudFormation template validation/template format (platform development/(debugging))	properties type true false null set property schema output msg wires return object fcd function schemas e.a cloudformation metadata serialnumber
32	Application debugging using remote console application debugging (software development/(debugging))	studio visual windows remote build deploy version iot update project raspberry debug machine installed debugger install application app package device
33	Unresovled ILMerge import for .Net assembly references (software development)	warning hwn method assembly null mcg files ilmtransform neutral publickeytoken type culture version leshan p-invoke string api unresolved
34	Using sort keys to organize data in Amazon DynamoDB (software development)	table timestamp query key select partition row create deviceid column index insert date cassandra columns document text primary rows queries
35	Json encoding (software development)	json string data message payload format object type array values function send bytes input number output convert code parse return
36	Design and implementation of IoT platforms (platform development)	arial iot service devices application solution cloud answer services system data general approach support mbp based site platform
37	IoT Greengrass troubleshooting (system management/(debugging))	var function aws err error const data clientid region lambda code iot thing host shadow function(err params amazonaws com thingname response
38	Update failed module for new version of IoT edge (software development)	info property jan version iotedged gronpid artifactid dependency jar http utc mgmt warn edgelet http logging compile camaraiq dependencies file error
39	Trigger a button click on the enter key in a textbox (software development)	button click page keyboard text press buttons add control pressed textbox enter input select content screen sender browser webview view
40	Publishing over the MQTT bridge (network management)	google cloud iot core pub/sub registry config device project jwt telemetry firebase configuration pubsub documentation function functions library events state
41	Connection lost exception and error (network management)	public string void import private catch static null return class final override throws client exception e.printstacktrace int java code mqttexception
42	Port forwarding – security and administration (network management)	network address port local internet connect access server connection router connected wifi device proxy mac firewall running open service devices
43	Initializing and creating IoT projects (software development)	arial general nhp iot create
44	Openssl libraries linking error while trying to run make (software development)	reference undefined tlsio_openssl function makes makefile hsm_client_riot hsm_client_tpm compiler openssl target executable error sources cflags compile file cxx linking tpm
45	Group bulk deployments/IoT Devices creation in AWS IoT Greengrass group (system management)	group greengrass provisioning enrollment consumer groups service created dps core registration create provision individual list main leaf readers attestation enroll
46	BackgroundTaskDeferral (system management)	bf,id latest du byte now nmn,nchar events event_count datetime,ullint glatest du byte taskinstance run ibackgroundtaskinstance deferral taskinstance.getdeferral ibackgroundtask schedule true alarm backgroundtaskdeferral
47	Windows Preinstallation Environment (Winpe) adding optional components and packages reference (system management)	image file windows error files package install microsoft failed cab info license ffu languages python build threadid source
48	Routing Protocol for Low-power and Lossy (RPL) networks – border router (network management)	packet contiki packets node linux ipv send udp mote cooja code socket addr nodes radio address bacnet host bytes data
49	manipulating files and directories, read/write files, accessing files and directories (software development)	file files path folder access directory create copy save read drive write download filename text local storagefolder open add project
50	Arduino core for ESP8266 WiFi chip (hardware and network management)	arduino esp board firmware module serial wifi nodemcu connect dev ide library sketch uno flash reset adafruit baud data code

Table 3.4 *Topic names and related top 20 key terms for top 100 topics-(2)*

NO.	Topic Name	(After stemming) Key Terms
51	Stream-processing realtime data feed (platform development)	data store time database sensor stream send analytics storage stored real process streaming sending events dashboard processing series real-time event
52	Constrained Application Protocol option for no server response (network management)	coap location resource resources gps longitude latitude map list request exchange path nm-cse discovery in-cse californium cse proxycres dtls coordinates
53	Sampling multiple channels (networking management)	define channel channels tsc_group bank dest grp_msk io_msk index tempo src uint state sampling default void enabled array const source
54	Software debugging (software management)	loaded win backgroundtaskhost.exe symbols file bit loadedmodule dll sig tpm module open find pdb code debug debugger band sos
55	Developing apps for devices_1 (software development)	app windows application iot uwp core run apps background running raspberry device store universal headless applications update comment console win
56	MQTT connection error (network management/(security))	error line aug failed info timeniil jun jar time version openssl waiting failure message localizedvalue statticjsonbuffer iothub
57	Issue setting up a development install (software development)	install python version build installed linux file run running error intel make npm directory edison ubuntu library command sudo package
58	Embedded SSL/TLS liberary (network management/(security))	wolfssl entering getobjectid debug return leaving amazonaws.com certificate ack ssl_tls.c iot_tls_connect pod error server ssl cipher up site sha cert verify
59	Platform for connecting devices (networking management/platform development)	iot hub azure device sdk messages send message comment connection iothub cloud connect devices sending sample answer amqp service code
60	General-purpose input/output scheme for hardware devices_1 (hardware management)	gpiopin private gpio int gpiopinvalue.low gpiocontroller val gpiopinvalue.high void pin vala convval gpiocontroller.getdefault num delay level task.delay code input low
61	IoT Device-Client hub (network management)	message messages deviceclient static received int string receive partition body sending receiver eventhubclient eventdata hub counter client exception class
62	Web security methods (network management/(security))	user api authentication credentials access key service account login token password users auth create authenticate authorization url credential username list
63	RStudio cannot find fonts to be used in plotting (software development)	arial error code issue problem mbp general working
64	Smarthome software (software development)	software home smart hardware internet project devices development power buy stack control sensors develop question programming security questions things off-topic
65	Connection establishment issues (network management)	connection connect client mqtt error password port connecting token clientid username connected host tls server disconnected closed failed connections refused
66	Identity and access management (network management/(security))	policy iot aws cognito resource action user pool identity effect iam role credentials connect policies version statement attached principal access
67	Defining application methods_1 (software development)	event function thread time method call code loop case running application run program handle start called problem true events
68	Firebase Cloud Messaging (software development)	app android mobile ios phone application push notification things send api apps connect user google services firebase build iphone notifications
69	Defining networking methods (network management)	def code import python print userdata on_message port on_connect client mid msg str obj result paho connection paho.mqtt.client flags
70	IoT hub connector (system management)	string connection hub hostname azure endpoint iothub registrymanager key template connectionstring access portal resource deviceid sharedaccesskey shared error primarykey arm
71	Defining hardware development methods (hardware management)	import private override public void string android.name android return null log.i(tag activity uses-permission class fun method app.static catch error
72	Event-driven, serverless computing (platform development)	aws iot lambda rule thing function shadow dynamodb sdk topic amazon select created message trigger things rules console create action
73	Platform authority (system management/(security))	system.err keystore string void public override status android final throwable null certificate region exception private keystorename keystorepath code log.e connection
74	General-purpose input/output scheme for hardware devices_2 (hardware management)	gpio led pin raspberry turn pins input light spi bus connected control pwm relay output read signal voltage controller motor
75	Object storage/grouping unstructured data in the Cloud platform (platform development)	blob storage upload file container tpm azure account step uploading create signature service job uri rest task import uploaded client
76	Web methods/REST API response codes and statuses (network management)	request http api response post rest url header call send body curl code headers data content requests content-type endpoint application json
77	Web protocols_1 (network management)	socket byte client buffer data exception stream read tcp bytes streamsocketlistener send catch request response buffersize ack listener message bytesread
78	Bug_1 (software development)	error dec pid remote thu line file resource octhread.c declaration iterable null type python typeerror object struct return function occurred script
79	Language usage for application development (software development)	var function err message const code node.js client return node deviceid res msg result nodejs console.log error app true date
80	Testing environment (software development)	kaa event sdk endpoint notification log context profile.key sandbox event.records endpoints return application collection user address eventprocessorhost appender information
81	Web protocols_2 (network management)	gateway nodes node lora network data lorawan onem gateways zigbee protocol mesh send join packets keys agent cbs translation session
82	Embedded system operating system (platform development)	windows iot raspberry core comment rpi device running win usb run card driver microsoft hardware supported install system support desktop
83	Software lightweight packages deliveray (softwarwe development)	edge module docker modules container iot running runtime azure edgehub image inf deployment iotedge logs device containers edgeagent deploy portal
84	Running programs with security privilage (software development)	command powershell script run running commands execute line file start shell administrator startup session task process add program sudo putty
85	Stream-processing realtime data feed (software development)	kafka info thingworx conf-file-poller spark source apache type cygnus configuration streaming mongo-channel channel val mongodb flume http-source lifecyclesupervisor hdfs shutdown
86	Communication interface (network management)	serial serialport port uart null data uint read await task serialdevice usb timespan.frommilliseconds test device datawriter datareader status.text loadasynctask create
87	Developing apps for devices_2 (software development)	make code problem read project create part
88	Web protocols_3 (network management)	mqtt protocol http client broker protocols tcp support websockets library amqp paho java connect application websocket communication connection transport udp
89	System processes and operation (platform development)	time clock date ntp datetime set rtc utc username system cpumsg sessionid workingsetsize virtualsize processid imagename privateworkingset timestamp timezone zone
90	Defining application methods_2 (software development)	byte int rfid static read const return public buf uint uid mfrc buffer data address bytes reader xff tag write
91	Session management (network management/(security))	token sas string var key access sharedaccessignature generate expiry sastoken shared signature static policy sku deviceid explorer return sig uri
92	Defining application methods_3 (software development)	var await public private void string async code return null class task static exception sender method catch comment set true
93	Computer vision/detection techniques (software development)	image ring measurements sample part vision measure frequency difference pixels samples measurement range algorithm machine result surface distance area detection
94	Software package compilation (software development)	capability capabilities package include true deviceapability manifest extensions package.appxmanifest logo compile extension version propertygroup add subtype uap generator usb project
95	Virtual assistance (software development)	alexa debug skill parking java streams vsensorloader-thread mqttcore toolkit handle building wrapper predicate init timed custom stream mypayload
96	Bug_2 (software development)	exception thrown trace stack task location error end movenext catch result previous hresult boolean exceptions object retry occurred
97	Alignment, margins and padding in WPF applications (software development)	xname grid width margin height horizontalalignment verticalalignment xaml text line textblock stackpanel binding center grid.row content fill top left grid.column
98	Realtime transmission web application (software development)	image camera video display stream images webcam screen speech recognition bitmapimage frame resolution uri capture preview mediacapture feed streaming set
99	Python library system (software development)	import python print line def file code message return true error time.sleep time protocol millib module status result call
100	SQL Database (platform development/(analytics))	stream analytics job azure query output input asa select hub sql data iot iothub timestamp storage jobs deviceid blob powerbi

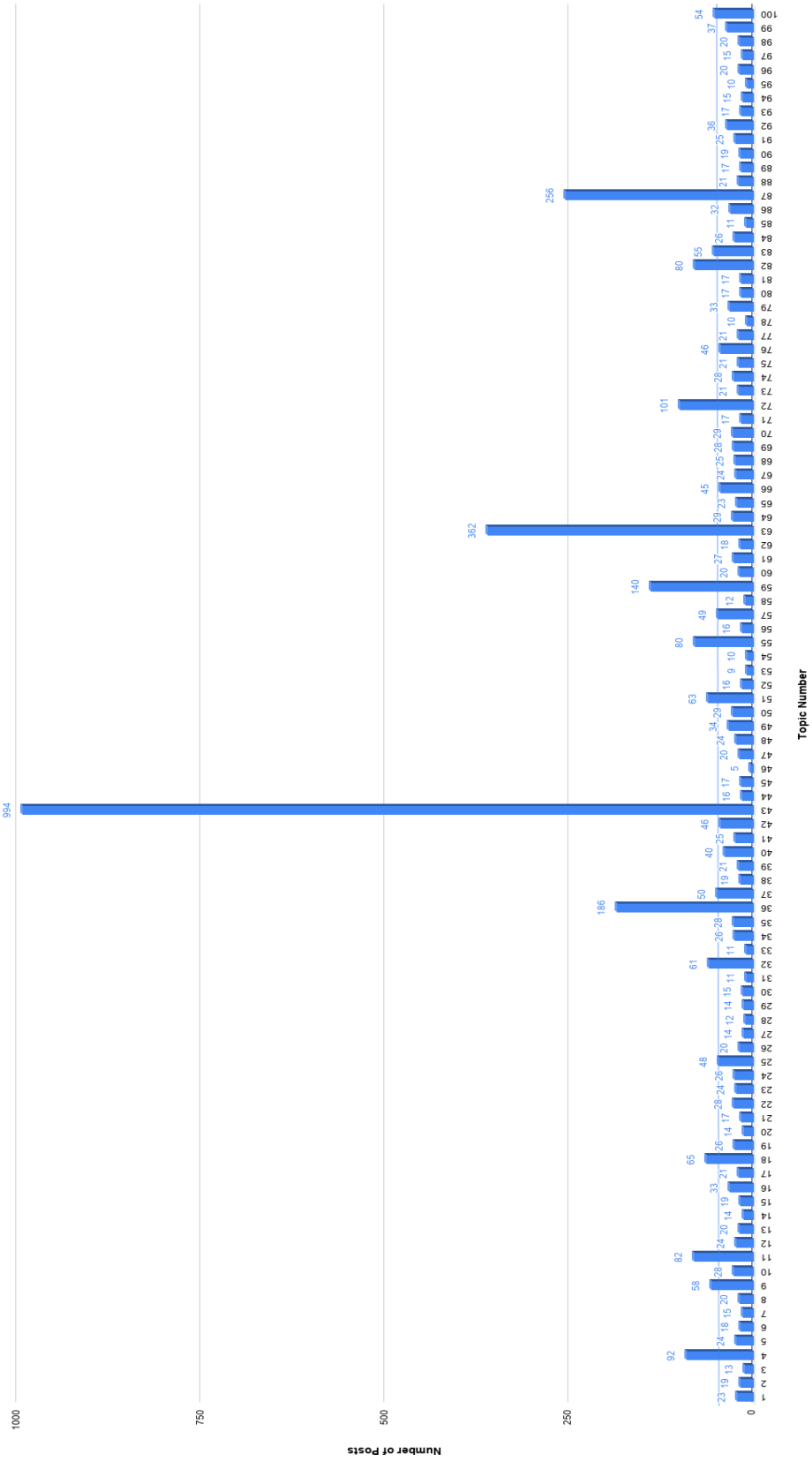


Figure 3.4 *Statistics for each topic of questions*

and 3.4 present the 100 topics including the topic names and the related top “up to 20” key terms.

From the depicted tables, we find that the topics of the IoT-related questions cover a wide range of issues. Some topics have a finer granularity whereas some others have a coarser one. Fig. 3.4 presents the number of questions belonging to each topic. From it, we can explicitly see the distribution of questions in different topics. The topic that has the most questions is “Initiating and creating IoT projects”, and the topic that has the least questions is "BackgroundTaskDeferral – and how it can be defined in a developed app".

Furthermore, we see that all the topics mainly belong to five main categories, i.e., network management (including automation and security), software development, platform development (including debugging and analytics), hardware management (including monitoring and testing) and system management (including debugging and security). Tables 3.3 and 3.4 also assign each topic to such identified categories in the brackets in the second column. Fig. 3.5 shows the number of questions that are belonging to each of the identified categories. From it, we notice that software development issues covers most of the IoT-related questions. It states that IoT-application implementation, development, and deployment issues are very popular among Stack Overflow users.

Discussion. On Stack Overflow, IoT-related questions can span a wide range of topics. Such topics mainly belong to five categories, namely network, hardware, system managements as well as software and platform development. And among them, most of the asked questions do belong to software development category and how to develop applications and–or address the bugs resulted out of deploying applications in the IoT framework. We further look into applications deployment challenges and–or issues in chapters 6 and 7.

- **RQ4.2.: Which topics are the most popular among IoT-related questions?**

Motivation. As we knew the trending topics of the IoT-related questions been asked on Stack Overflow, we would like to go further by investigating which IoT-related topics are the most popular in its time. Identifying and comparing the issues for the IoT-related topics could help, to a great extent, the practitioners understand the trending topics, including possible categories, about the IoT-related questions.

Approach. To be able to measure how popular a topic is, we first collect all the questions related to such topic, then we use four evaluation metrics based on the metadata generated by such questions, i.e., average number of comments of the questions, average number of views of the questions, average score of the questions, and average number of favorites of the questions. In the Stack Overflow data dump that we collected, the number of comments

Table 3.5 *Popularity of top 10 IoT-related topics*

Topic Name (Corresponding Category)	Avg. ViewCount	Avg. CommentCount	Avg. FavoriteCount	Avg. Score
Software Bugs (software development)	3810	1.65	1.06	5.44
Development apps for devices (software development)	2880	1.02	1.44	5.76
Initializing and creating IoT projects (software development)	2783	3.39	1.20	1.39
Design and implementation of IoT platforms (platform development)	2151	1.53	2.04	1.20
Platform for connecting devices (networking management/platform development)	2097	1.23	1.86	5.24
Creation of self-signed SSL certificate (network management/(security))	1712	1.14	1.71	5.70
Testing Bluetooth low energy devices (hardware management)	1676	1.52	1.63	1.67
IoT GreenGrass troubleshooting (system management/(debugging))	1392	3.48	1.28	4.64
SQL Database (platform development/(analytics))	1360	5.44	1.24	4.53
Web methods/REST API response codes and statuses (network management)	1167	4.02	1.16	1.16
Average value	*2,039*	*2.44*	*1.46*	*3.67*

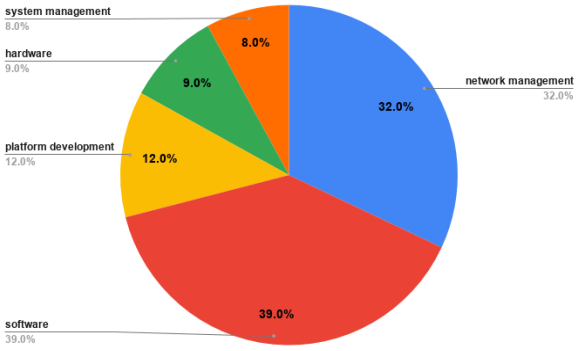


Figure 3.5 *Statistics for each category of questions*

that a question has can be directly retrieved from the attribute named "CommentCount" of the question post. The number of views related to a question can be directly obtained from the attribute named "ViewCount". The score that a question got can be retrieved from the attribute named "Score" of the question post, and the number of favorites that a question got can be directly obtained from the attribute named "FavouriteCount" of the question post.

Among the above four evaluation metrics, we use, by default, the average number of views as the main popularity evaluation metrics since it measures the average number of practitioners viewing the questions that are related to a specific topic. A popular question would attract more practitioners to view. Still, other metrics also have some reference values to estimate the popularity of the extracted topics.

Results. Table 3.5 depicts the four evaluation metrics and indicates the popularity of the extracted topics. We notice that "Software Bugs", "Development apps for devices", "Initializing and creating IoT projects", "Design and implementation of IoT platforms", and "Platform for connecting devices" are the, top five, most popular topics. "Creating IoT projects" is a classic topic in IoT. It has been discussed from different angles and "Platform for connecting devices" is a recently hot topic that is used in more and more applications; e.g., open-sourced Eclipse-IoT frameworks for connecting a diverse set of devices incorporating different pro-

tocols and mechanisms. "Heap size memory allocation", in the software bug category, is a common issue that many practitioners (may) face while developing and deploying their IoT applications.

Additionally, we can see that the average number of comments belonging to the topics "Software Bugs" and "Developments of applications for IoT devices" are all ranked in top, which further represents that such topics are very popular based on these metrics. On the other hand, "SQL Database" and "Web methods/REST API" seem to receive little attention from the community. However, for the "SQL Database" topic, it relatively has high average number of comments (5.44) and an average score of (4.53).

From Table 3.5, on average, each IoT-related question received 2,039 views, which - in turn - indicates that practitioners indeed value different aspects of IoT technology usages.

Discussion. On Stack Overflow, there are many IoT-related topics that are very relevant to practitioners. The top four most popular topics in the IoT domain are "Software Bugs", "Development apps for devices", "Initializing and creating IoT projects", "Design and implementation of IoT platforms", and "Platform for connecting devices".

- **RQ4.3.: Which IoT-related topics are the most difficult to answer?**

Motivation. Here, we would like to know whether some issues are more difficult to answer than others. Finding the most difficult issues will help technology builders identify important aspects of the technology to improve. It will also help researchers better focus their effort toward issues that are challenging for practitioners. To determine the level of difficulty of an issue, we examine how likely it is for questions related to those issues to be successfully answered. Additionally, we study how long it takes for the practitioners asking questions to receive satisfactory accepted answers. We also examine the percentage of all the questions posted on Stack Overflow that receive accepted answers and the average number of answers each question received.

Approach. When a question is posted on Stack Overflow, a number of users may post answers to that question. When the user who posted the question finds a satisfactory answer to their question, they can mark it as an accepted answer. Accepting an answer indicates that the question has been answered to the questioner's satisfaction. We measure the percentage of questions that have accepted answered for each of the topics found in *RQ1*. Moreover, we study the time it takes for those questions to receive an accepted answer. We do this by subtracting the creation dates of the accepted answers and the question posts. After, we calculate the mean times for each of the popular IoT issues. Prior research studies that analyzed Stack Overflow found that most answer activity takes place in the first hour of a

Table 3.6 *Average time until accepted answer – and percent of questions with accepted answers for the top 10 IoT-related topics*

Topic	Mean Time (Days)	% Accepted
Creation of self-signed SSL certificate	3.40	54
Platform for connecting devices	7.15	57
Initializing and creating IoT projects	7.36	13
Design and implementation of IoT platforms	7.54	12
Software Bugs	*7.65*	*52*
Development apps for devices	7.49	57
IoT GreenGrass troubleshooting	4.74	16
SQL Database	4.18	46
Testing Bluetooth low energy devices	1.76	45
Web methods/REST API response codes and statuses	0.09	11
Average value	*5.14*	*36.3*

question being posted [156]. Further, we also measure the number of answers each question received and use this information to measure the average number of answers a question in a topic received.

Results. Table 3.6 shows all the topics highlighting the average time it took for a question in a certain topic to receive an accepted answer and the percentage of questions that received a satisfactory answer as well. Such IoT topics are ordered by popularity based on the average views of questions in each identified topic.

Figure 3.6 shows the percentage of questions receiving satisfactory answers and figure 3.7 shows the average time, in days, for getting accepted answers for the related IoT topics. We see that the percentage of accepted answers vary from 11% to 57% with an average of 36.6% of posts receiving satisfactory answers. The average time it takes to receive a satisfactory answer varies from $\approx$ 0.09 to almost 7.65 days, with an average of 5.14 days to have the questions solved and accepted.

The hardest IoT questions among the popular topics exists in the following categories: software bugs, designing and troubleshooting of platforms, connecting different devices and securing the way they communicate with each other, and initialzing & creating different projects in IoT. Software bugs that occur while developing and implementing different applications has the most questions with accepted answers of (52%). On the other hand, APIs and their response codes has the least amount of questions with accepted answers of (11%), and the lowest average waiting time to receive an accepted satisfactory answer (0.09 days). It seems interesting to note that all of these kinds of questions are heavily related to APIs, such as REST APIs to transfer data from client to server in HTTP protocol, or using the SDK APIs for accessing hardware, i.e., devices, information.

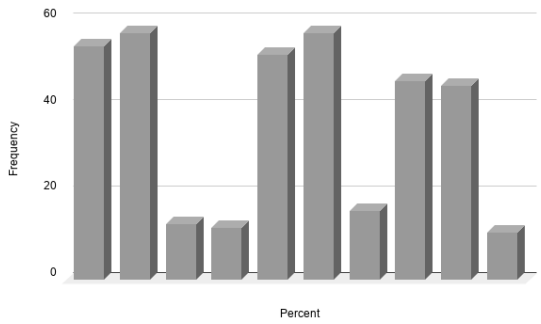


Figure 3.6 *Percent of questions w/ accepted answers*

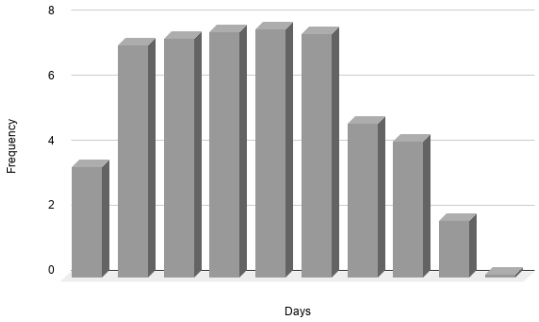


Figure 3.7 *Median time (Days) needed to receive satisfactory answers*

Discussion. The top five difficult IoT-related topics are "Software Bugs", "Design and implementation of IoT platforms", "Development apps for devices", "Initializing and creating IoT projects", and "Platform for connecting devices".

- **RQ4.4.: What concerns are Industrial individuals discussing about the new Industry 4.0 trend?**

Motivation. The interest and use of Industry 4.0 has grown recently. Hence, practitioners, including industrial individuals, targeting these Industry 4.0 manufacturing platforms may find new challenges compared to traditional IoT development. One of the first steps in helping us understand the concerns that they face is to ask: What concerns are being discussed on Stack Exchange?

Approach. Since there are a few number of Industry 4.0 related discussions, topic modeling can still be used to summarize them. In a similar way, we use LDA-based tuned topic models to discover the concerns that industrial individuals are discussing on Stack Exchange. Next, we manually labelled each set of topic keywords to the best of our ability into a classification. We examined the keywords of the questions whose dominant topic was assigned to this topic by LDA. Also, We want to go further by investigating which Industry 4.0 related topics are the most popular. Answer to this research question could help industrial individuals understand a general trend about the Industry 4.0 related discussions and–or concerns.

Results. By using our approach, we group Industry 4.0 related discussions into 20 topics. Table 3.7 presents the 20 topics including the topic names and the related "up to 20" key terms. From the illustrated table, we find that discussions span various concerns ranging from avoiding industrial IoT design pitfalls, communication technologies that can be used in the industrial IoT (IIoT) system, prototyping and development of industrial IoT applications,

to controlling of hardware devices in industrial IoT frameworks. With the small number of discussions we retrieve and the categorization we apply, we find that the topics that have repeated the most among discussions are the "Communication in IIoT" and "Control Strategies in IIoT". Whereas topic that has the least concerns is "Initializing IIoT system". Other general discussions that we also could infer are "IIoT design" and "Software development in IIoT".

Popular topics among Industry 4.0 related discussions

Table 3.8 illustrates the evaluation metrics and shows the popularity of the extracted discussions as well. We can see that "Software development in IIoT", "Control Strategies in IIoT", "IIoT design", and "Communication in IIoT", are the, top four, most popular topics. The average number of comments belonging to the "Software development in IIoT" is ranked in top, which represents that such topic is considered popular and needs to be taken into consideration properly. On the other hand, "Communication in IIoT" and "Control Strategies in IIoT" tend to receive a reasonable attention from the community as well. On average, each industry-4.0 related discussion received 458.8 views, which indicates that practitioners start to be aware of the importance of Industry 4.0 and Industrial IoT in modern systems.

Discussion. On Stack Exchange, Industry 4.0 related discussions are a brand new thread that covers a number of different topics. It is still expanding as the concept evolves. Industrial individuals are talking about some popular topic, such as "Software development in IIoT", "Control Strategies in IIoT", "IIoT design", and "Communication in IIoT".

- **RQ4.5.: What is the current State-of-Practice of the Industrial Internet of Things (IIoT)?**

Motivation. Industry 4.0 and IIoT are frequently mentioned as one of the emerging areas in computing that may have a high potential real-world impact in the coming decade. Here, we analyze the challenges posed when bringing IoT into industrial automation and the opportunities offered by the industrial IoT solutions from a number of different perspectives.

Discussion. IIoT provides opportunities to enhance efficiency, safety, and working conditions for workers. For instance, monitoring food safety using sensors, and using unmanned aerial vehicles allows inspecting oil pipelines. Schlumberger [157], for example, suggests monitoring sub-sea conditions using unmanned marine vehicles, which can travel across oceans collecting data for up to a year without fuel or crew, moving under power generated from wave energy. Via remote monitoring and sensing powered by IIoT, mining industries can decrease safety-related incidents, while making mining in harsh locations more economical and productive.

Table 3.7 *Industry 4.0 Topic names and related top 20 key terms*

NO.	Topic Name	(After stemming) Key Terms
1	Industry 4.0 initialization (Initializing IIoT)	chinese stumble process serial platform smart home executed german integrate digital steering tcp ip memory experimentation
2	Industrial IoT design pitfalls (IIoT design)	lora lte sigfox find disruptor mysterious etsi limits reason feeling finally articles france kpn fee lpwan lorawan simplify iot platform talend relevance
3	Software development for IoT industry applications (Software development in IIoT)	engineering level techniques life nodejs east region protocol match prices assets project map apache object qualify acquisition writing
4	Performance analysis on Industrial IoT (Control strategies in IIoT)	frequencies lorawan high speed computer understand terms consumers regulated bioteeh limitations legal investors shut swisscom works build
5	Communication technologies for Industrial IoT (Communication in IIoT)	bands higher transition invented simplify iot platform engineering source cards search goals
6	Examination of communication technology in IoT (Communication in IIoT)	mqtt connected lpwan link transportation strategic src blockquote flexy modbus end frontier provide dies reasons rich exploration designed
7	Understanding of global standards in Industrial IoT (Communication in IIoT)	interpreted controllers standard mongoose rebooted place limitations price mb providers belgium big talend
8	Prototyping industrial applications in IoT (Software development in IIoT)	implementing found globally mobile applications details relay constraints operating tightly language prototype manufacturer commercial sharing duty chirp
9	Controlling industrial robots in IoT (Control strategies in IIoT)	supported iiot page exact reading plc robotic operation line end higher office solution ase adapt effect amounts actuators difference url
10	Demystifying industrial IoT Security (Control strategies in IIoT)	amp technical goals end equivalent tend mobile data clear current afford european coverage lorawan covers autosar duplicate publish data load kind
11	Connecting Industries with IoT solution providers (Communication in IIoT)	chain servers reason proper revolution lower connections similar shot simple board relate stringent waterproofness enclosures human contrast controller sigfox
12	Technologies and business models for industrial IoT (Control strategies in IIoT)	phrase enterprise it controllers plc hobbyists spread uplink klips explore transition sigfox lora happen catching choices data driven browser start
13	Enterprise architecture for industrial IoT (IIoT design)	cnc hoping seconds europe choice sounds iiot access found it enterprise architectures microcontrollers set intense world bands spectrum subscription possibilities
14	Communication controllers in industrial IoT (Communication in IIoT)	lan ethernet requirement robotic relay board delay switch tcp udp controller module web server href cost services batch controller narrow baud
15	Industrial IoT over network connectivity (Communication in IIoT)	industry iot href industrial application web follow network device tcp ip things data devices mqtt raspberry
16	Strategies for low-cost industrial IoT (Control strategies in IIoT)	company research industry grade application requirement regulations small unacceptable primary failure responsible etsi transmit determine prices choice lte
17	Supporting applications, device, and platforms in IIoT (Software development in IIoT)	lte webpage software communications interfaces sort goran javascript supported coverage belgium early start
18	Designing of industrial IoT handbook (IIoT design)	interesting solution integration company buy microcontrollers failures leaks languages longevity bail sigfox map kinesis autosar
19	Industrial control using hardware devices (Control strategies in IIoT)	alt source chain cyber activation supersystem controller plc fertile mcus raspbian
20	Intelligent manufacturing in the context of Industry 4.0 (Initializing IIoT)	definition worldwide reference technical idea globally sufficient system smart cars high density implement market lan description send

Table 3.8 *Popularity of top 4 Industry 4.0 related topics*

Topic Name (Corresponding Category)	Avg. ViewCount	Avg. CommentCount	Avg. FavoriteCount	Avg. Score
Software development for IIoT applications (Software development in IIoT)	968	0	1	2
Strategies for low-cost industrial IoT (Control strategies in IIoT)	662	9	5	15
Industrial IoT over network connectivity (Communication in IIoT)	421	3	1	6
Industrial control using hardware devices (Control strategies in IIoT)	135	4	0	2
Connecting Industries with IoT solution providers (Communication in IIoT)	108	4	0	-2
Average value	*458.8*	*4*	*1.4*	*4.6*

Despite of its promise, there are a number of challenges in realizing the opportunities offered by IIoT [158], which have to be addressed in the future research. The key challenges stem from the requirements in energy-efficient operations, real-time performance in dynamic environments, the need for co-existence and user's privacy as described down below.

Energy efficiency: Many IIoT applications need to run for a number of years on batteries. This calls for the design of low-power sensors, which do not need battery replacements over their lifespans. Hence, this creates a demand for energy-efficient designs. Many energy-efficient schemes for wireless sensor network (WSN) have been proposed in recent years [159], but those approaches are not applicable to IIoT. IIoT applications need a dense deployment of numerous devices. Sensed data can be sent in queried or continuous forms, which in a deployment scenario, can consume a significant amount of energy. Green networking is thus important in IIoT to reduce power consumption and operational costs. It will lessen pollution and emission, and make the most surveillance and environment conservation. Low-Power Wireless Area Network (LPWAN) IoT technologies can achieve low-power operation using a number of energy-efficient design approaches. (1.) They usually form a star-like topology that eliminates the energy consumed via packet routing in multihop networks. (2.) They keep node design simple by offloading the complexity to the designated gateway. And (3.) They use narrowband channels, hence decreasing the nosie level and extending the transmission range [160, 161].

Real-time performance: IIoT devices are typically deployed in noisy environments for supporting mission- and safety- critical applications, and have timing and reliability requirements on timely collection of environmental data and proper delivery of control decisions. The QoS offered by IIoT is thus often measured by how well it satisfies the end-to-end deadlines of the real-time sensing and control tasks executed in the system [162, 163]. Time-slotted packet scheduling in IIoT has a role in achieving the desired QoS. For instance, many industrial wireless networks perform network resource management via static data link layer scheduling [164–171] to achieve end-to-end real-time communication. This takes a periodic approach to gather the network health status, and then recompute and distribute the updated network schedule information.

Co-existence and interoperability: With the rapid growth of IIoT connectivity, there will be many co-existing devices deployed in close proximity in the limited spectrum. Thus, interference between devices should be handled to keep them operational. Existing and near-future IIoT devices will most likely have limited memory and intelligence to combat interference or keep it to a minimum. To ensure good coexistence, it will become important that future IIoT devices can detect, classify, and mitigate external interference. The challenges of device diversity in IIoT can be addressed along three dimensions: multi-mode radios, software flexibility, and cross-technology communication [172]. Multi-mode radios allow diverse IIoT devices to talk to each other. Software flexibility enables support for multiple protocols, connectivity frameworks and Cloud services. Recently, cross-technology communication [173], without the assistance of additional hardware, has been addressed for communication across WiFi, ZigBee, and Bluetooth devices. Such approaches are specific technologies, and hence, future research is required to enable cross-technology communication in IIoT devices.

Security and privacy: Besides the requirements of energy efficiency and real-time performance, security is another important concern in IIoT. Generally, IIoT is a resource-constrained communication network which largely relies on low-bandwidth channels for communication among lightweight devices regarding CPU, memory, and energy consumption [174]. For this reason, traditional protection mechanisms are not sufficient to secure complex IIoT systems, such as secure protocols [7], lightweight cryptography [175] and privacy assurance [176]. To secure the IIoT infrastructure, existing encryption techniques from industrial WSNs may be reviewed before they are applied to build IIoT secure protocols.

Privacy is a diverse concept. Many definitions and perspectives have been provided in the literature. Generally, privacy in IIoT is the threefold guarantee [177] for: (1) awareness of privacy risks imposed by things and services; (2) individual control over the collection and processing of information; and (3) awareness and control of subsequent use and dissemination to any outside entity. The major challenges for privacy lie in two aspects: data collection process and data anonymization process. Typically, data collection process deals with the collectible data and the access control to these data during the data collection from smart things; data anonymization is a process to ensure data anonymity through both cryptographic protection and concealment of data relations. Due to the restrictions on the collection and storage of private information, privacy preservation can be ensured during the data collection. However, given the diversity of the things in data anonymization, different cryptographic schemes may be adopted, which is a challenge to privacy preserving. Meanwhile, the collected information needs to be shared among the IIoT devices, and the computation on encrypted data is another challenge for data anonymization.

Aspects, such as platform security, secure engineering, security management, identity management, and industrial rights management, must be taken into account throughout the whole life cycle of the systems and products.

3.5 Implications of our Findings

Thus far, we have examined the issues that practitioners ask about, in general, and how difficult such issues are, the issues that are IoT and IIoT specific, and the types of questions being asked. In this section, we further discuss the implications of our findings.

Implications

Our conducted study identifies the most popular topics that both IoT and industrial practitioners ask on Stack Overflow and Exchange. It examines how many of them receive accepted and satisfactory answers. While all topics identified are important in their own right, we believe that researchers and practitioners should pay more attention to the issues that are the most popular and difficult to address.

Implications for Educators

We can see that software development is the category that has the most questions in both Stack communities (i.e., Stack Overflow and Stack Exchange); for instance, it accounts for 39% of all the IoT-related questions found on Stack Overflow. This indicates that practitioners are in need of solutions and answers to questions about developing software for Industrial and IoT systems. Hence, educators should consider incorporating applications of software development for both systems in the software engineering curriculum. In particular, they should consider covering issues related to the deployment of IoT apps.

Implications for Practitioners

Our study shows that the most popular questions viewed and voted by practitioners are related to software Bugs, development of apps for IoT devices, creation of IoT projects, design and implementation of IoT platforms, and platform for connecting devices for IoT. Where as in the industry, they relate to "Software development in IIoT", "Control Strategies in IIoT", "IIoT design", and "Communication in IIoT". Such findings suggest that more effort should be devoted to developing debugging, testing, and fault localisation tools to support practitioners building different IoT systems. Technology builders should also consider

improving the documentation of their products to ease the work of practitioners building IoT systems.

Implications for Researchers

Having the ability to understand what practitioners ask on Q&A platforms, such as Stack Overflow and–or Exchange, is important to ensure that the research community tackles the right problems. Our study show that the most difficult IoT-related issues are related to software development bugs; whereas for Industry 4.0, discussions and concerns are still ongoing and for various topics. Future research should consider investigating novel techniques and tools to help developers locate and fix bugs in IoT systems efficiently. In addition, we also found that practitioners (including software developers) are asking questions about security aspects of IoT systems; such as the creation of self-signed SSL certificates. Moreover, they also asked about troubleshooting of IoT GreenGrass, and how to address SQL databases inquiries. Further research is needed in those areas.

3.6 Threats to Validity

As any empirical study, the work presented in this chapter is subject to multiple threats. We now discuss these threats following common guidelines for empirical studies [178]. Threats to **construct validity** concern the relation between theory and observation. In this work context, it is mainly due to possible mistakes in the detection of IoT – Industry 4.0 related posts. When determining whether a post is IoT- or Industry 4.0- related, we used information in both the title and the body of the identified posts. Then, we extracted the tags associated with such posts and used the tags to determine the final set of posts on both Stack Overflow and Exchange. Our keywords/tags search, in some cases, may not be able to capture an IoT- or Industry 4.0-related post. To alleviate such a threat, we also considered the related tags in each post found. Furthermore, when forming our topics (i.e., issues), we consider the titles of IoT- and Industry4.0- related Stack Overflow/Exchange posts. Using the body text may have improved the process, however, it will also lead to more noise. For instance, users who post questions and–or issues, may add details such as what they have tried, what they encountered, or other details that diverge away from the topic/issue that the poster is really asking about.

When we look at the popularity of the IoT- and Industry 4.0- related posts over the time, it is possible that a small group of the poster and–or users ask the majority of the questions each month, and this might cause a skew in our results a bit. To alleviate such a threat,

we also took a look at the new users who have never posted a question or an answer in the previous months. When performing our LDA computation, since LDA itself is a probabilistic method that will produce different results when executed several times on the same corpora, it is possible that our topic results are, to some degree, random. To mitigate such risk, we ran our final model a number of times and compared each of the topics identified from each model and found no significant difference.

We manually analyzed the results obtained using the LDA computations when we identified the IoT- and Industry 4.0- related posts. To the best of our knowledge, there is no tool, so far, that can give human-readable topics based on a grouping of terms or words. Hence, we read them to be confident that they were mapped to the appropriate issue topics. When having the identified questions paired with the different IoT categories, we considered the tags of the posts. It is also possible, to a some extent, that the practitioner(s) omitted to tag or mislabel their post(s). To alleviate this threat, we manually inspected the dataset and verified that they have proper questions related to the field. Furthermore, when determining if a question was answered successfully, we assumed that the majority of people asking such question-related post(s) on Stack Overflow/Exchange mark the answer(s) as accepted if they solve their problem. However, it is not an obligation for them to do so, and as such, there is a possibility that we did not capture successfully answered posts accurately. On the other hand, when mining for IoT- and Industry 4.0- related Q&As, we only considered the Stack Overflow/Exchange forums. It is also possible that our study does not include popular IoT/Industry 4.0-related topics that are being discussed on other forums and not on Stack Overflow/Exchange. To make sure we alleviated this threat, we, first, performed an exploratory study to look at the popularity and–or trends of the IoT/Industry 4.0-related questions being asked on such Stack communities. **Threats to Internal Validity** concerns our selection of tools. We use the MALLET framework to identify the topics. Other tools could have produced a different result.

Threats to External Validity: One potential threat is that we only used Stack Overflow/Exchange data for our study. Although these platform are very popular in the developer community, it is not the only platform on which developers discuss issues related to the development of IoT and–or Industrial IoT. Our study could be further enhanced by including data from other platforms such as *Hackster.io*, or by going out on the field and ask for feedback directly from IoT practitioners to better understand the major issues they are facing.

3.7 Chapter Summary

This chapter aimed at understanding the issues faced by both IoT and industrial practitioners. We analyzed Stack Overflow and Exchange data to identify what IoT and industrial individuals ask and what questions tend to be most problematic. Furthermore, our study found that IoT–Industry practitioners on Stack Overflow/Exchange mostly ask about network, hardware, and system management in addition to software and platform development or about Software development in IIoT, Control Strategies in IIoT, IIoT design, and Communication in IIoT – in IoT and IIoT respectively. The most popular questions include those that are related to software development for both environments (i.e., they span around 39% of the questions being asked on Stack Overflow). Moreover, we find that security questions (e.g., how to create a self signed SSL certificate) are difficult to answer on the Stack Overflow platform.

Our findings can help research communities and technology builders to better understand issues faced by IoT–Industrial practitioners.

The main contributions of this chapter include:

- We conduct an empirical study on Stack Overflow and Stack Exchange communities to figure out and cluster IoT and Industry-4.0 related questions. To the best of our knowledge, so far, it is the first large-scale study to investigate both IoT- and Industry-4.0- related topics, trends, and concerns on Stack Exchange communities.
- We investigate the popularity and difficulty of both IoT- and Industry-4.0- related topics and provide some recommendations for researchers and practitioners.

In the next chapter, we study the fragmentation, integration and interoperability issues on the IoT paradigm.

CHAPTER 4 IOT INTEGRATION AND INTEROPERABILITY CHALLENGES[1]

4.1 Chapter Overview

Interoperability remains a burden to the developers of the Internet of Things (IoT) systems. This is because resources and Application Programming Interfaces (APIs) are dynamically composed; they are highly heterogeneous in terms of their underlying communication technologies, protocols and data formats, and interoperability tools remain limited to enforcing standard-based approaches. Hence, in this chapter, we investigate the fragmentation, integration and interoperability threats in the IoT paradigm. We analyze the most relevant issues and suggest countermeasures to address them to meet the architectural challenges associated with it.

4.2 Context

IoT opens the door for new applications for machine-to-machine (M2M) and human-to-human communications. The current trend of collaborating, distributed teams through the Internet, mobile communications, and autonomous entities, e.g., robots, is the first phase of the IoT to develop and deliver diverse services and applications. However, such collaborations is threatened by the fragmentation that we witness in the industry nowadays as it brings difficulty to integrate diverse technologies of the various objects found in IoT systems. Diverse technologies induce interoperability issues while designing and developing various services and applications, hence, limiting the possibility of reusing the data, more specifically, the software (including frameworks, firmware, APIs, user interfaces) as well as of facing issues, like security threats and bugs, when developing new services or applications. Different aspects of handling data collection ranging from discovering smart sensors for data collection, integrating and applying reasoning on them must be available to provide interoperability and flexibility to the diverse objects interacting in the system. However, such approaches are bound to be challenged in future IoT scenarios as they bring substantial performance impairments in settings with the very large number of collaborating devices and technologies.

[1]Part of the content of this chapter "Is Fragmentation a Threat to the Success of the Internet of Things?", Mohab Aly, Foutse Khomh, Yann-Gaël Guéhéneuc, Hironori Washizaki, and Soumaya Yacout, is published in *Internet of Things Journal (IoTJ)*, DOI:10.1109/JIOT.2018.2863180

In this chapter, we want to understand the lack of interoperability among technologies developed for IoT and challenges that their integration poses. We also aim at providing guidelines for researchers and practitioners interested in connecting IoT networks and devices to develop services and applications.

4.3 Study Methodology

The following subsections describe the methodology followed to achieve our research objectives.

4.3.1 Conducting the Study

Data Sources

Literature collection was done by making a comprehensive systematic search on the major indexing databases following the guidelines given by [179]. We used ACM digital library, ScienceDirect, Springer, IEEE Xplore, Engineering Village, Web of Science and Google Scholar and did an electronically-based search considering the following terms: 'Internet of Things' AND 'interoperability' AND 'integration' AND 'architecture' AND ('platform' OR 'models' OR 'technology' OR 'framework' OR 'trend' OR 'protocols' OR 'standards') AND 'future directions'. We researched for the published scientific papers related to the IoT technologies between the years of 2000 and 2017, then we did constraint our study to a number of journals, conferences and white papers which having the highest quality in their fields. We carried out this step by choosing the published studies in journals, conferences with high impact factor and competitive acceptance rate. We also check the citation count of the studies being chosen on Google Scholar to evaluate their impact on the evolution of this emerging paradigm. Other studies are excluded for quality reasons (e.g., the study is only a small increment over a previous study, a technical report that is extended into a journal or a conference/workshop paper, etc.), however, if a conference paper is extended into a journal version, we only consider the journal version out of it. We excluded the studies that are not published by well-known publishers or did not pass through the well-defined referring processes as explained by [179, 180].

To gather information about the state-of-practice of IoT, we searched the websites of the major hands-on technology providers and downloaded the white papers published to get a grasp of them, also we searched for the tech blogs published by those technology leaders to identify the challenges in the implementation of IoT technologies. Such blogs provide up-to-date information news and all the technical aspects needed to dive deep into the fundamentals

of IoT; they cover various aspects ranging from technical point of views to use cases and white papers.

4.3.2 Search and Selection Processes

Relevant studies from the aforementioned data sources are organized in three rounds as described in Figure 4.3.1.

- Round 1: We perform electronic search and we narrow our scope review to identify and categorize the preliminary studies related to our subject, i.e., integration and interoperability. Then, we read and select the most relevant studies based on their titles and abstracts; any irrelevant studies are removed.
- Round 2: We read the remaining studies very carefully then any irrelevant studies are eliminated based on the selection criteria identified in the work of [181], we apply different inclusion and exclusion criteria on the remaining studies. These selection criteria can help decide whether to include a paper for further search. Only relevant studies that are retained will be used in this paper analysis. (1) Only papers describing issues related to IoT interoperability and integration are included. (2) Documents presented in the form of powerpoint presentations, abstracts and submitted papers are not included within this study.
- Round 3: Following existing guidelines [180, 182], we perform a snowball search using the reference list of our studies and citations obtained from the previous Round 2 to identify new studies and decide whether to include additional paper(s) or article(s); such technique helps us not to miss important and relevant papers related to the field. Those remaining papers are read carefully afterwards.

One way to narrow down the search space is to conduct a preliminary investigation of the field, i.e., integration and interoperability issues in IoT, by relying on snowballing. The investigation starts by studying publications known in advance and by iteratively extending the known literature set by following the references provided therein [183]. This procedure helps to provide an overview of the publication space and key contributors to conduct the review.

4.3.3 Quality of the Selected Papers

We apply various inclusion and exclusion criteria on the remaining set of studies that resulted from the second and third rounds. These selection criteria help to decide whether to include

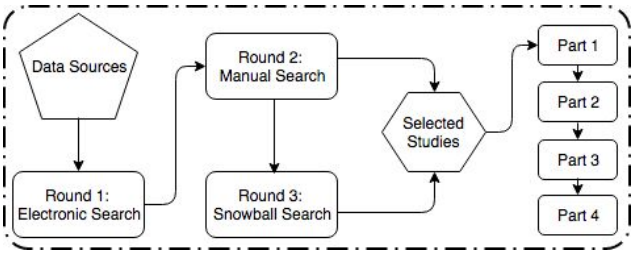


Figure 4.1 *Overview of the Fragmentation Study Methodology*

a paper for further investigation. Below are the criteria used in this study.

- Documents in form of abstracts, powerpoint presentations or submitted papers are excluded.
- Papers touching issues related to IoT interoperability, fragmentation as well as proposing solutions to address these issues are included.

4.3.4 Organization of the Study

The following subparagraphs describe the motivation behind the following tackled parts in this study:

Part 1: Integration and Interoperability challenges in IoT: this part describes the potential interoperability issues that can affect the IoT paradigm. Such study aims to provide a comprehensive overview of common integration and interoperability issues and challenges that hinder the IoT systems. Furthermore, it can help researchers aiming to improve interoperability in IoT identify future research directions.

Part 2: Interoperability Solutions in IoT: this part reviews the solutions and countermeasures proposed to improve IoT's integration and interoperability.

Part 3: Software development issues in IoT: this part sheds the light on software architecture and solutions to be followed to mitigate the interoperability issues identified in this study. It draws a roadmap for further studies in IoT software development.

Part 4: Research future directions: this part introduces some future research directions that can be considered to cover additional integration and interoperability issues in the IoT ecosystem.

4.4 Integration and Interoperability Challenges

It is hard to keep IoT physical parts up-to-date as all devices depend on an integration to provide access or information, hence, can span a wide diversity of technologies, locations, operations and sensitivity levels. The data that the devices provide is usually vast in nature, but might be hard to be transmitted because of the physical limitations of the contributed devices and their environments.

As the IoT evolves, future networks will continue to be heterogeneous, manufactured by multi-vendors, providing multi-services and will be largely distributed. As a consequence, the risk of non-interoperability will increase; this might lead to the unavailability of some of the provisioned services for end-users who can have harmful consequences with regards to the applications related, for example, to emergency health, etc. Or it could also mean that users/applications are likely to loose key information resulted out of IoT due to this lack of interoperability. Hence, it is important to ensure that network components will interoperate to unleash the full value of the IoT paradigm.

Interoperability is considered a key challenge in the realms of the IoT. This fact resides true due to the intrinsic fabric of the IoT as: (1) highly-heterogeneous, where vast systems are conceived by lots of manufactures and are designed for various purposes targeting variety of application domains, making it difficult (if not impossible) to reach out for global service agreements and widely accepted specifications; (2) high-dimensional, with the co-existence/collaboration of different systems (i.e., sensors, devices, machines, etc.) in an environment that relies on communication and exchanging of information; (3) dynamic and non-linear, where new things (that were not even considered at start) are able to join (and leave) the environment at any time and that support new unforeseen formats and protocols, but they need to be able to communicate and share data in the IoT paradigm; and (4) the hardness to describe/model due to the presence of different formats, described in various languages, that can or not share the same modeling principles. This qualifies interoperability in the IoT as a problem of complex nature. We, therefore, need approaches and comprehension of Interoperability for the IoT also making sure it endures, that is sustainable by discussing the protocols and standards that help achieving such task seamlessly.

4.5 Network-layer Interoperability

Power constrained devices require efficient networking standards and protocols. Conventionally, the paradigm is scattered between a number of different power networking protocols (e.g., ZigBee, Bluetooth), traditional networking protocols, like Ethernet, WiFi, as well as

hardwired connections. Such protocols are suggested for domain-specific applications that have the ability of provisioning of distinctive features. Solving interoperability issues at this stage requires standardizations at the software and hardware levels. Different products have been developed to support a number of networking protocols by grouping the required software and hardware components together.

Services in IoT require wireless communications to deploy IoT smart devices easily. As a consequence, routing and communication protocols are considered to be important function to realize practical wireless networks; Dynamic Source Routing (DSR) [184] and Ad-hoc On-Demand Distance Vector (AODV) [185] are renowned routing protocols to be used within the adhoc networks, also a geographic routing is suitable for unstable networks as Vehicular Adhoc Networks VANETs [186]. 6LowPAN makes an assumption that the IPv6 Routing Protocol for Low power and Lossy Networks (RPL) is a routing protocol in the IoT paradigm [187–189], thus, an implementation and evaluation software have been developed on real devices [190–196]. Services in the IoT paradigm rely on upper-layer protocols, such as an application layer [197, 198], a few numbers of research efforts have developed CoAP on real IoT devices to see its impact on them [199, 200]. With respect to communications level, different optional protocols, middleware, and applications programming interfaces (APIs) libraries are considered promising for M2M messaging. Nevertheless that they are based on various different techniques for different application scenarios, the goal is to achieve flexibility and interoperability in interactions among participating devices.

4.6 Messaging-protocol Interoperability

In newly developed IoT applications, a number of application-level protocols, see Section 4.8, are proposed by different enterprises to become the de-facto standards to help the provisioning of communication interoperability [201]. Each protocol possesses specific messaging architecture and unique characteristics that are helpful for various types of IoT applications, which require efficient utilization of the limited energy and processing power capabilities. However, the scalability nature of IoT architecture needs to be independent of the messaging protocol standards, besides providing translation and integration between different popular messaging protocols.

4.7 Data Annotation-level Interoperability

Conventional IoT service model provides raw data captured from the heterogeneous collaborated things found within the system. Such data do not contain intellectual annotation

that needs extensive manual efforts to build practical usable applications. Because of the proprietary approaches employed by the IoT providers, the IoT system has switched to a domain of vertical compartments of different applications with no proper horizontal connectivity among them. This lacks interoperability with self-dependent services endangers the acceptability and adoption of the IoT domains, especially for the applications that gain benefits from the number of different participating devices.

4.8 Proposed Solutions to Integration and Interoperability Issues

To address the above challenges, the research community and industry have been working on the development of standard and implementation practices that would allow better communication between the services provided by different providers and help ease the pain of their integration. In the following, we discuss some key standards IoT technologies along the different layers of the IoT architecture and outline proposed integration guidelines.

4.8.1 Standards and Technologies

IoT requires a number of different technologies. Especially, communication technologies are considered to be a fundamental framework to realize various IoT services. Standards, on one hand, help both developers and users to determine the best technical protocol for dynamic services and applications in IoT. On the other hand, standardization of technologies is crucial that can and will accelerate the speed of the IoT technology.

Recently, some efforts have been put in place to make the incorporation of IP protocol stack into smart object possible. The IP stack has to be adapted since the requirements of smart objects differ than that of the usual participant of the Internet nowadays [202]. Such incorporation has to be done in such a way that a transparent end-to-end connection between devices over the Internet is achieved. To achieve such purpose, a number of protocols have been standardized by the Internet Engineering Task Force (IETF), such as CoAP (constrained application protocol), 6loWPAN (IPv6 for low power wireless personal networks) and ROLL (routing over lossy links). On the other side, i.e. M2M, standardization processes that are driven by the European Telecommunications Standards Institute (ETSI) prepared the Long Term Evolution (LTE) networks for massive low-throughput non-human communications, therefore, full IP connectivity in individual things. Standards play a vital role for further developing and spreading of IoT services; they aim at lowering the entry barriers for both new service providers and users. This will improve the interoperability of different systems/applications and allow products/services to perform better at higher levels [40].

Table 4.1 *Relationship between M2M and IoT Standards*

Standard	**IoT Layer**	**Description**
OneM2M	Service	- Defines a common set of capabilities to support M2M applications, respective access interfaces and the protocols employed over such interfaces, without restricting the technological solutions that could be employed to achieve such capabilities.
ETSI	Service	- Defines a common set of capabilities to support M2M applications and the reference points at which such capabilities are accessed independently of the instrumenting technological solutions.
ITU-T	Service	- Defines a common set of capabilities to support M2M applications and the reference points at which such capabilities are accessed independently of the instrumenting technological solutions.
OASIS	Data	- Defines generic and flexible mechanisms for defining identity information for *things* and exchanging identity information between different administrative domains.
IEEE	Comm.	- Architecture harmonization and multiple application domains support (i.e., verticals).
IETF	Comm.	- Application guidelines provisioning to fit the operation of specific protocols in an IoT setting. Also, defines additional protocols to fill gaps in the protocol solution sets for IoT.

IoT standards have attracted research communities attention to its development [203]; internationally, Electronic Product Code global (EPCglobal), the ITU, International Electortechnical Commission (IEC), International Organizations for Standardization (ISO) and IEEE provided a number of standards to make the identification, capturing and sharing data using RFID technologies easy. On the other hand, the European Telecommunications Standards Institute (ETSI) and the European Committee for Electro-technical Standardization (CEN/CENELEC) released set standards on the IoT fundamental technologies, such as WSNs, RFID, etc. Moreover, the American National Standards Institute (ANSI) in the United States is proactively working on the management standards of IoT. It is worth to stress on the importance of standards for the IoT technological development. Not only standards help to determine the best technical protocols to be used for dynamic applications and service in IoT, but also it is important where they can help in accelerating the spread of the IoT technology.

M2M has a similar meaning of IoT in its context [204]; many authors consider that M2M is focusing on the automatic cooperation between participating entities comparing to IoT to achieve desired services. Although typical M2M devices are not equipped with enough computation power due to hardware specifications or limitations, simplified protocols for resource-constrained devices have been suggested. Industry 4.0 [205–207] and Industrial Internet Consortium (IIC) [208, 209] have considered standards for practical applications on M2M in an industrial domain; they are proactively trying to develop new application platforms for manufacturing floors, such as factories, manufacturing facilities, etc. In the future, direct communication mechanisms among different M2M devices are required to realize flexible and scalable service systems among a number of service domains [210]. Table 4.1

summarizes the relationship between M2M and IoT standards in a layered fashion.

The IoT puts into consideration both constrained nodes and networks, therefore, historical full-stack protocols are not adequate to be deployed on constrained nodes. As a consequence, constrained protocols have been suggested for an IP network as well as the application layers.

- **Message Queue Telemetry Transport** (*MQTT*) [211] is an M2M/IoT connectivity protocol that is lightweight publish/subscribe messaging transport; it is useful for the connections with remote locations where a small code footprint is required and–or network bandwidth is at a premium. Multiple clients connect to a broker then subscribe to one of the topics presented. When clients are connected to that specific broker, they are able to publish their messages to the topic(s) they are subscribed to. Since topics are seen as a hierarchy in their nature, clients are able to handle all topics in the same way as a file system. The protocol defines three Quality of Service (QoS) levels: (a) *QoS 0* that is related to delivering a message once with no confirmation, (b) *QoS 1* for delivering a message at least once with confirmation required, and, (c) *QoS 2* related to delivering a message exactly once by using a four-step-handshake. With those levels, clients and publishers can control the QoS delivery levels according to the service model being considered. Since MQTT protocol requires an underlying transport that provides order and reliable communications, TCP is exclusively used for MQTT to fulfill such necessity. Additionally, TLS is being used to realize a secure function on top of the MQTT protocol.

- **Constrained Application Protocol** (*CoAP*) [212] is a simple application layer protocol that is used by simple electronic devices. It enables such nodes to communicate with the wider Internet using similar protocols. Traditionally, CoAP is designed to easily translate different format to Hypertext Transfer Protocol (HTTP) for simplified integration with web systems while also meeting some specialized requirements, such as simplicity, very low overhead and multicast support [213, 214]. It provides a request/response interaction model between application endpoints, hence, proxying between CoAP and HTTP can easily have messages translated through an intermediary. Simplicity, very low overhead, and multicast are crucial for the IoT and M2M devices that tend to be deeply embedded and have very low memory and power supply; in that essence, efficiency is a very important factor. CoAP has the ability to run on most devices that support User Datagram Protocol (UDP) or UDP analogue, optionally to Datagram Transport Layer Security (DTLS), to provide a high level of communication security. Since it defines two types of messages (i.e., request and response), it uses two simple types of messages, named requests and responses. The format of the header

(short fixed-length binary 4 bytes) is shared by these types of messages where each message contains message ID which is used to detect duplicates. In CoAP, the message procedures are carried with either a method or a response codes, respectively.

- **IPv6 over Low power Wireless Personal Area Networks** (*6LowPAN*) [215, 216] is designed in favor of low-power devices with limited processing power capabilities. The 6LowPAN group defined the encapsulation and header compression mechanisms so that IPv6 packets can be transmitted and received over the IEEE 802.15.4 base networks. Such protocol provides some functionalities, such as adaption layer for interoperability and packet formats between IPv6 and IEEE 802.15.4 domains, addresses resolution between IPv6 addresses and IEEE 64 bit extended addresses on IEEE 802.15.4 and adapting packet sizes between a traditional IP network and an IEEE 802.15.4 network. Maximum message size is limited up to 128 bytes in the IEEE 802.15.4, hence, the fragmentation process associated with this protocol is optimized to covey an IPv6 payload effectively. The protocol uses a dispatch field that is found in the first part of the packet to recognize a type of the packet and defines two types of dispatches as well, first and subsequent fragments, for carrying an IP datagram. Conventionally, the first fragment is used to carry a compressed IPv6 header information, a transport layer header and the first part of a payload; on the other hand, the subsequent fragments are used to carry only a part of the IPv6 datagram payload since the compressed IPv6 header should be an overhead in the limited payload size of IEEE 802.15.4.

Wireless Local Area Network

IoT embedded devices use a communication facility to connect to the Internet. A well-known network standard, IEEE 802.3 (the Ethernet) [217, 218], is used to provide such functionality. Some of the IoT devices do employ such standard to connect to a network when the participating devices are fixed in a facility because power over Ethernet (PoE) can provide electric power to devices as well. Lately, the power usage of IoT devices has been rapidly reduced according to the advancement of semiconductor technology. As a consequence, modern IoT devices use wireless communication devices to get access to the Internet. Below is the mainstream standards to achieve a local network.

- **IEEE 802.11** [219,220] is a known standard for Wireless LAN; recently, WLAN devices support IEEE 802.11n/ac that support Multi-Input Multi-Output (MIMO) technologies on both 2.4 and 5 GHz bands. New standards for different bands have been suggested because existing wireless bands for WLAN are limited and crowded.

- **IEEE 802.11ad** is a Wireless Gigabit (WiGig) standard which supports 7 Gbps on 60 GHz band. The purpose out of it is to enable the provisioning of a high throughput performance in a limited space; this is because a 60 GHz signal attenuates proportionally with the distance.

- **IEEE 802.11af** is a similar Orthogonal Frequency Division Multiplexing (OFDM) technology on TeleVision White Space frequency spectrum (TVWS) which is sub-GHz bands. The OFDM signal's bandwidth in such standard is 6 or 7 MHzs' that is equivalent to the TV broadcasting signal. IETF provides a standard for channel sensing method since the TV towers are using TVWS bands; conventionally, IoT devices should have the ability to sense signals from TV towers or check TVWS databases for adoption. IEEE 802.11af is becoming the mainstream standard that provides high throughput performance on TVWS sub-GHz bands.

- **IEEE 802.11ah** resembles IEEE 802.11af in its functionality as both standards make use of the TVWS bands available. Despite their similarity, the target of each one differs than from the other; the IEEE 802.11ah focuses on long-distance communication with low power consumption and supports 1 and 2 MHz's bands to increase the communication distance. Accordingly, this standard needs to be a core stream standard within the IEEE 802.11 series for IoT devices as the participating devices in IoT require a long communication distance rather than of high throughput performance.

WiMAX

IEEE 802.16 is a collection of wireless broadband standards that provides data rates from 1.5 Mb/s to 1 Gb/s. The recent update (802.16 m) provides a data rate of 100 Mb/s for mobile stations and 1 Gb/s for fixed stations. The specifications are readily available on the IEEE 802.16 working group website (IEEE 802.16, 2014).

Wireless Personal Area Network/Wireless Neighborhood Area Network

1. **Alliance**: There exists alliance groups that develop products for IoT due to the reason that, barely, all IEEE standards for wireless communication define fundamental specifications and not to the extend to make products with mutual compatibility. **Wireless HART** [167], **Thread** [221], **Wi-Sun** [222, 223] and **ZigBee** [224, 225] are some of the specifications that are based on the IEEE 802.15.4.

   - *Wireless HART* is a wireless standard that expands HART (Highway Addressable Remote Transducer) standard of digital industrial automation protocols for processes

automation in factories. One of the advantages of HART is that it has backward compatibility to traditional HART instruments.

- *Thread* is an IPv6 based mesh topology network protocol that provides Thread networking stack on top of the IEEE 802.15.4; this will allow each Thread end device to connect to the Internet through native IP protocols.

- *Wi-Sun* stresses on field area networks for applications (e.g., home energy management, distribution automation, and advanced metering infrastructure); it helps in provisioning secure IPv6 communications over an IEEE 802.15.4g based wireless mesh topology network.

- *ZigBee*'s latest version, 3.0, provides a seamless interoperability among wide range of smart IoT devices. It also defines standard specifications to all levels of network specially applications levels for practical services.

IEEE 802.1 Time Sensitive Networking [226] aims at realizing deterministic communication that ensures real-time communication, transmission delay, and data throughput. The main advantage of this standard is to realize both real-time and critical message deliveries on standardized Ethernet components. This standard is an extension protocol of Ethernet AVB protocol (IEEE 802.1 Audio/Video Bridging) in which those extensions provide minimal transmission latency and high availability in comparison to traditional wireless standards.

CSRmesh, protocol runs over Bluetooth Smart [227], provides message relying over a number of Bluetooth Smart devices as well as enables different products, such as tablets, smartphones, etc. to employ Bluetooth Smart to interact with devices within the range of the CSRmesh network directly.

Z-Wave [228] listed as wireless communication protocol on a sub-GHz band that is used for home automation; it provides reliable, the low-latency transmission of data packets chunks at speeds up to 100Kbps. The physical and MAC layers of this standard use source-routed mesh network architecture to help to deliver messages to the indented destination. Such usage complies with ITU-T G.9959 recommendations [229]. Like typical standards, the Z-Wave network is identified via a Home ID where each participating device is identified via a Node ID as well. In general, Z-Wave has two basic device types: (a) controllers to control other Z-Wave devices, and (b) slaves which are controlled by other Z-Wave controllers. In addition, the Z-Wave alliance defines some profiles specifically for home automation, e.g., ZigBee 3.0, hence guaranteeing the interoperability factor between devices of different vendors.

2. **IEEE 802.15.4**: This standard is allocated to low-rate wireless personal area networks (LR-WPANs) [230], thus, it primarily focuses on both low transmission rate (250 Kbps) as well as power consumption as the participating IoT devices are assumed to operate in a tiny/limited battery capacity. There exist three types of nodes in such standard: (a) Reduced-function devices 'RFDs', (b) Full-function device 'FFD', and (c) PAN coordinator. Such nodes can coordinate to construct either peer-to-peer or star topology networks, but since the limitations that are associated with the smart devices, the maximum packet size is limited to 127 Bytes that include both a header as well as a payload to be transmitted.

   (a) *IEEE 802.15.4g* another standard that is an extension of the IEEE 802.15.4 for low-rate wireless neighborhood area network. It facilitates the control process of the very large scale applications, such as Smart Utility Networks (SUNs) with minimal infrastructure, in the presence of many fixed end-nodes. Metering of electricity, gas, etc., are the targeted utility out of this extended standard. In general, IEEE 802.15.4g builds a smart WNAN; it extends a data payload size of 2,048 Bytes compared to the traditional IEEE 802.15.4. The IEEE 802.15.4e is a standard for MAC layer mechanisms; it is used to realize a low power intermittent operations. It is used alongside with IEEE 802.15.4g as it does not define any physical layer in its configuration.

   (b) *IEEE 802.15.4k* is a physical layer specification for low energy critical infrastructure monitoring network. It is used to provide a low transmission rate ($<$ 10Kbps) as well as a long-distance communication ($>$ 1km). As a consequence, the IEEE 802.15.4 will be from the mainstream standards that are going to be used for the IoT devices.

Wireless Wide Area Network

IoT services are incorporating Cloud services to provide functions to the end-IoT nodes. Hence, communication technologies for wide area networks (WANs) are crucial to realize practical services for IoT. Below is a discussion of the mainstream for WANs.

- **Third Generation Partnership Project** [231] developed a number of different standards dedicated to the cellular network systems (i.e., 2G/3G/4G/5G mobile communication). The main objective of this standard is to achieve high-speed communication, hence, letting smart devices' communication over cellular networks and exchange data in a fast manner (where data rates range from 9.6 Kb/s (2G) to 100 Mb/s and 1Gb/s

(4G/5G)). Second-generation (2G) includes GSM and CDMA, third-generation (3G) includes UMTS and CDMA2000 and the fourth generation (4G) includes LTE. Trends for 4G and 5G cellular networks have two main categories: (a) high-speed communication, and (b) low-speed communication associated with low power consumption. For example, LTE Advanced Release 13 is shedding the light on the new specifications for the IoT devices as well as defining new terminal categories, such as category M1 which is designed to support a narrow-band communication and Narrow Band (NB)-IoT which limits a bandwidth that is less than 180 kHz. Furthermore, it does update the Power Saving Mode 'PSM' specifications and does define an extended Discontinuous Reception 'DRX' to prolongate an intermittent reception interval of paging mechanisms to drop the power consumed.

- **Low-Power Wide-Area Networks** (LPWAN) is a new designed standard for wireless communication technology; it supports low data rate, low power consumption and long-distance communication. As a concrete example of this standard, we highlight LoRa [232] and SIGFOX [233] below:

  - *LoRa* is intended for wireless battery operated smart things to enable IoT. It has a star topology where base stations are seen as a transparent bridge that relay messages between endpoints and servers. Data rates in LoRa are ranging from 0.3 Kb/s to 50 Kb/s and it operates in 868 and 900 MHz ISM bands. A base station of LoRa is not that expensive in comparison to that of *SIGFOX* because radio devices for endpoints and base stations almost have the same specifications. Mainly it aims at guaranteeing interoperability between different operators in one open global standard. In general, LoRa provides symmetric links for the endpoints available and helps attached things to have a battery life up to 10 years.

  - *SIGFOX* builds cellular style systems to serve communication services while employing an ultra-narrow band (UNB) technology, hence, it helps network operators to adopt their technology for conventional IoT deployments. SIGFOX helps vendors to develop their own products via the endpoint available; endpoints in this technology use bidirectional communication to provide high-quality communication service.

- **GSMA/eSIM** The specifications for GSMA Embedded SIM (eSIM) [234] provide standard mechanisms for M2M connection management. Conventional cellular devices need a physical traditional SIM card to connect to network operator; SIM card should be installed into a device's slot to function and connect to the networks available. GSMA/eSIM assume Embedded Universal Integrated Circuit Card (eUICC) as a new embedded SIM function for smart devices; the eUICC identification information (eIC-

CiD) associated to the eUICC allows both the over the air 'OTA' provisioning of an initial operator subscription as well as changing of subscription from one operator to another. An eSIM selects a profile from the installed profiles according to the commands from the subscription manager of a mobile network operator.

OASIS

The technical committee of OASIS has published the Extensible Resource Identifier (XRI) standards and the Extensible Resource Descriptor (XRD) [235, 236]. The syntax of the XRI leverages both the uniform resource identifier (URI) and the internationalized resource identifier (IRI) specifications [237]; it defines a generic syntax for structuring abstract identifiers so that they can be shared among various application domains and embedded transparently found within different URI schemes. Thus, the XRI provides a standard mechanism for resources identification in an abstract way and independently of its concrete representation. Whereas, the XRI Resolution Standard defines a generalized secure protocol for resolving XRI information by the means of the resource descriptions and the HTTP/HTTPS URI information [235].

4.8.2 Open-Source Internet of Things Platforms

In IoT systems, messages can travel from one end-device to another application and/or device via the available WAN, PAN, and platform layer. Recently, a number of open sourced platform layer standards have been suggested in some organizations to address fast delivery of messages between participating smart nodes and devices.

- **OneM2M** standard provides a common M2M service layer that can be easily embedded within different hardware and software components [238]; it defines a number of use cases and requirements for a common set of protocols, APIs, identification and naming of smart devices and applications, security and privacy mechanisms, interoperability, information model and data management, management aspects, as well as services. The benefit of the OneM2M standard is that it considers horizontal service domains in IoT to help to reuse information and creating new values out of such reused information as well.
- ***Web of Things (WoT)*** is a new standard that is used to handle real-world objects and help them be a part of the World Wide Web [239]; it provides a simplified application layer to create new IoT products. Despite the fact that this standard uses HTML5/JavaScript as developing languages, the developed codes should be able to

operate on different kind of hardware, software, and operating system components to realize such integration. Moreover, WoT defines additional standards to obtain information via Web APIs, as a consequence, web applications on every IoT smart device are able to connect each other via those APIs.

- **IEEE 2413** defines an architectural framework for IoT where it includes descriptions of different IoT domains, the definition of IoT domain abstraction and identification of commonalities between different considered IoT domains [240]. Furthermore, it provides some reference architectures to build a reference model according to a practical service application. The architectural framework defined in this standards focuses on cross-domain interaction, aids system interoperability, functional compatibility as well as fuels the growth of the IoT market.

4.8.3 System Model

Recent IoT services have been developed on a vertical system model in which each layer has been designed by an organization or a company. In turn, a recent trend of standardization considers horizontal system model to achieve scalable and interoperable operations in IoT's services [241]. As the previous sections indicate, different types of standards protocols have been suggested to establish cooperative mechanisms between horizontal services domain; those protocols usually focus on standardization in an applications layer. Hence, they assume that inter-accessibility between nodes is guarantee. Contrarily, practical IP networks have some issues with inter-accessibility because of the differences in IP protocol versions and firewalls. Thus, the proposal of a new IoT service layer design to realize inter-operation between end-nodes in different networks is needed.

The current IoT layer model assumes that IP networks are transparent which means end-nodes have the ability to access each other [241]. Such an assumption would be reasonable when an IoT service is operated in a closed IP network, e.g., smart metering systems. As a consequence, a new IoT layer model should be considered to have a middleware layer between the IP and transport layers to achieve transparent connectivity between those end-nodes.

4.9 Internet of Things Integration Practical Guidelines

A framework for sustainable interoperability in IoT is needed; this framework can (and should) learn from the best-of-breed interoperability solutions from related domains to take the good approaches while understanding the differences that the IoT poses [242]. The below five steps can address some of the most common challenge in IoT integration:

1. Adopt an API-first approach: this approach is relevant to the IoT project(s) as they depend on mobile/Cloud Computing technologies that use API-centric approaches. However, this concept should not be misinterpreted as an API-only approach. The API – on its own – is insufficient to meet all the criteria needed to scale up the integration in large-scale distributed systems in a secure way [243].

2. Communication requirements identification for IoT devices: As a first step, identify how things will communicate then select the best-suited technology accordingly (i.e., this can be ranging from cellular networks to short-range wireless technologies, e.g., ZigBee or Bluetooth) [34, 35]. Furthermore, it is also crucial to consider other factors, such as the number and types of things, and how different technologies can handle such variables. Afterwards, identify the optimal network topology "considering new trends as Fog edge computing or gateways" that best suits the requirements for devices' autonomy, aggregation, localized computing, etc. Once such areas are met, it will be easier to assess whether a bundled IoT platform meets the identified criteria for the integration or whether additional solution(s) would be needed to build a network for the participating things.

3. Leveraging Cloud for data and process integration: This step focuses on IoT platforms integration with core business processes [244]. The built-in integration capabilities of IoT platforms are good enough for initial deployments. For example, using an IoT platform for initial implementation, then use a commercial integration solution, e.g., iPaaS [2] platform, to scale up the project being worked on, support more complex integration to implement workflows or to access advanced integration features, e.g., high performance, general-purpose translation.

4. Using selective traditional software: Most enterprises have substantial investments towards on-premises integration middlewares. Despite the non-optimality of those tools for IoT devices connectivity or Cloud services integration, they can help if the used IoT platform must integrate with data and applications that are mostly on-premises [245].

5. API management tools usage: The capabilities of API change in different IoT platforms or other types of middleware. Good management of API involves adding a third-party API management solution to IoT platforms to ensure secure and reliable scaling as APIs widely increase [246]. This holds true if projects involve many APIs are exposed to public networks to provide sensitive or restricted data to end-users.

[2] see source: www.gartner.com/it-glossary/information-platform-as-a-service-ipaas/

4.10 Internet of Things Software Development and Challenges

The emergence of IoT brings a new class of software and–or applications and additional efficiency constraints due to the limited resources of the things. Software-specific requirements, communication and connectivity ability of devices have introduced new challenges for IoT systems' interoperability. With the ever-increasing number of interconnected embedded devices, there is a need for new software solutions to help developers manage software and hardware interoperability issues in a scalable, smart and an efficient way.

4.10.1 Internet of Things Proposed Software Architectures

IoT architecture is still under construction, it is not fixed and does not have a concrete shape yet. However, the rapid development of IoT has triggered a wave of unreasonable expectations [247]. For example, some industries have launched huge projects despite that the key technologies, including the basic architecture of IoT, are still not fully determined. Hence, it would be dis-advantageous to IoT development and may cause an unexpected loss.

A number of researchers adopted microservices architecture [248] which is a new software design pattern that aims at addressing interoperability issues by promoting lightweight–independent services that perform single functions and collaborate with other similar services using well-defined interfaces. Each microservices is dedicated for a single functionality, hence, independent services can be easily deployed into the production environment and any service modification would not affect the whole system. Microservices architecture has multiple advantages, such as independent deployment, the complexity under control, providing more options for technology stack and fault tolerance. All these desirable properties facilitate the development of IoT software and applications on a large scale. Furthermore, they help standardize services' interfaces allowing them to communicate with each other even if they have been developed and deployed on heterogeneous platforms.

4.10.2 Real-world IoT Deployment Platforms

Eclipse Hono, which originates from Eclipse IoT project [249], allows the provisioning of remote service interfaces for connecting different devices and interacts with them uniformly regardless of their type or communication protocol. The platform provides a number of protocol implementations, i.e., HTTP REST, MQTT, etc., Cloud front/back-ends, Machine-to-Machine (M2M) management, devices' authentication, and management among others. It also provides the possibility of multi-tenancy, meaning that the same infrastructure can be shared between different tenants to allow the scaling options for the development of huge

software platforms and applications.

In addition, Kuksa [250], an open-source platform, addresses specific demands of the connected devices where it uses and extends existing technologies to ease development, analysis, and activities for IoT and Cloud-based approaches for interconnected objects. It also provides a basis for new application fields as it contains a Cloud platform that interconnects a wide range of devices via Internet connections. This platform is supported by an integrated open-source software development environment including various technologies to cope with software challenges for devices collaborating in the IoT system.

4.11 Future Directions for the Internet of Things

After connecting people anytime and everywhere, the next important step is to integrate heterogeneous things among themselves and with the Internet [1]. This integration will allow the creation of value-added interoperable services and applications, enabled by their interconnections, in a way that they can be integrated with the current and new business and development processes.

Edge Computing introducing IoT data sources globally strengthens challenges already faced with "Big Data" [251], in particular when considering the typical deployment models of the IoT where data provided by smart sensors (i.e., at the edge of the infrastructure) are transmitted to data centers (i.e., at the core of the infrastructure) for processing. Conveying entire data sets across infrastructures becomes an unrealistic proposition, where instead, approaches that strive to collect data and do computations closer to the sensors (e.g., Edge Computing [252], Near-Data processing [253], etc.) are more practical alternatives to such scenarios. Edge Computing disseminates data to be processed away from the core of the infrastructure and closer to the latter's edge, as close to the data sources as possible, even trying to make such processing on/at the device itself. Placing the processing near to data sources is beneficial in cases of video, for example, whose transport across infrastructure can claim considerable network resources (i.e., not forgetting the resources that are needed for its storage). It is considered to be more resource-effective to process real-time information at its source than extract all relevant features, such as objects, faces, etc. in the Cloud. With Big Data, the cost of transmitting data from its source to the computing facilities (i.e. destination) is a major concern. Reducing data movement by making computations near to data sources is an approach that is known as Near-Data Processing (NDP) [253]. Edge computing converts used communication protocols by the participating devices into a language that modern smart *things* can understand; this makes it easier to connect things

with modern IoT platforms.

Integrating Social Networking with IoT Social Internet of Things (SIoT) has been suggested by Atzori et al. [254] to address the strong interest of using social networking to enhance the communications among different IoT things. There is a trend of moving from IoT to a new vision of the WoT that allows smart objects to become active actors and peers on the Web [255,256]. Social networks are proposed to perform automatically the discovery of things and services and, thus, improve the scalability of IoT similar to human social networks.

Developing Context-Aware Middleware Solutions for IoT By 2020, billion of things will be connected together. When such a huge number of things is connected to the Internet, it will not feasible for individuals to process all the data collected by them. Context-awareness computing techniques, e.g., middleware solutions for IoT, are suggested to understand sensor data in better ways to help decide which data must be aggregated and processed [257]. Currently, most middleware solutions do not possess context-awareness capabilities. The European Union has identified that context-awareness is a crucial research area and specified a time-frame (2015-2020) for context-aware IoT computing R&D [258]. Middleware solutions deals with heterogeneous devices and manages interoperability among them by understanding the sensory data collected besides providing support to process and store those data and make their interpretation easy.

Internet of Nano-Things Another vision that involves integrating even more devices into the IoT is the Internet of Nano-Things. The Internet of Nano-Things is viewed as the interconnection of nano-scale devices via the Internet and communication networks. While those devices are purposed to communicate via electromagnetic communications, there are a huge number of technical challenges that should be addressed before such an idea becomes feasible [259]. The Internet of Nano-Things is considered a more granular approach to ubiquitous computing than the conventional IoT by embedding nano-sensors inside the devices to communicate together through nano-networks via the Internet for global connection among devices around the world.

4.12 Chapter Summary

In this chapter, we studied the integration and interoperability issues in IoT and some proposed some viable countermeasure to them. Since its inception, IoT services and applications have been developed to be adapted in a vertical service model, i.e., in which every layer has been designed by a company or organization. Interoperability and fragmentation among various service domains are a major challenge. As a consequence, the latest trends of IoT technologies are to adapt horizontal service domain to achieve interoperability between participating things. Various standards and protocols have been proposed by a number of IoT consortiums to tackle the integration issues. However, current IoT technologies do not fully make an inter-operation between various devices in different networks. With the help of IP mobility technologies, interoperability between devices in different networks has been realized. As the IoT market develops, interoperability will be of a crucial factor to the commercial success of IoT services and applications to enable Internet-based collaborative technologies, hence, knowledge and understanding of the IoT standardization landscape and the established architectures for the IoT paradigm is essential.

In the next Chapter, we study the security threats in the IoT paradigm. We investigate their impact and effect on IoT's evolution.

CHAPTER 5 ENFORCING SECURITY IN IOT SYSTEMS[1]

5.1 Chapter Overview

The Internet of Things (IoT) is increasingly becoming a ubiquitous computing service, requiring huge volumes of data storage and processing. Unluckily, due to the unique characteristics of resource constraints, self-organization, and short-range communication in IoT, it always resorts to the Cloud for outsourced storage and computation, which has brought about a series of new challenges security and privacy threats. Therefore, in this chapter, we investigate the security threats affecting the IoT paradigm. We study the most relevant issues and suggest countermeasures to address them.

5.2 Context

With the rise of the IoT technology, the number of IoT devices/sensors has increased significantly. It is anticipated that large-scale sensor-based systems will prevail in our societies, calling for novel methodologies to design and operate those new systems. To support the computational demand of real-time delay-sensitive applications of largely distributed IoT devices/sensors, the Cloud is migrating to the edge of the network where resources such as routers, switches, and gateways are being virtualized.

The open structural design of IoT architecture and the extensive usage of the paradigm cause to encounter conventional security issues for the existing networking technologies. Moreover, cooperation generates challenges as new security challenges can disrupt the systems' regular functionalities and operations. Furthermore, the new horizons in IoT has led to several public security concerns including threats of cyber-attacks, privacy issues, and organized crimes. In this chapter, we want to understand the security threats and challenges across the different layers of the architecture of the IoT systems. We aim at providing and suggesting guidelines for researchers and practitioners interested in understanding IoT security issues based on the solutions and countermeasures proposed in the literature. We want to give them the opportunity to explore the facts that attacks have been launched, and how such challenges can be addressed and which issues still persist.

[1]Part of the content of this chapter "Enforcing security in Internet of Things frameworks: A Systematic Literature Review", Mohab Aly, Foutse Khomh, Mohamed Haoues, Alejandro Quintero, and Soumaya Yacout, is published in *Internet of Things: Engineering Cyber Physical Human Systems*, DOI:10.1016/j.iot.2019.100050

5.3 Study Methodology

The following subsections describe the methodology followed to achieve our research objectives.

5.3.1 Conducting the Study

Data Sources

Literature was collected by conducting an exhaustive systematic search on the major indexing databases following the guidelines given in [179]. An electronically-based search was performed with IEEE Xplore, ScienceDirect, Springer, ACM Digital Library, Engineering Village, Web of Science and Google Scholar using the following terms: “Internet of Things" AND “security" AND (“issue" OR “challenge" OR “threat" OR “solution" OR “countermeasure" OR “mitigation"). Scientific papers pertaining to IoT security published between 2000 and 2018 were searched. Then, some results were eliminated to ensure this study would only include data from journals, conferences and white papers of the highest quality in their respective fields. This step was carried out by choosing the published studies in journals, conferences with high impact factors and competitive acceptance rates. The citation count of the studies chosen on Google Scholar was also verified to assess their impact on the evolution of this emerging paradigm. Some studies were excluded due to their obvious quality problems (for example, a study is only a small increment over a previous study, a technical report extended into a journal or a conference/workshop paper, etc.). However, if a conference paper is extended into a journal version, only the journal version was considered.

Search and Selection Process

Relevant studies from the above data sources were organized in three rounds as depicted in Figure 5.3:

Round 1: An electronic search and a scoping review were conducted to identify and categorize the study-related primary studies according to their scope. The titles and abstracts were read in order to select the most relevant studies. Then, all irrelevant studies were removed from the initial list.

Round 2: At this step, the remaining studies from Round 1 were read carefully and other irrelevant studies were removed following the selection criteria proposed by [181].

Round 3: A snowball search was carried out according to the guidelines suggested by [260]. More specifically, a backward snowball search using the reference list of studies obtained from

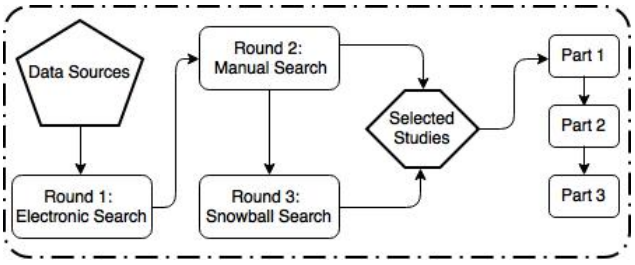


Figure 5.1 *Overview of the Security Study Methodology*

Round 2 was applied to identify new studies and decide whether to include them. These new papers were read carefully afterward.

5.3.2 Quality of the selected papers

Different inclusion and exclusion criteria were applied to the remaining set of studies generated by the second and third rounds. Such selection criteria help decide whether to include a paper for further investigation. Relevant studies that were retained were only used for the purpose of analyzing the study and answering the research questions stated herein. Moreover, only papers that address issues related to IoT security requirements, threats and those that propose countermeasures to overcome such challenges were selected. Furthermore, documents in the form of abstracts, papers submitted and PowerPoint presentations were excluded from this study.

5.3.3 Organization of the Study

The following paragraphs describe the motivation for three distinct parts in this study, once the targeted studies were selected:

Part 1: The IoT reference models were described. This part provides a comprehensive overview of the different IoT frameworks presented. It aims at identifying the trade-offs followed to represent the IoT topologies and highlight the challenges and issues that will help draw the map designed to illustrate the state of research on the security aspects at each layer in these frameworks.

Part 2: Studying the general security requirements in the scope of IoT while identifying the threats in the edge-side layers of the proposed reference models. This part investigates the potential security threats that can affect the paradigm while stressing on what is needed in order to enforce good security practices in IoT frameworks. This exhaustive analysis can help provide a comprehensive overview of the common security issues and challenges that

hinder the IoT ecosystem. Furthermore, it can help identify some future work that can be addressed by researchers to better improve the security aspects and address them in such frameworks.

Part 3: Analyzing the proposed countermeasures so that possible security challenges can be addressed. This part identifies different countermeasures at each layer to overcome the security threats by intruders to lead to better use of the system. And, last but not least, the emerging security challenges that have yet to be explained in depth in previous works are introduced. This part identifies some of the future work that can be conducted in order to cover additional threats that might be exposed to the IoT frameworks.

5.4 Research Questions

This chapter answer the following research questions:

- **RQ6.1.: What are the main issues pertaining to IoT security?**

  Identifying the main security issues faced by the IoT ecosystem will help develop solutions to mitigate them. This question aims at providing an overview of the current IoT security challenges and identifying the main studies that address such a topic. RQ1 is a preliminary facilitator for both RQ2 and RQ3.

- **RQ6.2.: What solutions are proposed to address the IoT security?**

  This question illustrates the proposed solutions in the retrieved literature that address the IoT security issues identified in RQ1. It provides a comprehensive overview of the proposed solutions, benefits as well as their limitations.

- **RQ6.3.: What further research might be needed to extend and improve IoT security?**

  This question aims to identify the main aspects of IoT security that need to be improved. Based on the limitations of the proposed solutions generated by RQ2, this question identifies some future works for the research and industrial communities to better improve the IoT security.

5.5 Internet of Things Security Requirements

IoT-based systems often manage huge amounts of information to be used for sensitive services, ranging from industrial management to e-health care monitoring. Hence, IoT systems have

become an attractive target for attackers interested in eavesdropping and stealing sensitive information that may be of interest to interfere with the system they are trying to access.

They attempt to compromise IoT components, such as Edge nodes and computing devices, to launch attacks against third party entities. In the IoT system, each connected device could be a potential path to personal data or the IoT infrastructure (HP Enterprise and [261]). Concerns in data security are very important nowadays due to their impact, but the potential risks associated with the IoT will reach new levels as autonomous decision-making and interoperability begin to embed complexity, security loopholes, and potential vulnerabilities. More privacy risks will evolve since the complexity that could be found may create more weaknesses related to the services provisioned within the IoT system (i.e., such information will be related to our personal data; for example, location, date of birth, budgets) and others that will need to be protected to maintain the secure transmission of data between objects. As this consists of one of the challenges in Big Data, security researchers and staff need to ensure they think through the potential privacy risks that will be associated with the entire data sets generated.

A critical requirement of IoT is that objects must be interconnected. This gives them the possibility to perform specific tasks, such as sensing, communicating, processing, etc. IoT is able to acquire, transmit and process information from end-nodes, such as sensors, RFIDs, gateways via the network to perform and accomplish highly complex tasks. The paradigm should be able to provide applications with strong security protection for online payment applications. For example, IoT should be able to protect the integrity of payment information [262]. There are widespread concerns about the need for more comprehensive support of security in the IoT system. At a broad level, security requirements are broken down into several categories:

- **Confidentiality** ascertaining that the data flow is unintercepted and that the end-system is not corrupted to prevent intruders from stealing data, credentials or configuration parameters. It implies that the data is accessible solely to authorized entities. The IoT system processes then stores confidential data either at the edge components or on the Cloud to be retrieved afterward. It is also a major aspect for the organization since they may store sensitive information or strategic industrial secrets.

  For instance, client databases are valuable assets. Thus, disclosing it may be disastrous not only for the organization but also for the users involved. This criterion is important for IoT devices as they handle sensitive information. For instance, unauthorized access that broadly discloses personal information in the health sector could cause life-threatening situations [263, 264].

- **Integrity** ascertaining that IoT information received/stored has not been compromised in an unauthorized manner. Infection caused by viruses and worms can be utilized by an inimical agent to alter the original data, thus impacting integrity (among other possible damage). Integrity can also be considered in the context of authentication: making sure that devices are actually what they claim to be and that the identity of the systems/users is not compromised, which is also a necessary component to provide reliable service in the IoT ecosystem. IoT infrastructure hosts critical information; any data manipulated can cause severe damage to the users due to mis-computations. Devices therein must thus ensure that collected information and received commands are legitimate. For instance, a sensitive IoT-based service, e.g., military service, must maintain a high integrity level. Integrity attacks on medical device [265] may have life-threatening outcomes, an IoT Enabled Enterprise Resource Planning (EERP) service must prevent fraud and collusion. There are regional laws, such as HIPPA, that forces organizations to maintain a high level of integrity when processing data related to sensitive services. Data integrity is very important since a cybercriminal attacker may try to manipulate sensitive data by exploiting a misconfigured device or exploiting a software vulnerability.

- **Availability** refers to the assets that are easily accessible and usable upon request by authorized entities (e.g., parties or systems). Such criterion allows data collection and prevents service interruptions. The IoT availability is needed to provide a fully-functioning Internet-connected environment. A large service provider, Amazon Web Services (AWS), faced several unavailability issues [266] which caused services disruptions for a number of popular websites, such as IMDb, Netflix, and Amazon's Instant Video and Books websites. Also, BitBucket, a well-known website that hosts open-source coding projects, was down for an entire weekend because of a Distributed Denial of Service attack (DDOS). It took AWS 18 hours to be able to identify and fix the situation they experienced [267]. Denial of service and flooding attacks are not the only threats to availability; hardware and software failures can also disrupt services.

Authors in [268] discuss four additional key security aspects that should be considered alongside the aforementioned requirements in order to achieve a secured communication framework for people, software, processes, and things:

Authentication:- Things in IoT should be able to identify and authenticate other things. Despite the fact that this is an important factor, this process can be very challenging due to the nature of the IoT system: a variety of entities are involved, such as devices, services, providers, processing units, etc., that may need to interact with each other for the first

time [269]. Hence, a mechanism to mutually authenticate entities in every interaction is needed.

Lightweight solutions are a unique security feature that came into place because of the limitations found within the computational and power capabilities of the devices participating in the IoT. This is not a goal in itself, but rather a restriction that should be taken into consideration while designing and implementing different IoT protocols, either when encrypting or authenticating data and devices in the ecosystem. Algorithms need to be compatible with the limited devices' capabilities.

Policies and standards must be in place to ensure that data is managed, protected and transmitted in a secure and efficient way. In addition, mechanisms to enforce such policies are needed to help entities consistently apply the standards. Service Level Agreements (SLAs) and Service Level Objectives (SLOs) must be clearly elaborated in every service provisioned. Currently used policies in the computer and network security fields may not be applicable for IoT due to the heterogeneous and dynamic nature of the ecosystem. Enforcement of policies will introduce trust between users which eventually results in the growth and scalability of the IoT system.

There is a need to have **key management systems** to allow devices (including sensors) to exchange some encrypted materials to ensure data confidentiality. Lightweight key management systems need to be present in all frameworks to enable trust between different entities and distribute keys among them by consuming minimum resources.

- **RQ6.1.: Internet of Things Security Issues and Challenges**

Before addressing the security issues that are related to the IoT paradigm, we would first like to provide some details regarding the terminology used in the IoT paradigm. Two expressions are most commonly used in IoT security: "secure thing" and "security attack". A secure thing is a thing that meets the aforementioned security requirements (see Section 5.5), whereas, a security attack threatens the balance of the requirements and disturbs the system. It is important to understand the characteristics that shape security when defining what a secure thing is: a number of properties contribute in making the IoT unique in terms of the challenges that need to be addressed and coped within it [270]. According to the Italian National Research Council (INRC), six key features are characterizing the IoT paradigm [271]:

Heterogeneity is related to the variety of devices, technologies, services, and environments. The IoT ecosystem is expected to be highly heterogeneous as a multitude of things from various manufacturers will be integrated to accomplish certain tasks or provide specific ser-

vices. Therefore, compatibility and interoperability must be considered to ensure that the IoT becomes effective.

Scalability is needed to avoid the explosion of resources, exchanging data and operations. Both the heterogeneity and scalability are of primary importance in a complex and dynamic system such as IoT systems. Highly scalable protocols must be adopted in IoT to interconnect the vast number of things found. This, in turn, will influence the security mechanisms.

Cost Minimization can be guaranteed by optimizing the development, maintenance, costs and energy consumption as well as by developing from scratch energy-efficient solutions.

Self as IoT will exhibit a low human intervention (if not completely absent) as objects should offer self-capabilities. Among the notably offered self-capabilities: (a) high degree of configuration autonomy, (b) self-organization and adaption to various scenarios, (c) self-reaction to events and stimuli to which objects are subjects and (d) self-processing of huge amounts of data exchanged which can also be used by third parties.

Observance of **Quality of Service (QoS)**, e.g., bandwidth, delay, is mandatory for the services and applications characterized by sensitive inelastic (real-time) traffic.

Secure environment must be guaranteed in terms of security of communication, authentication, integrity of data and devices, the privacy of users and personal data and trustworthiness of the environment and of the involved parties.

- Internet of Things Vulnerabilities

Since the adoption of the Cloud-IoT paradigm enables several new applications, an in-depth analysis of the security threats and vulnerabilities at each layer that appears in both worlds is provided. For the sake of simplicity, the seven-level IoT architecture was reduced to a four-level model: sensing, network management, service (Fog-edge), and Data Center/Virtualization (Cloud) layers where all the functionalities are found. The attacks that reside at each level are highlighted along with a summary of the possible countermeasures against them in a level-by-level fashion.

1. **Sensing layer** In this layer; security concerns can be classified into ***two*** main categories:

    - Security requirements at the IoT end-node: physically security protection, access control, authentication, non-repudiation, confidentiality, integrity, availability and privacy.
    - Security requirement in the sensing layer: confidentiality, data source authentication, device authentication, integrity, availability.

Below is a summary of the potential threats at the aforementioned categories.

- **Unauthorized access**: Edge computing devices operate in unstable environments where physical access to the devices may be possible, making them vulnerable to a great extent possible to software/hardware attacks. Intruders with physical access to the devices participating in the system may be able to extract critical cryptographic information, tamper with the circuit and modify their programming configuration or even change their operating systems [272, 273]. Physical attacks on edge computing devices may cause permanent destruction for the system exposed, the main purpose of such an attack is to extract valuable information for future use, e.g., find fixed shared keys between devices. In recent attacks on Nest thermostats [273], the attacker tried to replace the default configuration firmware with a malicious one to alter its functionalities. This attack enabled the intruders to control thermostats in spite of the fact that they do not have physical access to the devices.

- **Spoofing attack**: This attack aims to obtain unauthorized access to the network. Malicious nodes injected into a specific network can get access to other nodes where it will be possible to control the network on behalf of the attackers [274]. A malicious node can also be used by the attacker to feed the system with false data or to prevent the delivery of genuinely authenticated messages [275].

- Selfish threat and node replication attacks: Attackers add new nodes, e.g., a malicious one, to a set of existing nodes by replicating the identification number of an existing node. Then, they send it into the formed network to operate. Such kinds of attack lead to a serious reduction in network performance. The attacker can easily corrupt/misdirect packets that arrive at the replica [276]: this leads to severe damage to the system by giving the attacker the ability to obtain authorized access to extract cryptographic shared keys between existing nodes [274]. Authorized nodes can be revoked via node replicas simply by executing node-revocation protocols [276, 277] to unbalance the system.

- **Malicious code** (can also be launched at the network layer): codes, such as Trojan, virus and junk messages, have emerged as major security concerns in the integrated circuits manufacturing [278,279]. It is a malicious manipulation of an integrated circuit that enables the attackers to use the circuit in the way they want or to exploit its functionalities to gain access to data or software running on that piece of circuit [280]. The attacker alters the design before/during the fabrication process and specifies a triggering mechanism to active such malicious behavior of the Trojan and gain access to the system being attacked [278]. There are two categories of Trojans based on their

triggering mechanisms:

(a) internally-activated Trojans that can be activated once certain circumstances are met inside the targeted integrated circuit [280] (e.g., Trojans wake up after a specified timespan when a triggering signal is received from a countdown circuitry added by the attacker), and;

(b) externally-activated Trojans that can be triggered wirelessly by an antenna or a sensor that can easily interact with the outside world [279].

- **Distributed-DoS (DDoS) attack**: On October 21st, 2016, Mirai, a powerful malware was able to infect hundreds of thousands of connected smart devices all over the world via, what is called a dictionary attack, which is composed of 50-62 entries [281]. It relies on the fact that those connected devices are using default login credentials which users almost never change. This massive botnet (i.e., network of infected devices) was in turn considered the largest DDoS attack ever seen in the world, reaching a magnitude of 1.2 Tb per second. This malware was designed to target mostly IoT devices, such as Closed-Circuit Television (CCTV) Cameras, DVRs and home routers. In general, it is meant to infect and control them using the entries of the dictionary attack where when the attack is exploited, devices are then reported to a control server to be used as a part of a large-scale botnet [282]. The formed botnet is used to commit several types of DDoS attacks, exploiting a wide range of network protocols, such as TCP, UDP, HTTP, DNS, and GRE [283] where traffic types (e.g., TCP SYN packets, HTTP GET, and POST requests and GRE packets) often appear legitimate and intended for the victim's network, making them difficult to detect. Hence, the botmaster is able to control the Mirai Control and Command (C&C) servers to take down both the victim's devices and network.

- **Denial of Service (DoS) attack**: Battery draining, sleep deprivation, and outage attacks are three renowned DoS attacks perpetrated against edge computing devices. They are viewed as follows:

(a) *Node sizes* are usually limited. They must thus carry tiny batteries with limited energy capacity to operate. This has allowed battery-draining attacks, a very powerful one that might lead to serious consequences in an indirect way, such as failure to report a security emergency to the sink node, or node outage. If intruders can find ways to exhaust or drain a node's battery on a device, they would be able to bring down the system they are trying to attack [284]. Such kinds of attack could destroy a network if recharging any of the nodes found is difficult [285]. To drain a

battery, the attacker floods a node with tons of random packets and forces it to run its checking mechanisms, an authentication mechanism for example. Numerous attacks regarding battery-draining were thoroughly discussed by [286, 287].

(b) *Sleep deprivation* is a tricky DoS attack in which the victim, from the inside, drains the battery-powered node endowed with a limited energy capacity. In such type of assaults, the attacker tries to send an undesired amount of requests that seem to be legitimate to the limited energy capacity nodes. Thus, detecting such kind of attack is much more difficult compared to a simple battery-draining attack. The idea behind this attack was depicted by [288] and it helped Martin et al.'s, a research team who examined meticulously the impact of the sleep deprivation attacks on energy-constrained devices [286].

(c) *Node outage* happens due to the stoppage of functionalities of an edge device participating in a system. In some cases, a number of devices or administrator devices may stop performing due to an unintended error in their manufacturing process, battery draining, sleep deprivation, code injection or unauthorized physical access to any of the nodes found. Injection Stuxnet [289] is a very famous example of outage attacks that affected Iran's nuclear process control program a few years ago. The attack modifies the industrial process control sensor signals in which the targeted system loses the ability to detect abnormal behavior. Hence, the system cannot be turned off, not even in the presence of an emergency situation [289,290].

- **Availability**: where end-nodes stop working as they are physically captured or logically attacked.

2. **Network management layer** The network layer's functionality is to connect all things found within the IoT environment and allow them to be aware of their surroundings. It is capable of aggregating data from different existing IT infrastructures before transmitting them to other layers, such as sensing, service layers, etc. The IoT connects a variety of different networks which may cause a number of difficulties on the network such as security and communication problems. The deployment, management, and scheduling of networks are vital for the network layer in the IoT. This will enable devices to perform tasks in a very collaborative way.

   - **DoS attacks**: The standard DoS attack aims to jam the transmission of radio signals in order to flood the system. Two types of active jamming attacks are analyzed

in [291, 292]: namely, (i) intermittent (aka non-continuous) jamming where the jamming itself is periodic and as a result, nodes can send/receive packets periodically; (ii) continuous jamming, which involves complete jamming of all transmission. The goal of constant jamming consists of blocking all transmission that occurs in the system. With intermittent jamming, attackers intend to manipulate and reduce the performance of timely-sensitive systems. Consider a fire detection system designed to monitor the air of a confined environment in order to notify the fire department automatically once the levels of certain gases reach a predetermined threshold. Attackers can make the intercepted system unreliable by intermittently jamming node-to-node and node-to-base transmission easily. This way, the system becomes unresponsive if constant jamming is used. A number of research efforts have examined the effectiveness of launching DoS attacks against various transmission protocols, including Bluetooth [293], 6LowPan [294] and ZigBee [295]. Attackers can launch DoS using malicious nodes or routers by inserting a node/router that can intentionally violate the communication protocol to generate collisions or create communication jams [274]. A malicious node or router may refuse to route messages or it could intend to misdirect them to take down the service, in a constant or intermittent way. Constant DoS attacks are usually easily detected, whereas intermittent ones require accurate and efficient monitors to be noticed.

- **Transmission threats**: Fraudulent packets can be injected into communication links using three different attack methods: (i) insertion, (ii) manipulation and (iii) replication (aka replay) [274]. During the insertion phase, attackers add new packets into the communication network: the insertion attack can generate and send malicious packets that seem legitimate to the system. The manipulation phase involves attacks to capture packets before modifying them, e.g., updating the header information with newly added values, checksum, and data, then sending the manipulated packets to other nodes. In the replication attack, assailants capture the packets exchanged between two things before replicating the same packets. A stateless system that does not keep track of previous packets or system states is prone to replication attacks.

- **Data breach**: Eavesdropping (aka sniffing) refers to accessing private conversations over the communication mediums or links [296]. Such an attack can provide information to the assailant when data is unencrypted. In such a situation, confidential information, for example, usernames and passwords, can easily be extracted. Packets transmitted also carry access control information, such as node configuration and identification, shared network password, etc., eavesdropping can provide valuable information about the system being attacked. Attackers can use and process such kind of information obtained to design other custom-made attacks. For instance, if attack-

ers can intercept and extract the information needed to add a new node to the set of authorized nodes, they could easily add malicious nodes to the system and manipulate the system accordingly.

- **Routing attacks** (can also be launched at the sensing layer): Such kind of attacks affect how messages are routed from the source to destination. Attacks that affect the route of the messages are called routing attacks. Intruders may use the routing attacks to spoof, re/mis-direct or drop packets at the communication level. Altering attacks are considered the simplest type of routing attacks in which attackers can change the routing information, for example by generating routing loops or false error messages so that messages are easily misrouted. A number of other serious attacks have been identified in the literature, such as Black Hole, Hello Flood, Gray Hole, Worm Hole, and Sybil. They are briefly described below:

(a) *Black Hole* [297, 298]: this attack is launched by using a malicious node that attracts all of the network traffic with a nasty trick: it advertises that it has the shortest path to destinations in the network. Consequently, all packets are sent to that malicious node and the attacker can process the packets accordingly or simply drop them to disturb the system.

(b) *Hello Flood* [299, 300]: is based on the fact that a node disseminates "HELLO PACKETS" to indicate its presence to its neighbors. The nodes that receive such notifications may assume that they are within a communication range of the sender. Therefore, in this case, the attacker uses a malicious node with high transmission power to send "HELLO PACKETS" to every participating node in the network and claims it is its neighbor.

(c) *Gray Hole* [298]: this is a variation of the Black Hole attack where nodes drop some packets in a selective way.

(d) *Worm Hole* [301]: it is a severe attack that can be launched even when the confidentiality and authenticity methods are guaranteed within the communication system. During this attack, first, assailants record packets at one location in the network and then they tunnel them to a different location.

(e) *Sybil* [302]: the attacker adds/uses nodes with fake identities so that legitimate nodes can be out-voted from the system; e.g., if a node ID is represented by 32 bits, attackers can randomly create 32 bits identities. In some networks, where new nodes are not allowed to join, attackers can steal the identities of legitimate nodes and use them to conduct a Sybil attack.

3. **Service layer** Fog-edge computing is an emerging technology. Hence, its vulnerabilities have yet to be adequately explored. In some scenarios, such as smart city infrastructure, applications and services will require to bring Cloud-like resources or computing entities closer to end-users where computations can be done. This will reduce latency and unnecessary traffic to be sent over to the Cloud in a significant way. With the help of Fog nodes, Fog computing can achieve such a goal. This resides in the fact that Fog nodes are much tinier than Cloud nodes but more powerful than the smart devices or systems at the extreme end of the IoT spectrum. Such nodes offer high performance and low latency to process and aggregate localized data. Thus, an IoT Fog that spans across the access, aggregation and core layers within the network domain and the things of the IoT will improve the feasibility of any system being dealt with in a significant way.

   A few research efforts that address the attacks on edge computing primarily focus on the possible threat to the sensor networks [303,304]. Below are some of the attack scenarios against the service layer-based scheme. Despite the fact that such attacks were designed to target conventional networks, they also apply to Fog-edge based systems.

   - **Privacy threats**: Malicious input injection happens due to insufficient input data validation. Attackers inject a malicious input that causes service providers to carry on operations on behalf of the attacker. They may inject malicious input into the servers by adding an unauthorized component to one of the lower levels (i.e., edge node or communication levels) and they are then able to steal data, bypass authentication levels, and compromise databases' integrity. Standard database error and database messages may also help them to easily compromise the system. In situations where the attackers have no idea or knowledge of the databases' tables, forcing an exception may reveal more details about each of the tables found, as well as the names of their fields [305]. Attackers might use the leaked information from additional components, such as servers and service providers, to perpetrate a number of attacks, for example, traffic analyses and tampering attacks. Moreover, a service which can generate verbose fault warnings provides a useful tool for developers and designers. However, the same kind of warnings can provide excessive information in operational environments.

   - **Service abuse**: Logging helps to detect an intrusion or a hacking attempt. Developers need to log events, such as un/successful authorization, authentication attempts, and application errors. Edge computing-based systems may be damaged as a consequence of insufficient logging [306]. It is recommended to encrypt log files to ensure secure information placing.

4. **Data Center/Virtualization layer** Recently, IoT and Cloud Computing have experienced an independent evolution. However, a number of mutual advantages were identified in the literature and are foreseen in the future as a result of their integration [307]. The Cloud benefits from IoT by extending its scope to be able to deal with real-world things in a more distributed manner while delivering new services in a large number of real-life scenarios. On the other hand, IoT can benefit from the unlimited virtual resources and capabilities of the Cloud to overcome its technological constraints (processing, storage, energy, etc.). The Cloud offers effective solutions to implement different IoT services and applications that exploit the things and the data produced by them: it acts as a middleware between the things and the applications where it hides all the implementation complexity and the functionalities necessary to deliver the latter. This kind of interaction will have a huge impact on future application development where information gathered from things, processed and transmitted will open the door for new challenges to be addressed [308].

   Here is a summary of the security issues that impact such integration and the possible countermeasures against them.

   - **Data security**: There are different data issues for the Cloud paradigm: security risks vary from minor to significant losses. **Data breaches** [309] take place when an attacker succeeds to disclose users' confidential data thus hindering data confidentiality. Threat agents may be internal or external; they can be a malicious employee or a hacker. Data breaches usually cause serious financial losses. In **Data Lock-in** [310], clients are unable to move their data from a Cloud Service Provider (CPS) to another; portability is an issue especially in SaaS and PaaS products due to lack of standards. Since resources are shared among different customers, a malicious user may try to recover data stored in the volatile memory. **Data remanence** issue pertains to the residual representation of data that was nominally erased or removed [311].

   - **Virtualization**: The success of the Cloud relies on a highly virtualized infrastructure. Although virtualization allows multiple users to share the same physical hardware, it is one of the major concerns for Cloud users since users are isolated logically. Protecting the virtualized environment against attacks requires protecting virtual machines and underlying hypervisor. Many attacks were developed by exploiting the VMs or the hypervisors. For instance:

     - *VM Poaching* is a type of denial of service attack. It takes place when a malicious VM consumes more resources than it is allocated and starves other VMs within the hypervisor. Virtual Machine Monitor (VMM) must limit resources that can

be assigned to a single VM [312]. It impacts availability (labelled "A").

- *VM Rollback* is an attack in which a malicious hypervisor executes an older version of a VM without the VM owner's awareness. This attack resets the VM to an earlier version and state which allows an attacker to delete traces including logs and execution history. An attacker may use this technique to return to a previous version to undo security patches, which leaves the VM vulnerable [313]. This impacts Integrity (I).
- *VM sprawl* [314] occurs when many unused VMs continue to consume resources (energy, CPU, memory, storage, licenses) so Cloud administrators can no longer manage them efficiently. This phenomenon causes wastes as unused computing resources are not recycled nor reused. Lacking adequate controls and governance over VMs management causes VM sprawl. This impacts Availability (A).
- *Hyperjacking* takes place when an attacker injects a rogue to control the hypervisor and gain complete access to the virtual environment. As hypervisors run in privileged levels and control all VMs. This approach is more interesting for hackers as it provides them with access to all underlying VMs [315]. This impacts Confidentiality, Integrity, and Availability (C, I, A).
- *VM Escape* [306] is similar to Hyperjacking as it uses a VM to access the host. The attacker starts by infecting a VM before accessing lower layers. [316] describe a VM escape vulnerability that was discovered in Xen. This impacts Confidentiality, Integrity, and Availability (C, I, A).
- In *VM hopping* [317], intruders exploit a vulnerability to allow their migration from one VM to another one on the same hypervisor. This attack compromises both the VM's separation and the host. Since different users can be on the same host, it can compromise and impact the Confidentiality, Integrity, and Availability (C, I, A) of their data. VM hopping attacks are a considerable threat.
- *VM migration* is a benefit of virtualization. A VM can move or copy itself from one host to another which improves availability. However, VM migration brings many security issues as a VM may be altered during migration [318]. An attacker can intercept the VMs which can lead to data leakage. Also, if a VM contains malicious code, it can spread it to the other hosts and VMs. This impacts Confidentiality and Integrity (C, I).
- *Side-Channel Attack* [319] is based on the information acquired from the physical implementation of a system such as CPU usage, memory usage, and other resources. This very sophisticated attack is hard to implement. Timing attacks,

power consumption attacks, and differential fault analysis are some examples of this type of attack [320]. It impacts Confidentiality (C).

- **Availability**: Attacks on availability such as DoS attacks have always been a major threat. They are more frequent and sophisticated in the Cloud environment. If attackers target an organization by launching a DoS, they may also harm other organizations who share the same Cloud resources. Attacks on availability vary from the networking layer to the application layer [321]. In flooding attacks, an attacker tries to saturate the network bandwidth to prevent it from responding to legitimate user traffic: attackers can exploit TCP, ICMP, DNS, etc. The attack can be directed towards the network, the application or reflective attacks may pass through a botnet. Spoofing attacks usually occur with flooding attacks to hide packets origin or to bypass firewalls and filters.

In addition to the well-known traditional DoS attacks that can target the Cloud infrastructure, a new kind of DoS attacks emerged since the democratization of Cloud Computing: Distributed-DOS (DDoS) attacks in the Cloud often result into Economical Denial of Service (EDOS) attacks since it is hard to cause resource starvation when computing resources are auto-scaled on-demand. The real problem with DDOS attacks in Cloud is that consequences are not limited to the victim VM and its resources, but also to CPSs infrastructure and consequently other Cloud users. During a DDOS, a CSP may waste money to restore its services, other tenants hosted on the same physical machine may be affected since VMs are co-hosted. The negative impact ranges from a quality of service degradation to, in the worst-case scenario, complete unavailability.

- **Authorization**: Many known attacks target weak authorization checks such as TOCTOU (Time of Check, Time of Use) [321]. These kinds of attacks are dangerous as successful attackers obtain privileged access to the Cloud infrastructure. Missing authorizations checks is another type of TOCTOU attacks [322]. An example would be a user who shares a DropBox link that provides access to his file: anyone who knows the URL can access the file.

5.6 Review of Machine Learning and Deep Learning Applications in IoT Security

Common Machine and Deep Learning (ML & DL) algorithms are contributing to the IoT security perspective. Such algorithms handle the construction of machines that progress automatically through experience [323]. Recently, learning algorithms have been widely used and applied in practice. The current advancement of learning algorithms has been driven

by the development of new algorithms and the availability of big data in addition to the emergence of low-computational-cost algorithms [323].

In the following subsections, we discuss the most promising ML and DL algorithms in IoT. Firstly, we discuss traditional ML algorithms, their advantages, disadvantages, and applications in IoT security. Secondly, we discuss DL algorithms, their advantages, disadvantages, and applications in IoT security.

5.6.1 Machine learning (ML) methods for IoT security

In this subsection, we discuss common ML algorithms (i.e., Decision Trees (DT), Support Vector Machines (SVMs) and Baysen algorithms, and their advantages, disadvantages, and applications on IoT security.)

1. **Decision Trees (DTs).** DT-based methods are classified by sorting samples according to their feature values. Each node (vertex) in a tree represents a feature, whereas each branch (edge) denotes a value that the node can have in a sample to be classified. Samples are classified starting at the origin node and with respect to their feature values. The feature that helps to split the training samples optimally is deemed to be the origin node of the tree [324]. A number of measures are used to identify the optimal feature that best splits the training samples, including the Gini index [325] and information gain [326]. Kotsiantis et al., in [327] summarized the main points for simplifying DT construction. First, pre-pruning or post-pruning is applied to the tree to reduce its size. Secondly, the space of the states searched is adjusted. Thirdly, the search algorithm is enhanced. Next, the data features are reduced by disregarding or removing redundant features through the search process. As a final step, the structure of the tree is converted into an alternative data structure, such as a set of rules. The main drawbacks of DT-based methods are summarized as follows [327]. Firstly, they require large storage due to the nature of construction. Secondly, understanding DT-based methods are easy only if few DTs are involved, however, certain applications involve a massive construction of trees and several decision nodes. In such applications, the computational complexity is high and the underlying model for classifying samples is complex.

   A DT is used as the main classifier or collaborative classifier with other ML classifiers in security applications, such as intrusion detection [328, 329]. For instance, a prior study suggested the use of a fog-based system call to secure IoT devices [330]. The study used DT to analyze network traffic to detect suspicious traffic sources and consequently detect DDoS behavior.

2. **Support Vector Machines (SVMs).** Are used for classification by creating splitting hyperplanes in the data attributes between two or more classes so that the distance between one hyperplane and the most adjacent sample points of each class is maximized [331]. In theory, SVMs were established from statistical learning. Initially, they were created to categorize linearly divisible classes into a two-dimensional plane including linearly separable data points of different classes (e.g., normal and abnormal). SVMs are suitable for datasets with a large number of feature attributes but a small number of sample points [332, 333]. They are expected to produce excellent hyperplanes that deliver maximum margin by increasing the distance between one hyperplane and the most adjacent sample points of each class. SVMs are scalable and capable of performing real-time intrusion detection and updating training patterns dynamically.
   SVMs have been widely used in a number of security applications, such as intrusion detection [334–336], they are efficient in terms of memory storage due to the creation of hyperplanes to divide data points with a time complexity of $O(N^2)$, where N refers to samples number [332, 333]. In its relation with IoT environments, a study [337] developed an Andriod malware detection system to secure IoT systems and applied a linear SVM to their system. They compared the performance detection of SVMs' with other ML algorithms, including, naive Bayes (NB) and DT. The results showed that SVM outperformed the other ML algorithms and confirmed the robust application of SVM for malware detection. In another research work direction, the SVM was used as a way to secure smart grids besides detecting abnormal attacks in them [338]. Such work showed that ML algorithms:- SVM, perceptron, ensemble learning, and sparse logistic regression are important enough in detecting un/known attacks, performing better than traditional methods used for attack detection in smart grids.

3. **Bayesian theorem-based algorithms.** This theorem explains the probability of an incident on the basis of previous information related to the incident [339]. For example, DoS attack detection is associated with network traffic information. Hence, compared with assessing network traffic without knowledge of previous network traffic, using Bayes' theorem can evaluate the probability of network traffic information. A common ML algorithm based on Bayes' theorem is Naive Bayes (NB) classifier.
   NB classifier is an outstanding supervised classifier which is known for its simplicity. It is able to calculate posterior probability and use Bayes' theorem to forecast the probability that a particular feature set of unlabelled examples fits a specific label with the assumption of independence amongst the features. For example, in intrusion detection, NB can be used to classify the traffic as ab/normal. The features that can be used for traffic classification, such as connection protocol (i.e., TCP and–or UDP), connection

duration and connection status flag, are treated independently by the NB classifier despite that such features may depend on one another. On the other hand, in NB classifications, all features contribute to the probability that the traffic is normal or not; thus, the modifier (naive) is used. NB has been used for anomaly detection [340, 341] as well as network intrusion detection [342, 343]. The key advantages of NB classifiers include simplicity, low training sample requirements, ease of implementation, robustness to irrelevant features (features are preserved independently), and applicability to binary and multi-class classification [344]. However, NB classifiers cannot capture useful clues from the interactions and relationships among features. Features interaction can be important for accurate classification, particularly in complex tasks in which the interaction can significantly help the classifier to increase its discrimination power among classes [345].

5.6.2 Deep Learning (DL) methods for IoT Security

Recently, DL applications to IoT systems have become an imperative research topic [346]. The vital advantage of DL over traditional ML is its superior performance in large datasets. Since several IoT systems produce a large amount of data every day; hence, DL methods are suitable for such systems. Furthermore, DL can automatically extract complex representations from data [346]. Its methods can enable the deep linking of the IoT environment [347]. Deep linking is a unified protocol that permits IoT-based smart devices and their applications to interact with one another automatically with no human intervention. For instance, the IoT devices in a smart city/home can automatically interact to form a fully smart city/home [346]. DL methods provide computational architectures that combine several processing layers to learn data representations with several levels of abstraction. Compared with the traditional ML methods, DL methods have enhanced the state-of-the-art applications in a very considerable way [348]. DL is an ML sub-field which uses several non-linear processing layers for generative or discriminative feature abstractions, and transformation for pattern analysis. In addition, DL methods are best known as hierarchical learning methods as they can capture hierarchical representations in deep architectures. The working principle of DL is inspired by the working mechanisms of neurons for processing signals and the human brain. Deep networks are constructed for supervised learning (discriminative), unsupervised learning (generative learning) and the combination of such learning types, which is called hybrid DL. Recurrent Neural Networks (RNNs) is an example of discriminative DL methods. Restricted Boltzmann Machines (RBMs) is an example of hybrid DL methods.

Recurrent Neural Networks (RNNs)

RNNs were suggested to handle sequential data. In different applications, forecasting the current output depends on the analysis of the associations from several prior samples. Hence, the neural network's output is based on the present and past inputs. In such a formation, a feed-forward NN is inappropriate because the association between the input and output layers are preserved with no dependency [349]. For applications that consist of sequential inputs, such as text, speech and–or sensor data, RNNs are recommended [348, 350]. RNNs integrate a temporal layer to capture sequential data to learn multifaceted variations through the hidden units of the recurrent cell [351]. Such hidden units are modified according to the data presented to the network where these data are updated continuously to reveal the current condition of the network.

The RNN processes the current hidden state by estimating the subsequent hidden state as an activation of the formerly hidden state. RNNs are used because of their ability to manage sequential data effectively. This ability is advantageous for different tasks, such as threat detection, where patterns of the threats are time-dependent. Thus, using recurrent connections can improve neural networks and reveal important behavior patterns. The main limitation of RNNs, however, is the issue of vanishing or exploding gradients [352]. Furthermore, RNNs can be used in IoT security. For instance, smart devices generate large sequential data from different sources, e.g., network traffic flows, which are considered a key feature for detecting several potential network attacks. [353] discussed the feasibility of RNNs to examine network traffic behavior to detect malicious behavior and confirmed the usefulness of RNNs in classifying network traffic for accurate malicious behavior detection. Thus, RNNS provide practical solutions in real-world scenarios. Exploring RNNs and their variants are of significance in improving IoT system security, more specifically for time series based threats.

Restricted Boltzmann Machines (RBMs)

RBMs are deep generative models that are developed for unsupervised learning [354]. RBMs are undirected models that have no link(s) between any two nodes in the same layer. Such models consist of two types of layers: (1.) visible and (2.) hidden layers. The visible layers hold the unknown inputs, whereas the hidden layers consist of multiple layers that include the latent variables. The research in [355] developed a network anomaly detection model that is able to overcome the inherent challenges in developing such a model. Those challenges include the generation of labeled data required for the effective training of the model because a network traffic dataset is multi-part and irregular.

Another challenge is the constant evolution of anomaly behavior with time. Hence, the

Table 5.1 *Securing IoT systems using potential ML–DL methods*

Methods	Working Principle	Advantages	Drawbacks	Potential Application in IoT Security
DT	DT-based method uses a DT to establish a model (i.e., a prediction model) to learn from training samples by representing them as branches and leaves. The pr-trained model is then used to predict the class of the new sample.	Simple, easy-to-use and transparent methods.	Requires large storage because of its construction nature. Understanding DT-based methods is easy only if few DTs are involved.	Intrusion detection [328, 329] and suspicious traffic sources [330].
SVM	Form splitting hyperplanes in the feature dimension of two or more classes such that the distance between hyperplane and the most adjacent sample points of each class is maximised [331]	Are best known for their generalization capacity and suitability for data consisting of a huge number of feature attributes, but only a small number of sample points [332, 333].	Difficulty of selecting optimal kernel. Understanding and interpreting SVM-based models are difficult.	Intrusion detection [334–336], malware [337] and attacks in smart grids [338].
NB	Calculates posterior probabilities. It uses "Bayes" theorem to forecast the possibility that a particular feature set of unlabeled samples fit a specific label with the assumption of independence among features.	Are best known for their simplicity, ease of implementation, low training sample requirements [344] and robustness to irrelevant features (Features are preserved independently).	Handles features independently, hence cannot capture useful clues from the relationships and interactions among features. (It may work effectively in applications whose samples have dependents and–or related features.	Network intrusion detection [342, 343].
RNNs	Integrate a temporal layer to take sequential data and learn multi-faced variations with the hidden unit of the recurrent cell [351].	RNNs and their variants have achieved excellent performance in many applications with sequential data. In some cases, IoT security data consists of sequential data; hence, RNNs have potential application in IoT security	The main disadvantage of RNNs is the issue of vanishing and–or exploring gradients [352].	Can classify network traffic with high accuracy in detecting malicious behavior [353]. RNNs and their variants show considerable potential in improving IoT system security, more specifically for time series-based threats.
RBMs	Are deep generative models developed for unsupervised learning [354] . They are completely undirected models with no link between any two nodes in the same layer.	Using a feedback mechanism or RBMs allows for the extraction of numerous vital features via an unsupervised approach.	Have high computational cost; hence, implementing them on resource-constrained IoT devices to support onboard security systems is challenging.	Can be used for network anomaly detection [355].

model should be adapted dynamically to detect new forms of attacks and generalized to detect anomalies in different network environments. To solve such kind of challenges, the researchers proposed a learning model based on a discriminative RBM; that model was selected due to its ability to combine generative models with suitable classification accuracy to detect network anomalies in a semi-supervised fashion even with an incomplete training set of data [355]. Table 5.1 shows Potential ML–DL methods for securing IoT systems and their advantages, disadvantages, and applications in IoT security.

- **RQ6.2.: Internet of Things Security Countermeasures**

In this section, several countermeasures against the aforementioned threats are presented. Moreover, each defense is described in a level-by-level fashion.

5.6.3 Countermeasures for the Security Threats in the Sensing Layer

Privacy and Side-channel analysis

Such analyses provide an effective approach to detect both malicious software/firmware installed on an entity and hardware Trojans.

A.1.1) Malicious software/firmware detection

Were addressed by several research teams. For example, [356, 357] identified the effectiveness of traffic signal analyses to detect malicious software/firmware installed on an IoT entity.

As mentioned above, malicious nodes and traffic analysis signals can reveal critical valuable information about the device while in operation. Similar to the Trojan detection mechanism, methods of malware detection can process side-channel signals to find abnormal behaviors of such participating device, e.g., a significant increase in the device's power consumption, which is the outcome of the malware installed therein.

A.1.2) Trojan detection

Traffic analysis signals, including timing [280,358,359], power [360,361] and spatial temperature [360,362] have the ability to detect malicious injection and Trojans. Hence, they can be used to help countermeasure a device's infection. Malicious injection and Trojans in circuits affect the power and delay characteristics of the circuit's wires and gates and manipulate heat distribution on the silicon Integrated Circuits (ICs). To detect hardware Trojans, side-channel signal-based Trojan detection mechanisms are used to compare the characteristics of a Trojan-free reference IC with the physical characteristics and the heat distribution map of the suspicious IC. The timing-based methods simply detect Trojans by testing the IC with several efficient delay tests that are highly sensitive to small changes in the circuit delay alongside the affected paths then differentiate Trojans from process variations. Power-based analyses offer an activity monitoring method that can be used to detect suspicious activities, i.e., Trojan detection enabled within the IC being worked with/on. Furthermore, the spatial temperature-based mechanisms rely on infrared imaging techniques which offer IC thermal maps. Generally, silicon is transparent in the infrared spectral region: this transparency provides users with infrared thermal emissions maps generated by infrared imaging techniques [362].

Trojan Activation Strategies

Aim at partially/fully activating the Trojan circuitry to facilitate Trojan detections. Several approaches were proposed [280, 363, 364] in which the common goal consists of detecting and magnifying the disparity between the behavior, outputs or side-channel leakages of a Trojan-free circuit and a Trojan-inserted circuit. [365] proposed an efficient methodology, called MERO, to derive a compact set of test patterns, minimize test time and cost while maximizing the probability of covering Trojan detections. It can be used to increase the detection sensitivity of many side-channel Trojan detection. The concept behind that is to detect low probability conditions at internal nodes, select candidate Trojans trigger-able by a subset of such rare conditions then derive an optimal set of vectors that can track each of the selected low probability nodes.

Intrusion Detections

They make sure that general rules are not modified nor broken down by providing reliable approaches to defend against battery attacks (see subsection 5.5) caused by DoS where they have the ability to detect unusual requests to the targeted node. A number of ongoing efforts provide efficient intrusion detection designs in order to monitor edge nodes as well as to detect potential threats that may occur because of them [366, 367]

5.6.4 Countermeasure for the Security Threats in the Network Management Layer

Intrusion detection systems (IDSs)

These kinds of systems are needed to be placed at the communication level as a second line of defense and protection: they monitor network operation and communication links and raise alters in case of an anomaly, i.e., if a pre-defined policy is bypassed or ignored. Conventional IDS approaches found in [99, 368] are usually custom-made for WSNs or for the traditional operation of the Internet. However, few recent IDS proposals address the security and privacy found in the IoT systems. SVELTE [369] is from the first IDSs specifically designed for such purpose: it meets the requirements of the IPv6-connected nodes of IoT and it can detect routing attacks, such as spoofed or manipulated information and Black Hole attacks.

Reliable routing

A vital characteristic of IoT networks that makes it hard to implement secure routing protocols is that intermediate nodes/server might require direct access to message content before forwarding it. As mentioned above, a number of valid attacks against routing mechanisms were proposed in the literature. [370] addressed most of the major attack scenarios and they provide the first detailed security analysis of major routing protocols and practical attacks against them alongside with countermeasures. Other teams also tried to address both security and privacy concerns in routing, including [371, 372].

Role-based authorization

A role-based authorization system verifies if a unit – for example a service provider, an edge node or a router – is protected against requests by intruders or malicious nodes in the system. All participating entities involved in a communication event should be validated to ensure they are duly authorized via the authorization system [373].

Cryptographic schemes

Several encryption methods have been introduced to address security issues in communication [374]. Strong encryption schemes are used to secure communication protocols. They are considered one of the most effective defenses against most attacks, including eavesdropping and simple routing attacks at the communication level. Unfortunately, the encryption and decryption techniques developed for traditional wired networks cannot be directly applied to most IoT components, more specifically to small battery-powered edge nodes. Edge nodes are usually tiny sensors with limited battery capacity, memory, and processing power. Using encryption usually increases memory usage, delay, packet loss, and energy consumption [375]. Variants AES encryption methods yielded promising results to provide secure communication platforms in IoT. Furthermore, different lightweight encryption methods, such as CLEFIA [376] and PRESENT [377] were proposed. However, at this time, no promising public key encryption methods can provide sufficient security while meeting lightweight requirements [375].

5.6.5 Countermeasures for the Security Threats at the Service Layer

Intrusion detection systems (IDSs)

Can sense and detect the existence of malicious nodes that try to inject invalid data into the system or even try to violate the policies identified above. A number of research efforts proposed IDS based methods to address injection issues [378, 379]. For instance, [378] suggested the design and implementation of DIGLOSSIA: a new tool designed to detect code injection attacks on servers in a very precise way.

Pre-testing

The behavior of the whole systems and their components, e.g., edge nodes, servers, routers, etc., should be carefully examined by feeding different inputs into the system then monitoring the outputs. Testing updates and design implementation are crucial before using any system, especially the critical ones [380]. More particularly, simulating different scenarios to see how the systems react to possible attack attempts is one of the main goals of the pre-testing method [381].

Pre-testing specifies what data should be logged and what information is too sensitive to be stored to ensure the security of the edge computing levels. Moreover, input files should be tested to prevent the danger of malicious injections that could happen. For example, the

attacker should not be able to execute any command by injecting it into the input files.

5.6.6 Countermeasures for the Security Threats at the Data Center/Virtualization Layer

Hypervisor Techniques

HyperSafe [321] is designed to provide control-flow integrity to hypervisors. This technique uses the Trusted Platform Module (TPM) that is designed for the load-time integrity of the hypervisor such as verifying OS components signatures before loading them. HyperSafe suggested implementing a non-bypassable memory lockdown which prevents unauthorized malicious writes on the memory pages.

It performs a restricted pointer indexing which uses the memory lockdown technique. It precomputes the control flow targets, then, stores them in a table. Hypervisor Integrity Checking Solution [382] provides stealth and in-context integrity checking of the hypervisor. It does not require additional hardware, but it uses the TPM instead as an integrity component. It implements an out-of-band channel which keeps it stealth. This technique protects hypervisor from scrubbing attacks where an assailant tries to delete evidence. A verifiable and a non-interruptible measurement agent is used to check the current state of the CPU as it is an in-context agent.

[383] proposes an intrusion detection framework for Cloud environments that uses performance signatures to detect attacks. This Hypervisor-based Cloud Intrusion Detection System (HCIDS) does not require knowledge of the underlying operating system or the applications running on VMs. Patterns of performance metrics are examined from outside of the instance, directly from the hypervisor, without placing a burden on the Cloud user. Thus, monitoring is independent of the operating system or applications running on the virtual machines. The authors suggested installing an agent which communicates with a central decision node, on each hypervisor node, to ensure central nodes can detect an attack if it occurs. The HCIDS is a signature-based detection system which only limits its detection to known attacks.

VM Isolation Techniques

This technique ensures that intruders cannot tamper with the guest VMs even if they managed to completely subvert the host. Thus, a guest VM cannot access nor modify the hypervisor nor a separate VM. CloudVisor [384] uses a nested virtualization technique by adding an extra software layer under a conventional hypervisor to protect the privacy and integrity of users' VMs. Using cryptography, CloudVisor guarantees the integrity and confidentiality

of data and software running on the VM even in the presence of an intruder at the hypervisor level. [385] proposes a Mandatory Access Control based (MAC-based) security architecture for hypervisors: the technique uses a MAC-based policy to allow coalitions of VMs to share resources – for example network, disks, memory, domain operations. Bind-time authorization is used for high performance whereas a Chinese wall policy is enforced to counter traffic analyses and side-channel attacks. The latter ensures that certain VMs cannot run on the same hypervisor system simultaneously.

Secure live Migration Techniques

During the migration process, a VM is vulnerable to tampering attacks. Many approaches were proposed to secure VM during migration. Inter-Cloud mobility [386] uses a secure channel dedicated for VM migration between Cloud proxies with non-shared storage and virtual network migration components. Proxy servers are used in source and destination Clouds to hide details of their networks which ensure privacy. Besides, proxies restrict access to hosts to protect them from malicious access. An SSH tunnel ensures that the data is transferred through a secure path between authorized Clouds.

[387] suggested the Trusted Token migration technique, which consists of strict policy, migration policy, and some audit components. Policies are defined by the Cloud users: they specify the acceptable Trust Assurance Level (TAL) value of the target Cloud platform for VM migration and then, the CPS implements them. The migration audit component helps users to validate that the CPS follows the defined migration policy. VM migration only occurs if the TAL value of the destination platform is higher than the TAL value of user's migration policy.

Virtual Machine Introspection (VMI)

This feature consists of monitoring VMs from the hypervisor level or a privileged VM to inspect states and activities of Operating Systems running inside them (called guest OSes). The VMI is effective to detect malicious behavior in VMs. Lares [388], an in-VM technique, uses hooks which are injected in kernel components to target VMs. These hooks trap execution in the untrusted VM. When hook handlers are invoked due to the execution of the monitored activities, they draw information that will be communicated to the hypervisor. In-VM VMI requires knowledge and modification of the underlying operating system, as usual, VMI agents are deployed on guest machines to collect data for analysis. Livewire [389] is a VMI based on an out-of-VM technique that aims to protect the guest VM kernel code and critical data structures from being modified. The proposed IDS maintain high visibility on the

monitored VMs. Livewire retrieves information about VM OS data structures, periodically, using a hypervisor adapted version of crash logs. If inconsistencies are detected between both views, it determines whether the guest OS has been compromised. Livewire isolates the IDS from the VMM to reduce the risk of an IDS compromise leading to a compromise of the VMM.

5.6.7 Blockchain Solutions for IoT Security

A blockchain is fundamentally a decentralized, distributed, shared and immutable database ledger that stores registry of assets and transactions among peer-to-peer networks. It has chained blocks of data that have been timestamped and validated by miners [390]. The blockchain uses elliptic curve cryptography (ECC) and SHA-256 hashing to provide strong cryptographic proof for data integrity and authentication [391]. Fundamentally, the block data contains a list of all transactions and a hash to the previous block.

Such technology has a full history of all transactions taken and provides across-border global distributed trust. Centralized authorities and services or Trusted Third Parties (TTP) can be disrupted, compromised or hacked. They also can misbehave and become corrupted in the future, even if they are trustworthy now. In the blockchain, each transaction in the shared public ledger is verified by a majority consensus of miner nodes that are actively involved in verifying and validating transactions. Blockchain can be built as (1) permission-ed (or private) network that can be restricted to a certain group of participants, or (2) permissionless (or public) network that is open for anyone to join in. Permission blockchains provide more privacy and better access control [391].

Potential blockchain solutions

Despite a number of potential benefits, digital disruptions constitute many challenges related to information privacy and security. In the context of IoT, blockchain has been foreseen, by both industry and research communities, as a disruptive technology that is poised to play a crucial role in controlling, managing and securing smart devices. It can be a key enabling technology for providing viable security solutions to today's challenging IoT security issues.

We discuss and summarize some of the intrinsic features of blockchain that can be immensely useful for IoT in general and IoT security in particular.

- **Address Space**. Blockchain has a 160-bit address space, as opposed to IPv6 address space which has 128-bit address space [391]. A blockchain address is 20 bytes or a 160-bit hash of the public key generated by the Elliptic Curve Digital Signature

Algorithm (ECDSA). With a 160-bit address, blockchain can generate and allocate addresses offline for around $1.46 * 10^{48}$ IoT devices. The probability of address collision is approximately 10^{48}, which is considered sufficiently secure to provide a Global Unique Identifier (GUI) that requires no registration or uniqueness verification when assigning and allocating an address to an IoT device. With blockchain, centralized authority and governance, like that of the Internet Assigned Number Authority (IANA), is eliminated. Currently, IANA oversees the allocation of global IPv4 and IPv6 addresses. Moreover, blockchain provides 4.3 billion addresses more than IPv6, hence making blockchain a more scalable solution for IoT than IPv6. Many IoT devices are constrained in memory and computation capacity and therefore will be unfit to run an IPv6 stack.

- **Data Authentication and Integrity**. By design, data transmitted by IoT devices connected to the blockchain network will always be cryptographically proofed and signed by the true sender that holds a unique public key and GUID. This ensures both the authentication and integrity of transmitted data. In addition, all transactions that are made to or by an IoT device are recorded on the blockchain distributed ledger and hence can be tracked securely.

- **Authentication, Authorization, and Privacy**. Blockchain smart contracts have the ability to provide decentralized authentication rules and logic to provide single and multi-party authentication to smart devices. They can also provide more effective authorization access rules to connect IoT devices in a less complex way when compared with traditional authorization protocols, such as OAuth 2.0, Role-Based Access Management (RBAC), OpenID and Open Mobile Alliance (OMA) Light Weight Machine-to-Machine (LWM2M). These protocols are widely used nowadays for IoT smart device authentication, authorization, and management. Furthermore, data privacy can be ensured via using smart contracts which set the access rules, conditions and time to allow a certain individual, group of users and–or machines to own, control or have access to reside data at rest or that in transit. In addition, the smart contracts can spell out who has the right to update, upgrade, patch the IoT software or hardware, rest the IoT device, change ownership and provision or re-provision of the device.

- **Identity of Things (IDoT) and Governance**. Identity and Access Management (IAM) in IoT must address a number of challenging issues in an effective, efficient, secure and trustworthy manner. Ownership of a device changes during the lifetime of the device from the manufacturer, supplier, retailer, and consumer [392, 393]. The consumer ownership of a smart device can be changed and–or revoked if the device gets re-sold, compromised or decommissioned. Attribute management and relationships of

an IoT device is another challenge; attributes of a device can include manufacturer, make, type, location, deployment GPS coordinates, serial number, etc. Apart from attributes, features, and capabilities, IoT devices have relationships that may include device-to-human, device-to-service or device-to-device. Smart device relationships can be deployed by, sold by, used by, shipped by, upgraded by, repaired by, etc.

Blockchain has the ability to address all of the above challenges and solve them easily, securely and efficiently. It has been widely used for providing trustworthy and authorized identity registration, ownership tracking and monitoring of products, goods, and assets. Approaches like TrustChain [394] are proposed to enable trusted transactions using blockchain while maintaining the integrity of the transactions in a distributed environment. Blockchain technology can be used to register and give identity to connected IoT devices with a set of attributes and complex relationships that can be uploaded and stored on the blockchain distributed ledger.

In addition, Blockchain also provides trustworthy governance, management and tracking at every point in the lifecycle and supply chain of IoT devices, as shown in the above figure, Fig. 5.2. The supply chain includes multiple players, such as vendors, suppliers, factories, distributors, shippers, installers, owners, repairers, re-installer, etc. As in figure 5.2, keypairs can be changed and–or re-issued at multiple points within the lifecycle of an IoT device. Issuance of keypairs can be done initially by the manufacturer, then by the owner, periodically after the deployments.

- **RQ6.3.: Internet of Things Emerging challenges**

So far, a comprehensive study of the numerous attacks against the security of things and individuals, along with the countermeasures to respond to such assaults, has been provided. However, emerging security challenges have yet to be explained in details due to their novelty in the field. In the following subsections, we investigate the challenges being envisaged for effective implementation of security for IoT devices.

5.6.8 Forensics challenges in the IoTs environments

Most aspect of our lives, if not all, will be changed in the near future, thanks to the IoT. This will be ranging from managing our homes, cars, and cities, etc. It is only a matter of time before individuals and–or industries sue one another for misusing and–or misconducting their smart things and appealing on the attackers who compromised any of the smart sensors

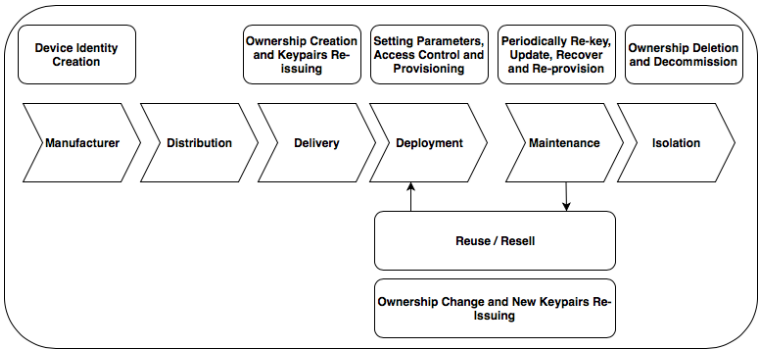


Figure 5.2 *Lifecycle of IoT devices in security management*

using the widely known security threats and vulnerabilities. Nowadays, the Internet of Everything (IoE) is developing a haystack that includes a number of valuable forensics artifacts to identify, collect, preserve and report evidence or attacks, a challenging endeavor in the environments. [395] identifies the major forensics threats alongside proposing the potential promising solutions for them to pave the way towards a secure and forensically sound deployment of IoT networks. This area still, however, needs to be matured enough to address such an aspect in the IoT world.

In addition, IoT forensics represent various IoT-based aspects and infrastructures that have the potential of being digitally investigated using forensically accepted methods. This has been represented via three different forensic aspects: device level forensics, Cloud forensics, and network forensics [396].

- Device level forensics: it involves collecting potential digital evidence from the participating IoT devices. Evidence can be collected from the physical devices, for example, audio, video, memory, graphics, Near Field Communication (NFC) as well as other IoT devices.

- Cloud forensics: conventionally, most of the IoT based devices involved in the IoT paradigm were integrated to interact over the network via applications by sharing resources in the virtualized environment. Attacks in IoT based Cloud environments target data generated therein [397]. This is due to the evolution of the sophisticated security threats from the diverging set of devices where most data generated is being moved to the Cloud.

- Network forensics: they represent IoT based environments that bear a different kind of networks. In those environments, potential attack logs can be extracted and used to conduct the digital investigation process. This could apply to industrial or home

networks, LANs, WANs, and MANs. Any potential evidence extracted from such kind of environments may be used to develop a hypothesis that can be used in a court of law following an IoT- based investigation.

5.6.9 Unexpected data usage

With the widespread usage of ubiquitous computing, that is enabled by IoT technologies, deployment of Internet-connected sensors in modern-day living has been noticeably pervasive. Recently, a few research efforts have attempted to highlight the unexpected usage of different types of user/environment- related data gathered by Internet-connected sensors [398–401]. For instance, McKenna et al. provided a list of privacy-sensitive information, such as - number of residents, daily routine and habits that can be inferred from smart homes' electricity load data collected by smart meters [398]. In spite of the presence of such previous efforts, the extent of private information that could be inferred from presumably non-critical data is either not well-known or well-understood yet.

5.6.10 Resources limitations and computational complexity

IoT devices are resource-constrained devices. Resources, such as memory, computation, and energy, which are required for ML and DL deployments, are limited and created an important bottleneck in the adoption of ML and DL for real-time onboard implementations [402]. Existing current solutions of computational offloading and execution in the Cloud suffer from high wireless energy overhead. In addition, the availability of applications for such solutions is based on network conditions. As a consequence, if the network connectivity is weak, the Cloud offloading will be unattainable and that leads to the unavailability of the existing applications.

Another solution that may advance the implementation of ML and DL for IoT security is the development of edge computing GPUs (mobile GPU), however, GPUs on mobile handles can still consume considerable mobile battery reserves [402]. On the one hand, enhancing GPU-based solutions and suggesting efficient offloading strategies are truly important in the advancement and implementation of ML- and DL-based IoT security, this will boost the performance of IoT DL applications in IoT systems with Cloud and Edge computing [346]. On the other hand, developing real-time detection and protection systems are crucial to provide effective security mechanisms for large-scaled IoT systems. Hence, reducing computational complexity holds practical importance in future research directions.

5.6.11 Interoperability of security protocols

For standardizing a global security mechanism for IoT, the protocols implemented at different layers need to interoperate by providing conversion mechanisms. Within the global mechanism, an effective combination of security standards at each layer can then be defined through consideration of architectural constraints.

5.6.12 Single points of failures

With the heterogeneous networks, architectures, and protocols, the IoT paradigm becomes more vulnerable to single points of failure than any other paradigm. A significant numbers of research work yet needs to be carried out to ensure adequate availability of IoT elements, especially for mission-critical applications. It would require mechanisms and standards to introduce redundancy while keeping in view the trade-off between costs and the reliability of the entire infrastructure.

5.6.13 Trusted updates and management

One of the open issues for future research is providing scalable, trusted management and updates of software to millions of IoT smart devices. Furthermore, the issues related to secure and trusted governance of IoT devices ownership, supply chain, and data privacy are open research problems that need to be addressed by the research community to foster a wide and massive scale adoption for IoT. Blockchain technology can be an enabler for such IoT security solutions, however, the technology itself poses research challenges to be tackled with regards to its scalability, regulations/arbitration, and key collision.

5.6.14 Blockchain vulnerabilities

Despite providing robust approaches for securing IoT, the blockchain systems are also vulnerable [403]. The consensus mechanism depending upon the miner's hashing power can be compromised, thereby allowing attackers to host the blockchain. Similarly, the private keys with limited randomness can be exploited to compromise the blockchain accounts. Effective mechanisms yet need to be defined to ensure the privacy of transactions and avoid race attacks which may result in double-spending during transactions.

5.6.15 Hardware/firmware vulnerabilities

With the low-cost and low-power devices becoming ubiquitous, the IoT architecture may become more exposed to hardware vulnerabilities. It is not just the physical malfunctioning, instead, implementation of security algorithms in hardware, routing, and packet processing mechanisms also need to be verified before deployments in IoT. Any vulnerabilities exploited after deployment become difficult to detect and alleviate. A standard verification protocol is, hence, an essential requisite for harnessing IoT security.

5.6.16 ML and DL Challenge

– Possible misuse of ML and DL algorithms by attackers (breaking cryptographic implementations by ML and DL methods).

The continuous advances in ML and DL algorithms have enabled them to be used in breaking cryptographic implementations. For example, previous studies [404, 405] used ML to break down cryptographic systems using Support Vector Machines (SVMs), which outperformed the template attack. Furthermore, another study [406] showed that RNNs can learn decryption. More specifically, an RNN with a 3000-unit long short-term memory (LSTM) can learn the Enigma machine decryption function by learning effective internal representations of these ciphers; the results suggested that DL algorithms, such as RNNs, can capture and learn the algorithmic representation of polyalphabetic ciphers for cryptanalysis.

5.7 Chapter Summary

The emergence of the IoT paradigm has led and will continue to lead, to a number of threats and possible breaches against its security. Unfortunately, security risks are not well-recognized in this domain. In this chapter, we investigated the specific properties of the IoT, devised a comprehensive security and privacy requirements list, studied several IoT attacks and their countermeasures at each level to establish a secured IoT framework.

The main goal of this chapter is to open the door to the readership to explore the threats that could be found within the IoT system and how they can be tackled to ensure a secure platform for operations. Given the wide applicability of the IoT, the security threats should be continuously addressed by the academic research communities, the industry, as well as manufacturers in a very proactive and aggressive way. To the best of our knowledge, this is the most comprehensive study on the security issues affecting all the layers ranging from things to the back-end Cloud of IoT systems.

In the next Chapter, we investigate challenges related to developing and deploying IoT apps given the specificity of Things, edge and Cloud platforms. We investigate the thread-offs of different technologies to provide concrete guidelines for practitioners.

CHAPTER 6 DEPLOYMENTS OF SERVERFUL IOT APPS[1]

6.1 Chapter Overview

Not only IoT architectures provide almost no hints on how applications could be deployed efficiently, but also practitioners, including developers, are ill equipped when it comes to choosing the right technology for their IoT development. Thus, in this chapter, we investigate the deployments of serverful applications on top of open-sourced IoT-platforms, e.g., Eclipse IoT-Hono, using a number of container setups to provide some viable guidance and principles.

6.2 Context

IoT new pillars, e.g., smart cities, Industry 4.0, etc., require specific platform(s) to allow different components to communicate. The value of the IoT systems often correlates directly with the ability of those platforms to connect different devices efficiently and integrate them into higher-level solutions. The Eclipse IoT-Hono framework allows the provisioning of remote service interfaces to connect smart things (i.e., devices) to a back-end and interacts uniformly with them regardless of their types and–or communication protocols. There is a variety of possibilities for using the framework in production. However, such deployments decisions have important performance implications that the practitioners, including developers, are not often aware of. We step up loads in container setups to clear out the performance costs of their deployment scenarios, with the aim to provide the practitioners with guidelines to help understand the performance implications of their design and deployment decisions.

On the other hand, resources utilization of IoT-based systems and applications have become an emerging topic in IoT and software engineering research communities [407, 408]. It has complex dependencies on the IoT platforms and the various components used by the applications built on top of them, all contribute in raising the resources footprint; making resource optimization a vital and challenging problem.

6.3 Research Problem and Contribution

When building, deploying and implementing applications on top of open-sourced IoT frameworks, e.g. Eclipse IoT-Hono, practitioners must seek a compromise between choosing the

[1]Part of the content of this chapter "Kubernetes or OpenShift? Which Technology Best Suits Eclipse Hono IoT Deployments", Mohab Aly, Foutse Khomh, and Soumyaya Yacout, is published in *Proceedings of the 11th Conference on Service-Oriented Computing and Applications (SOCA)*, DOI:10.1109/SOCA.2018.00024

right deployments, resources utilization and the platform's Quality of Service (QoS) so that the maximum efficiency is acquired. Finding such compromises manually is a very daunting task; practitioners need guidelines to help them in the selection of efficient design and deployment strategies.

Currently, there are many possible deployment options for Hono in a production pipeline; it can be deployed on top of a number of container setups. However, these deployments decisions have important performance implications that the practitioners should consider carefully. Moreover, without a good knowledge of the performance deviations across different deployment platforms, it is a bit challenging to predict the impact of applications migrations across different IoT environments.

Here, in this chapter, we conduct an empirical study that aims to investigate the performance implications of deploying Eclipse Hono in two different virtual environments, i.e., containers: Kubernetes and OpenShift. We perform performance test comparisons while incorporating EnMasse, i.e., a messaging infrastructure, using the following performance metrics: CPU cores usage, Memory consumption, and Network I/O usage.

Our objective is to provide evidence to confirm or refute the efficiency of such technologies and comprehend the interplay between them. We selected Kubernetes and OpenShift for this study because the latter actually distributes Kubernetes so practitioners, including end-users, may believe that both technologies offer similar performance.

6.4 Study Design

We depict the design of our study that aims to understand the discrepancy of the container setups while deploying IoT-Hono on top of them. An overview of our case study setup is presented in Figure 6.1. We select two container setups (i.e., Kubernetes and OpenShift) which are described as good deployment practices by the Eclipse Community, and we address the following research questions:

6.4.1 Research Questions

This chapter answers the following research questions:

- **RQ7.1.: Does Eclipse Hono display similar performance(s) when bare-ly deployed on container technologies?**
- **RQ7.2.: Does EnMasse display similar performance(s) when added up on Hono to scale it up?**

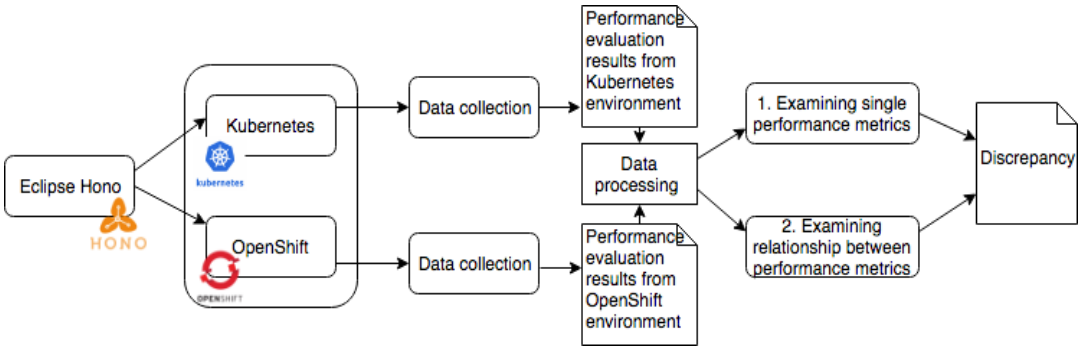


Figure 6.1 *An overview of the study design setup*

- **RQ7.3.: Do deployed serverfull applications display similar performance(s) when being added on top of both EnMasse and container technologies?**
- **RQ7.4.: To what extent does the relationship between the performance metrics change across container environments?**

To address the above-identified research questions, we carry out a number of different experiments with multiple deployments strategies to test and analyze the aforementioned performance metrics and look deeper into their behaviors. We analyzed three different versions of the selected platform, i.e., Hono, summarized in the following table 6.1. Deployments were all built from scratch each time a new analysis is performed and the results were collected by performing a series of stress tests on the platform (fixing the number of the issued requests and applications built on top of it) and tracing their executions. The same test sets were used for all experiments to ensure comparable results.

6.4.2 Environmental Setup

The performance evaluation is conducted on a Linux machine, i.e., Ubuntu 17.10, in a lab environment; this machine has an Intel i7-870 Quad-Core 2.93GHz CPU with 16GB of memory, 630GB SATA storage, 8MB Cache and is connected to a local gigabyte Ethernet cable. The hosting machine is configured to have available, Kubernetes and OpenShift, single-node clusters inside a Virtual Machine (VM). Those clusters are created via the below instances.

- Minikube: tool that helps running a single-node Kubernetes cluster inside a VM locally. This makes it easier to try Kubernetes or develop with(in) it.
- Minishift: helps running OpenShift locally by running a single-node OpenShift cluster inside a VM.

Table 6.1 *Kubernetes & OpenShift Setup Designs*

Criteria	**Experimental Designs**	
Deployments	***Kubernetes and OpenShift (KO)***	***Version***
Basic Version	Bare metal deployment of Hono	KO-0
EnMasse	Adding EnMasse on top of Hono	KO-1
EnMasse-App	Deploying apps on top of both Hono and EnMasse	KO-2

Since our goal is to compare the performance metrics in such created clusters, we set up the VM instances using enough resources so that the deployment becomes successful, i.e., 4CPUs, 10GB of Memory and 30GB of disk-size. Default instants' configurations provide a small subset of the host machine resources, in turn, it may not be sufficient to allow the instances to start correctly. That's why it is recommended to scale up the resources whenever possible, "the more resources used, the better", based on the physical machine's capabilities.

6.4.3 Design and Procedure

To assess the benefits and trade-offs of the different deployments considered, the experimentation were orchestrated using two different types of issued requests `(REST HTTP and MQTT)`. For each type, we simulated the registered devices sending both requests simultaneously in a telemetry fashion (i.e., not to expect a response in return). Each experiment was performed five times (with the number of devices being incremented gradually after each simulation: starting with 10 devices until 100 registered devices – 100 is the limit for this platform by default) to obtain min, max, and average values of the resources been consumed (i.e., a total of 450 readings for each simulation were recorded). We chose to repeat each simulation five times to mitigate the effect of variabilities (that are common in virtual environments) on our results. Table 6.1 shows the three deployment versions of the platform, the basic version KO-0 don't use any additional overhead, just bare metal deployment of Hono on top of both Kubernetes and OpenShift.

6.4.4 Performance Tests and Evaluations

Minikube and Minishift are released with `DOCKER_HOST` environment variable to point to the `Docker daemon` running inside the virtual instances; such `daemon` is used to have the final `Docker images` available inside the Minikube/Minishift VMs and make them ready for Hono's deployment. Both deployments provide access to the platform by means of different *services*, the main ones are:

1. dispatch router: router network for business applications to consume data.
2. mqtt-adapter: protocol adapter for publishing telemetry data and events using the `MQTT` protocol.
3. rest-adapter: protocol adapter for publishing telemetry data and events using the `HTTP` protocol.
4. service-device-registry: component for registering and managing devices.

To ensure the consistency between the performance tests, we destroy the environments (VMs) and restart them after every single experiment. This ensures that the formed clusters remain healthy.

Performance Metrics

We applied heapster[2] and Prometheus[3] to record the values of the performance metrics. On one hand, Heapster enables container cluster monitoring as well as performance analysis for Kubernetes; it collects and interprets various signals, such as compute resources utilization. Whereas on the other hand, Prometheus, is a service monitoring system. It collects metrics from configured targets at a given time interval, evaluates, displays the results, and triggers alerts if some conditions are observed to be true. We ran both monitoring tools on each of the constructed clusters then performed our statistical analyses on the collected data. In addition, we recorded the metrics with an interval of "starting the cluster within the VMs until the destroy phase".

System Throughput

We used the collected measurements to calculate the minimum, maximum and average values of the systems' resources by measuring the number of telemetry messages, `HTTP` and `MQTT`, sent from each registered device while adopting two serverful applications on top of Eclipse Che, e.g., Java and Nodejs apps, on top of the messaging as a service infrastructure, enMasse. The Java app is a "Hello World" app just to warm up the pods, and the latter is about simulating the workload generated from the platforms themselves. The aim is to combine the performance metrics and system throughput while minimizing their gathering noise. The combination is based on the time stamp – on a per-minute basis; a similar approach has been applied to address mining performance metrics challenges in [140].

[2] `https://github.com/kubernetes/heapster`
[3] `https://prometheus.io`

6.4.5 Hypotheses

To answer our research questions, we formulate the following null hypotheses, KO-x ($x \in \{0, 2\}$), and KO-0 is the basic version of the platform described in Table 6.1:

- HR_x^1: there is no difference between the amount of CPU/Memory/Network consumed by Hono's design when deployed on top of both Kubernetes and OpenShift.
- HR_x^2: there are no differences in the utilized resources consumed by the messaging infrastructure, i.e., EnMasse when being added on top of Hono.
- HR_x^3: there is no difference between the amount of resources consumed by serverful applications when deployed on top of both Hono and EnMasse's backend.

6.4.6 Analysis Method

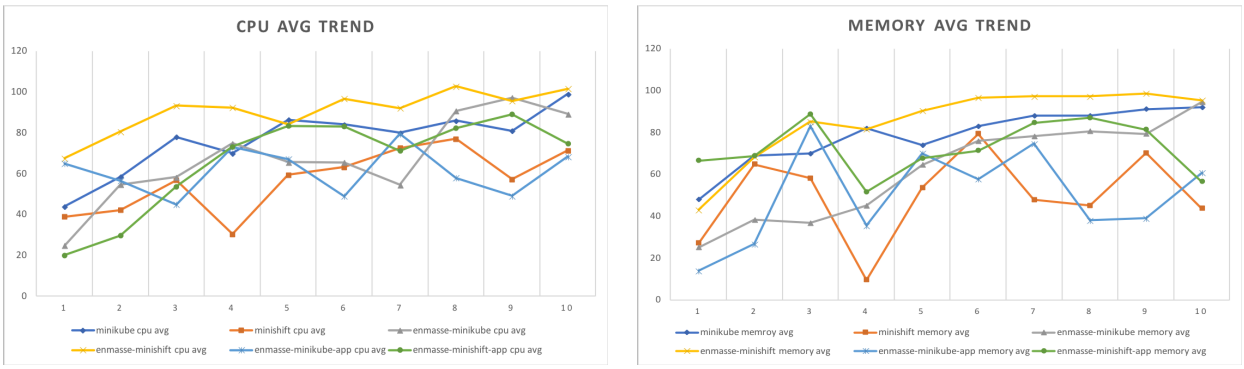
We performed the Wilcoxon test [409] to accept or reject HR_x^1, HR_x^2 and HR_x^3. We also computed the Cliff's δ effect size [410] to quantify the importance of the differences obtained between metrics values. All the tests are performed using a 95% confidence level (i.e., p-value ≤ 0.05).

A p-value that is less than or equal to (0.05) indicates that the outputted results are statistically significant. In this occurrence, we reject the null hypothesis (i.e., two populations are from the same distribution) and accept the alternative hypothesis that helps to state whether the performance metrics in the Kubernetes and OpenShift environments have the same distribution. We chose the Wilcoxon test because such a technique does not make any assumptions on the distribution of the metrics. *When it happens to have a statistically significant value that is $\leq$ (or just very close) to the value determined, we depict both its median and effect size (EF) values to highlight which technology performs best and which one is the worst for such kind of deployment(s).*

Wilcoxon test is a non-parametric statistical test that assesses whether two independent distributions and–or trends are the same. Cliff's δ is a non-parametric effect size measure that represents the degree of overlap between two sample trends [410]. A Wilcoxon test ranges from -1 (when it happens that all selected values in the first group are larger than the second one) to +1 (if all selected values in the first group are smaller than that of the second group). It is zero when the two sample trends are identical [411]. *A Cliff's δ effect size is considered negligible if it is < 0.147 , small if < 0.33, medium if < 0.474, and large if ≥ 0.474.*

Table 6.2 *p-value of Wilcoxon Test (p-VAL) – (Median Kubernetes, Median OpenShift) – (Cliff δ Effect Size (ES))*

# Devices	CPU cores usage (p-value)			Memory usage (p-value)			Network consumption (p-value)		
	min	max	avg	min	max	avg	min	max	avg
HONO Bare metal Deployment KO-0									
10	0.6065	0.8413	0.8413	0.5269	0.2948	0.2101	1	0.6762	1
20	0.3095	0.492	0.4206	0.4034	0.4034	0.4206	0.5476	0.6065	0.2476
30	0.2673	0.4206	0.4206	0.2101	0.1437	0.2222	**0.03175** *(70, 0.138) ; ES=0.84*	0.05556 *(70, 33.38, ES=0.76)*	0.0601081
40	0.05556 *(58, 42.997) ; ES=0.76)*	0.09524	0.09524	**0.007937** *(70, 4.169) ; ES=1*	**0.02157** *(91, 16.02) ; ES=0.92*	**0.01587** *(84, 9.58) ; ES=0.92*	**0.02157** *(70, 43.47) ; ES=0.92*	**0.03175** *(95, 63.34) ; ES=0.84*	0.0660284
50	0.09369	0.09524	0.09524	0.09369	0.09449	0.3636 *(68, 42.13, ES=0.76)*	**0.02157** *(76, 43.47) ; ES=0.92*	**0.03571** *(103, 30.57) ; ES=0.84*	**0.02137175** *(83, 55.83) ; ES=0.92*
60	0.2222	0.2101	0.2222	0.6761	0.5269	0.834	0.09369	**0.03571** *(103, 30.57) ; ES=0.84*	0.09369294
70	0.6065	0.6065	0.6065	0.1437	0.09449	0.0601	**0.0469** *(85.5, 57.04) ; ES=0.68*	0.1508	0.09524
80	0.09524	1	0.5508	**0.02157** *(90, 19.44) ; ES=0.92*	**0.01219** *(99, 50.1) ; ES=1*	**0.007937** *(68, 15.21) ; ES=1*	**0.01587** *(90, 52.57) ; ES=0.92*	**0.02157** *(97, 75.42) ; ES=0.92*	**0.02157175** *(90.82, 61.03) ; ES=0.92*
90	0.4206	0.09369	0.2222	0.0601	0.0601	**0.03571** *(88, 70.25) ; ES=0.84*	0.2101	0.222	0.230075
100	0.09524	0.3065	0.09524	0.2063	0.2101	0.02222	0.1437	0.4034	0.1467272
HONO-EnMasse KO-1									
10	0.1508	0.09469	0.1467721	0.5508	0.2063	0.230075	**0.01587** *(72.08, 61.60) ; ES=0.92*	**0.007937** *(97.72, 74) ; ES=1*	**0.01218578** *(77.081, 60.46) ; ES=1*
20	**0.03175** *(98, 367, 71.787) ; ES=0.84*	0.1437	0.5508	0.1437	0.2101	0.1467272	**0.02157** *(97.81, 62.18) ; ES=0.92*	0.09469	**0.03671386** *(45.669, 68.88) ; ES=0.84*
30	0.222	0.5508	0.1437	0.09524	0.1437	0.09369294	**0.01587** *(44.88, 73.10) ; ES=0.92*	**0.01219** *(60.47, 88.870) ; ES=1*	**0.02157175** *(59.839, 81.88) ; ES=0.92*
40	0.1437	0.09449294	0.9524	0.05556 *(58.09, 75.34) ; ES=0.76*	0.09449	0.09524	0.09524	0.09524	0.1467272
50	0.1467272	**0.01219** *(75.719, 89.904) ; ES=1*	0.9469	0.5508	0.09524	0.1467272	0.2222	0.14034	0.230075
60	0.09369	0.1425	0.09369	0.2222	0.2063	0.4034363	0.09524	0.09601	0.06011281
70	0.1508	0.09469294	0.1437	0.09469	0.09449	0.09524	0.09469	0.09469	0.1467272
80	**0.01587** *(81.704, 92.705) ; ES=0.92*	0.1437	**0.03671** *(90.709, 102.855) ; ES=0.84*	0.1437	0.4034	0.1467272	0.09524	0.09601	0.09469294
90	1	0.5269	1	0.09469	0.09524	0.9469294	0.5476	0.230075	0.2962699
100	0.09524	**0.01218578** *(98.367, 110.456) ; ES=1*	**0.007937** *(69.347, 110.461) ; ES=1*	0.9065	0.6761	1	0.1425	0.1437	0.1467272
HONO-EnMasse App KO-2									
10	0.09524	0.09469	0.09469294	0.05556 *(5.06, 26.79) ; ES=-0.76*	0.09449	0.06011281	0.4506	0.09601	0.09469294
20	0.09469	0.1437	0.9469	0.4206	0.2063	0.4034363	0.8413	0.6761	0.6065
30	1	0.4034	0.4034363	0.8413	1	1	1	0.6065	0.6761033
40	0.6065	0.8345	0.8413	0.5476	0.6761	0.5369683	0.4206	0.2101	0.4034363
50	0.4206	0.4034	0.4206	0.5269	0.8345	0.8345306	0.4034	0.5309	0.6761033
60	0.05556 *(36.852, 74.02) ; ES=-0.76*	0.1437	0.09469294	0.6065	0.8345	0.84113	0.6065	0.8345	0.6065
70	0.6065	0.5269	0.5476	0.2222	0.2063	0.230075	0.8413	0.6761	0.6761033
80	0.1508	0.09469	0.9524	0.5476	0.4034	0.4034363	0.8345	0.8413	1
90	0.0601	0.09524	**0.03671386** *(0.108, 89.158) ; ES=0.84*	1	1	1	0.3065	0.4034	0.4034363
100	0.5476	0.8345	0.6761033	0.6761	0.6065	0.5369683	0.6761	0.5309	0.8345306



(a) *Median trend of the CPU avg* (b) *Median trend of the Memory avg*

Figure 6.2 *Results obtained for CPU and Memory trends*

6.5 Study Results

This section presents and discusses the results of our research questions. Table 6.2 summarizes the results of the Wilcoxon test, median values and Cliff's δ effect size for each performance metric. Significant results are marked in **bold**.

- **RQ7.1.: Does Eclipse Hono display similar performance(s) when bare-ly deployed on container technologies?**

Results of Table 6.2 show that there is no statistically significant difference between the overall **CPU cores usage** while deploying Hono on top of Kubernetes and OpenShift, hence we cannot reject HR_x^1 for KO-x ($x \in \{0..2\}$) for the CPU usage. However, effect size values and Figures 6.2a, 6.2b, and 6.3 – (i.e., samples) show that the trend of the CPU, in Kubernetes, is slightly larger than that of the OpenShift in the simulations conducted for the bare metal deployments. The trend tends to fall down towards the statistically significant difference value of 0.05, where Kubernetes is greedy while consuming its CPU to handle Hono's deployment on top of it.

In addition to the previous observation, results also show that there is a statistically significant difference between the **Memory usage** for the container technologies. Furthermore, we obtained statistically significant results with the **Network consumption**, for all Kubernetes deployments in contrast to OpenShift (i.e., all effect sizes are large). Hence we reject HR_x^1 for all KO-x ($x \in \{0..2\}$) for Memory and Network usages. We explain such phenomenon by the overhead induced by Kubernetes to be able to cope up with the processes being issued within the system, i.e., setting up the VM, building `DOCKER` images, deploying Hono Platform as well as sending telemetry messages from the participating devices.

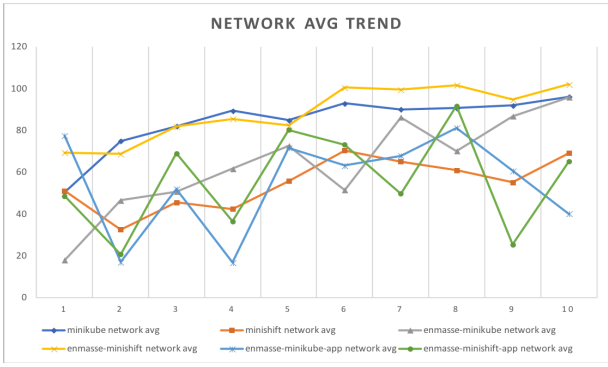


Figure 6.3 *Median trend of the Network I/O avg*

- **RQ7.2.: Does EnMasse display similar performance(s) when added up on Hono to scale it up?**

Results from table 6.2 show that the addition of the messaging infrastructure, EnMasse, on top of the container technologies does affect the overall **CPU cores usage** of the platform as well as the **Network consumption**. This addition is statistically significant for OpenShift, therefore we can reject HR_x^2 for KO-x ($x \in \{0..2\}$). Effect size values (i.e., large) as well as Figures 6.2a, 6.2b, and 6.3 show the impact of EnMasse on the performance metrics when combined with Hono to scale up the platform.

Regarding the **Memory usage**, we did not obtain a significant difference between Kubernetes and OpenShift, when performing the deployment of Hono; hence we cannot reject HR_x^2 in this case for the memory usage. However, figures, as well as effect size values, show that the trend of Memory usage, in OpenShift, is larger than that of Kubernetes in the simulations performed while attempting to add EnMasse on top of Hono. The trend tends to approach the statistically difference value of 0.05, where OpenShift utilizes more memory to sustain the messaging infrastructure on top of it.

- **RQ7.3.: Do deployed serverful applications display similar performance(s) when being added on top of both EnMasse and container technologies?**

Results from table 6.2 show that the deployment of the two serverful applications on top of the messaging infrastructure, EnMasse, and the container technologies do affect the overall **CPU cores usage** of the platform. Such overhead is statistically significant for OpenShift as well, therefore we can reject HR_x^3 for KO-x ($x \in \{0..2\}$). Also, depicted figures and effect size values (i.e., large) show the impact that adding applications to EnMasse and Hono has on the performance metrics.

Regarding the **Memory usage**, the trend tends to also lean towards the statistically significant difference value 0.05, where OpenShift consumes more memory to allow the deployment of applications. On the other hand, for the **Network consumption**, we did not notice any significant difference between both container technologies, when allowing the deployment of applications on top of the platform. Hence, in these cases, we cannot reject HR_x^3 for each of the memory and network consumptions.

- **RQ7.4.: To what extent does the relationship between the performance metrics change across container environments?**

The relationship between performance metrics may significantly change and be different between environments, which may be a glimpse of system regression or performance issues. As of [412], combinations of performance metrics are more predictive towards performance issues than a single metric. A change in such combinations can pose discrepancy of performance and help practitioners identify the behavioral changes of a system between different environments. For instance, in one system, the CPU may be correlated to a great extent with network (e.g., when network's operations are high due to the workload being generated, eventually CPU is high to accommodate such increase); on the other hand, on the same system, the correlation between CPU and memory may become low. Such change identified may expose performance issues (i.e., high CPU without memory and–or network I/O operations might be due to a performance failure). For example, in our experiments, we experienced lots of unready pods as they had been running for more than five minutes and had not passed their readiness check, hence, we destroyed the formed clusters and started all over again.

However, if there is a significant difference in correlations simply due to the platform being used, i.e., Kubernetes vs. OpenShift, then practitioners may need to be warned that a correlation discrepancy may be false. Hence, we examined whether the relationship between performance metrics has a discrepancy between both container setups.

Approach

We measured the Spearman's rank correlation coefficient among all performance metrics in the container setups and studied their behavior and whether they are different. Spearman's correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data. In a given sample, it is denoted by r_s and constrained to -1 $\leq r_s \leq$1. It is interpreted as the closer r_s to $\pm$1 the stronger the monotonic relationship.

Correlation is an effect size (ES) measure and the strength of the correlation for the r_s is described as:

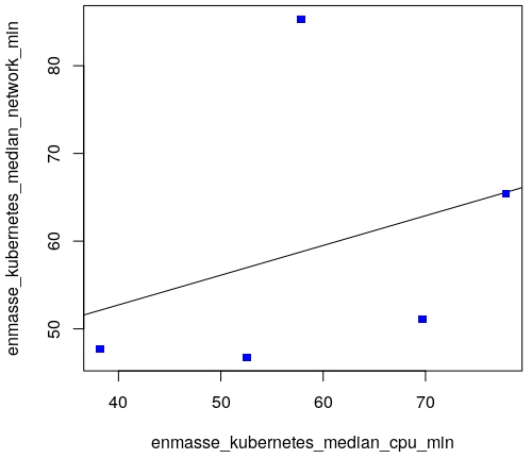


Figure 6.4 *Correlation changes for Kubernetes-Hono-EnMasse (CPU–Network) min.*

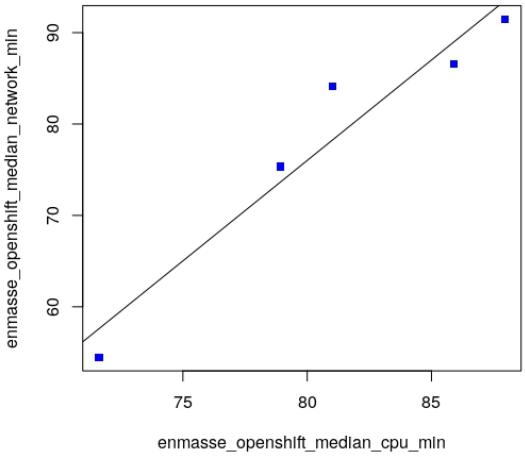


Figure 6.5 *Correlation changes for OpenShift-Hono-EnMasse (CPU–Network) min.*

0.00-0.19 “very weak", 0.20-0.39 “weak", 0.40-0.59 “moderate", 0.60-0.79 “strong", and 0.80-1.0 “very strong".

Discussion

There exist differences in correlation between the performance metrics in Kubernetes and OpenShift. Tables 7.7, 7.8, and Figures 6.4 and 6.5 (i.e., samples) show the changes in the correlation coefficient among the resources utilized in both environments. By closely looking at them, we find that in bare metal deployments, (CPU–Network) in Kubernetes has stronger correlation than that of OpenShift (i.e., noticeable network’s operations due to Hono’s deployment workload); whereas (Memory–Network) coefficients in OpenShift are stronger than in Kubernetes. Furthermore, adding EnMasse results in stronger coefficients in Kubernetes, but that is not the case in OpenShift where all performance metrics show strong coefficients to handle such addition on top of Hono (i.e., resources have been utilized fiercely where EnMasse has its own additional processes to be run on top of Hono).

For the sake of brevity, we do not show the detailed analysis here which can be consulted in our replication package indicated in the next Chapter, Section 7.6. Box-plots, as well as Spearman’s correlation trend-lines, are depicted there-in to show the correlation changes among related metrics.

Table 6.3 *Spearman's rank correlation summary of performance metrics in Kubernetes*

# Devices	CPU–Memory			CPU–Network			Memory–Network		
	min	max	avg	min	max	avg	min	max	avg
HONO Bare metal Deployment KO-0									
10 –> 100	-0.1025978	-0.1	-0.3077935	**0.6 (strong)**	**0.7 (strong)**	**0.6 (strong)**	0.2051957	0.1	0.2051957
HONO-EnMasse KO-1									
10 –> 100	0.1	0.1	0.2	**0.6 (strong)**	**0.6 (strong)**	-0.1	0	0	-0.1
HONO-EnMasse-App KO-2									
10 –> 100	-0.2	-0.3	-0.2	**0.7 (strong)**	**0.7 (strong)**	**0.6 (strong)**	0.2	0.3	0.5

Table 6.4 *Spearman's rank correlation summary of performance metrics in OpenShift*

# Devices	CPU–Memory			CPU–Network			Memory–Network		
	min	max	avg	min	max	avg	min	max	avg
HONO Bare metal Deployment KO-0									
10 –> 100	**0.6 (strong)**	0.1	0.3	-0.1	0.1	0.1	-0.3	**0.8 (v. strong)**	**0.6 (strong)**
HONO-EnMasse KO-1									
10 –> 100	**1 (v. strong)**	**0.6 (strong)**	**0.8 (v. strong)**	**1 (v. strong)**	**0.6 (strong)**	**0.9 (v. strong)**	**1 (v. strong)**	**1 (v. strong)**	**0.9 (v. strong)**
HONO-EnMasse-App KO-2									
10 –> 100	0.1	0.4	0.5	**0.6 (strong)**	**0.8 (v. strong)**	**0.6 (strong)**	0.3	-0.1	-0.1

6.6 Chapter Summary

Performance assurance activities are critical in enforcing software and systems' reliability. The discrepancy between performance testing in Eclipse's IoT container technologies hasn't been attempted to be evaluated yet, hence, we studied whether there are differences between Kubernetes and OpenShift environments while handling different Hono deployments. In this chapter, we step up loads to examine the performance costs related to containers deployment scenarios.

By examining the results, we find that there exists discrepancies between performance metrics while considering three different modes: (1) Bare metal deployment of Hono on top of Kubernetes and OpenShift. (2) Scaling out Hono and incorporating EnMasse (as a messaging as a service) infrastructure to the architecture. (3) Adding serverful applications to the platform so that we have the complete paradigm of the system.

Such results not only provide important guidelines for building and deploying Eclipse Hono, EnMasse and developing IoT based applications, but also aim to provide an understanding of the performance implications of their design so that practitioners, including end-users and developers, can make right deployment decisions.

The main contributions of this chapter include:

- This study is one of the first attempts to evaluate the discrepancy in the context of analyzing performance testing in Eclipse IoT–Hono.
- We found unbalanced relationships among related performance metrics (i.e., ranging between strong, moderate and weak) between Kubernetes and OpenShift environments. Therefore, practitioners cannot assume a straightforward overhead from container setups (such as a simple increment of CPU).
- Our findings highlight the need to be aware of and to reduce the discrepancy between performance testing results in container setups (i.e., especially in open-sourced platform(s)), such as Eclipse-IoT Hono.

In the next Chapter, we continue to investigate the previous conceptual problem related to the challenges of developing and deploying serverless IoT apps. We study the thread-offs of different technologies to provide them concrete guidelines.

CHAPTER 7 DEPLOYMENTS OF SERVERLESS IOT APPS[1]

7.1 Chapter Overview

Practitioners need to select optimal deployment strategies for their applications in order to avoid dysfunctional performance hiccups. They ought to know the performance implications of deploying an IoT applications using a serverless infrastructure in comparison to deployments performed on an origin server, i.e., as indicated in the previous chapter so that the behavior is well observed.

7.2 Context

Serverless computing provides a small run-time container to execute lines of codes; relieving developers from the management of the underlying infrastructure. In a serverless computing architecture, all administration operations are handled by cloud service providers. Thanks to its lightweight nature, ease of management, and ability to scale quickly, serverless computing is becoming a top choice for application development in the IoT community. However, like every new technology, there are limitations as well as advantages. For example, the highly-distributed, loosely coupled nature of serverless applications can cause latency issues. Therefore, the advantages of a serverless architecture may come with a performance cost.

To clear up this suspicion, this chapter examines the performance implications of deploying an IoT application using a serverless infrastructure. Leveraging the results of our analysis, we aim to provide guidelines to help practitioners select optimal deployment strategies for their applications.

7.3 Research Problem and Contribution

Serverless computing makes it easy to patch, fix, or add new features to an application, since the application is broken up into separate, smaller functions. Using a serverless infrastructure, developers can run different functions of the applications on different servers located closer to their users. They can easily scale resources up or down by controlling the number of functions that are deployed on different nodes. However, the flexibility of the serverless architecture may come with a performance cost. In fact, because multiple parts of the code are not

[1]Part of the content of this chapter "On the Performance Implications of Deploying IoT Apps as FaaS", Mohab Aly, Foutse Khomh, and Soumaya Yacout, is submitted to *Special Issue on Cloud Computing in the IoT Revolution, Internet of Things Journal (IoTJ)*, Under review as of August 2019.

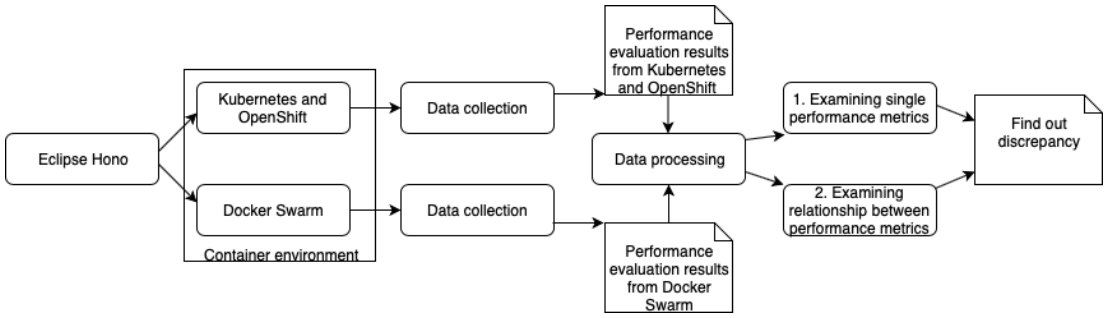


Figure 7.1 *An overview of our case study setup*

constantly running, they may need to 'boot up' every time they are invoked. This startup time may degrade the performance of the application. Furthermore, the highly-distributed, loosely coupled nature of serverless applications can induce latency issues. Hence, we investigate the performance implications of deploying an IoT application using a serverless infrastructure. We also examine the effect that a serverless architecture can have on performance testing operations.

7.4 Study Design

The aim of our study is to understand the performance implications of different serverless application deployments scenarios under various container setups. In particular, we are interested in understanding the impact of a serverless architecture on the performance of applications deployed on Eclipse Hono using different containers technologies. A load driver is used to exercise our subjected platform; after collecting and processing the performance metrics, we analyze and draw the conclusion based on (1) single performance metrics, and (2) relationship between performance metrics. An overview of our case study setup is depicted in Figure 7.1. We opted for three container setups (i.e., Kubernetes, OpenShift, and Docker Swarm) which are recommended for applications deployment in the Eclipse Foundation Community.

7.4.1 Research Questions

This chapter answers the following research questions:

- **RQ8.1.: Does Eclipse Hono display similar performance(s) when deployed on top of Kubernetes, OpenShift, or Docker Swarm setups?**

- **RQ8.2.: o serverless applications deployed using Eclipse Hono display similar performance(s) in Kubernetes, OpenShift and Docker Swarm setups?**
- **RQ8.3.: To what extent does the relationship between the performance metrics change across Kubernetes, OpenShift, and Docker Swarm setups?**

To answer these three research questions, we performed a series of experiments with multiple deployments configurations to collect and analyze differences between performance metrics. Table 7.1 provides details about our configuration(s); the serverless application has a workload generator that can be used to generate a synthetic container workload(s), it attempts to generate performance records at a relatively steady rate across the specified container configuration(s). All deployments were built/deployed from scratch each time a new analysis is conducted, the results were collected by performing a chain of stress tests on the platform (i.e., fixing the number of issued requests and applications built on top of it) and tracing back their executions. The same set of configurations and–or tests were used for all experiments to ensure comparable results. The rest of this section provides more the details about our experiments.

7.4.2 Environmental Setup

The performance evaluation is conducted on a Linux machine, i.e., Ubuntu 18.04, in a lab environment; this machine has an Intel i7-870 Quad-Core 2.93GHz CPU with 16GB of memory, 630GB SATA storage, 8MB Cache and is connected to a local gigabyte Ethernet cable. These are the same configurations as in the previous chapter 6, but the hosting machine, this time, is configured to have available Kubernetes, OpenShift, and a cluster of Docker Engine nodes running in Swarm mode. The main deployment target for the Swarm mode, in this study, is a multi-node cluster running on Amazon Web Services (AWS). We created the clusters as follows:

- Amazon Web Services (AWS): platform that helps running Docker Swarm Clusters on top of the Cloud. Eclipse Hono components are distributed by means of `Docker images` which can be deployed to arbitrary environments where Docker is available (i.e., Docker Swarm on AWS).

Since our goal is to compare the performance metrics in such created clusters, we set up the Swarm stack using five Swarm manager nodes and 10 worker nodes; this was the limit for the free tier we were using. On the other hand, and as previously stated, for both Kubernetes

Table 7.1 *Setup Design on Amazon (AWS)*

Criteria	Experimental Designs	
Deployments	Kubernetes, OpenShift–(KO), and Docker Swarm- (DS)	Version
Basic Version	Bare metal deployment of Hono on Kubernetes, OpenShift, and Docker Swarm	KO-0 & DS-0
Hono-Serverless app	Deploying serverless apps on top of Kubernetes, OpenShift, and Docker Swarm	KO-1 & DS-1

and OpenShift, we setup the VM instances using enough resources so that the deployment becomes successful, i.e., 4CPUS, 10GB of Memory and 30GB of disk-size.

Container Environments

Alongside the configurations we opted for in the previous chapter for both Kubernetes and OpenShift (i.e., Minikube and Minishift), we considered this additional setup for Docker Swarm:

- Docker Community Edition (CE) for Amazon Web Services (AWS) – stable – that uses existing Virtual Private Cloud (VPC) on our Cloud infrastructure for the Docker Swarm setup.

7.4.3 Design and Procedure

Table 7.1 shows the three deployment versions of the platform, the basic versions KO-0 & DS-0 do not use any additional overhead, just the deployment of Hono on top of Kubernetes, Openshift, and a cluster of Docker Engine nodes running in Swarm mode with the aforementioned set up configurations as illustrated in the previous chapter 6.

7.4.4 Performance Tests and Evaluations

System Throughput

We used the collected measurements to calculate the minimum, maximum and average values of the systems' resources by measuring the number of telemetry messages, `HTTP and MQTT`, sent from each of registered device while adopting the serverless application on top of the messaging infrastructure, Amazon Web Services (AWS).

7.4.5 Hypotheses

To answer our research questions, we formulated the following null hypotheses, KO-x & DS-x ($x \in \{0, 1\}$), KO-0 & DS-0 are the basic versions of the platform described in Table 7.1:

- HR_x^{1-0}: there is no difference between the values of the performance metrics, *CPU Cores*, obtained for Hono's design when deployed on Kubernetes, OpenShift, and Docker Swarm setups.
- HR_x^{2-0}: there is no difference between the values of the performance metrics, *Memory Usage*, obtained for Hono's design when deployed on Kubernetes, OpenShift, and Docker Swarm setups.
- HR_x^{3-0}: there is no difference between the values of the performance metrics, *Network Consumption*, obtained for Hono's design when deployed on Kubernetes, OpenShift, and Docker Swarm setups.
- HR_x^{1-1}: there are no differences in the utilized resources, *CPU Cores*, consumed by the serverless workload generator app deployed on top of Hono's Kubernetes, OpenShift, and Docker Swarm setups.
- HR_x^{1-2}: there are no differences in the utilized resources, *Memory Usage*, consumed by the serverless workload generator app deployed on top of Hono's Kubernetes, OpenShift, and Docker Swarm setups.
- HR_x^{1-3}: there are no differences in the utilized resources, *Network Consumption*, consumed by the serverless workload generator app deployed on top of Hono's Kubernetes, OpenShift, and Docker Swarm setups.

7.4.6 Analysis Method

We performed the Wilcoxon test to accept or reject HR_x^{y-0} and HR_x^{y-1}, where y $\in \{1,2,3\}$. We also computed the Cliff's δ effect size to quantify the importance of the differences obtained between metrics values. All the tests were performed using a 95% confidence level (i.e., p-value ≤ 0.05).

7.5 Study Results

This section presents and discusses the results of our research questions. Tables 7.2, 7.3, 7.4, and 7.5 summarize the results of the Wilcoxon test, median values and Cliff's δ effect size

values for each performance metric. Significant results are marked in **bold**.

- **RQ8.1.: Does Eclipse Hono display similar performance(s) when deployed on top of Kubernetes, OpenShift, or Docker Swarm setups?**

Motivation In this research question, we want to investigate the performance implications of scaling the Eclipse Hono framework using different container setups. If there is a significant performance implication, we will examine how the deployment of serverless applications is impacted by it.

Approach Since performance metrics are likely to have different value ranges because of differences in the architecture of the platforms (Kubernetes/OpenShift environment(s) may have a higher network I/O consumption than Docker Swarm), instead of comparing the values of each performance metrics across the different environments, we instead study whether the trends of the metrics are the same.

Results Tables 7.2 and 7.3 show that there are statistically significant differences between all the **CPU cores usages**, **Memory usages** and **Network throughput** when deploying Hono on top of both Kubernetes and OpenShift rather than on top of Docker Swarm. We, therefore, reject HR_x^{y-0} for KO-x & DS-x ($x \in \{0..1\}$), where y $\in \{1, 2, 3\}$ for all the performance metrics identified. Both container setups are greedy while consuming their resources to (1.) cope up with the processes being issued within the system, i.e., setting up the VMs, building `Docker` images, deploying Hono platform, and (2.) sending telemetry messages from the participating devices.

Observation

In both Kubernetes and OpenShift, containers are not allowed to be used above their CPU and–or memory limits. If a container, in such setups, allocates more resources than their limits, it becomes a candidate for termination. Thus, if the container continues to seize resources beyond its limits, it is then terminated to ensure that the environment is properly configured and formulated. And in our case here, both of the above mentioned setups hit the limits several times leading to the excessive resources utilization until the complete deployment has occurred successfully by adding up more resource specs. This was not the case while dealing with Docker Swarms on Amazon Web Services (AWS), where the Cloud infrastructure is scaled in–out based on the workloads generated by the platform and deployed application(s).

Table 7.2 *p-value of Wilcoxon Test (p-VAL) – (Median Kubernetes, Median Docker Swarm) – (Cliff δ Effect Size (ES))*

HONO AWS-Swarm Deployment DS-0									
# Devices	CPU cores usage (p-value)			Memory usage (p-value)			Network consumption (p-value)		
	min	max	avg	min	max	avg	min	max	avg
10	0.1508	0.1508	0.1508	0.8413	0.834	0.6905	0.4206	0.5476	0.6905
20	**0.007937** ***(65, 1.77)*** ***(ES=0.68)***	0.1193	**0.007937** ***(58.5, 7.32)*** ***(ES=0.68)***	0.3095	0.2222	0.3095	0.1508	0.6905	0.6905
30	**0.01167** ***(67, 8.13)*** ***(ES=0.68)***	**0.007937** ***(88, 32.18)*** ***(ES=0.68)***	**0.007937** ***(78, 21.22)*** ***(ES=0.68)***	0.8413	0.8413	0.8413	0.2222	0.1508	**0.03175** ***(82, 39.06)*** ***(ES=0.68)***
40	**0.007937** ***(58, 18.3)*** ***(ES=0.68)***	**0.007937** ***(82, 37.28)*** ***(ES=0.68)***	**0.007937** ***(70, 28.98)*** ***(ES=0.68)***	0.4206	0.2222	0.3095	0.09524	0.1508	0.1508
50	**0.01193** ***(80, 26.02)*** ***(ES=0.68)***	**0.007937** ***(93, 45.86)*** ***(ES=0.68)***	**0.007937** ***(86.5, 34.98)*** ***(ES=0.68)***	**0.03615** ***(68, 46.98)*** ***(ES=0.68)***	**0.03175** ***(80, 64.26)*** ***(ES=0.68)***	**0.03175** ***(74, 54.84)*** ***(ES=0.68)***	0.05556 *(76, 40.98)* *(ES=0.68)*	**0.03175** ***(95, 63.27)*** ***(ES=0.68)***	**0.03175** ***(85, 53.72)*** ***(ES=0.68)***
60	**0.007937** ***(78, 35.11)*** ***(ES=0.68)***	**0.03175** ***(84.18, 53.63)*** ***(ES=0.68)***	**0.007937** ***(84.18, 44.01)*** ***(ES=0.68)***	0.05556 *(75, 3.61)* *(ES=0.68)*	0.1508	0.1425	0.09369	0.05556 *(103, 74.07)* *(ES=0.68)*	0.09524
70	0.6905	0.5476	0.6905	0.1508	0.05556 *(95, 72.65)* *(ES=0.68)*	0.09524	**0.007937** ***(85.5, 46.26)*** ***(ES=0.68)***	**0.007937** ***(98, 62.03)*** ***(ES=0.68)***	**0.007937** ***(90.17, 52.75)*** ***(ES=0.68)***
80	0.1587	0.1587	**0.03175** ***(86, 71.44)*** ***(ES=0.68)***	0.1508	0.2222	0.3095	0.09524	**0.03175** ***(97, 53.55)*** ***(ES=0.68)***	**0.03175** ***(90.82, 39.57)*** ***(ES=0.68)***
90	0.6905	0.2087	0.3095	0.09524	0.05556 *(100, 39.59)* *(ES=0.68)*	0.05556 *(91.17, 25.48)* *(ES=0.68)*	0.4206	0.2222	0.4206
100	0.4206	0.3095	0.4206	0.1508	0.09524	0.09524	0.05556 *(87, 73.37)* *(ES=0.68)*	0.2222	0.2222

Table 7.3 *p-value of Wilcoxon Test (p-VAL) – (Median OpenShift, Median Docker Swarm) – (Cliff δ Effect Size (ES))*

HONO AWS-Swarm Deployment DS-0									
# Devices	CPU cores usage (p-value)			Memory usage (p-value)			Network consumption (p-value)		
	min	max	avg	min	max	avg	min	max	avg
10	**0.007937** ***(34.682, -5.04)*** ***(ES=0.68)***	**0.007937** ***(43.212, 13.38)*** ***(ES=0.68)***	**0.007937** ***(38.914, 4.46)*** ***(Es=0.68)***	1	0.4206	0.4206	0.402	0.6752	0.6752
20	0.1508	0.1508	0.09524	0.6905	0.6905	0.6905	0.4206	1	1
30	0.1508	0.5556 *(66.327, 32.18)* *(ES=0.68)*	0.05556 *(56.849, 31.22)* *(ES=0.68)*	0.4206	0.3095	0.3095	1	1	0.4206
40	**0.007937** ***(42.991, 18.3)*** ***(ES=0.68)***	0.4206	0.3095	**0.03175** ***(2.09, 47.44)*** ***(ES=0.68)***	**0.01587** ***(16.62, 63)*** ***(ES=0.68)***	**0.01587** ***(9.58, 53.21)*** ***(ES=0.68)***	0.6905	0.6905	0.8413
50	0.1587	**0.01587** ***(64.305, 45.86)*** ***(ES=0.68)***	**0.007937** ***(59.507, 34.98)*** ***(ES=0.68)***	0.6905	0.6905	0.6905	0.8413	1	1
60	0.1508	0.3095	0.1508	0.1508	0.2222	0.2222	1	1	0.8413
70	0.1587	0.09524	0.05556 *(72.536, 55.57)* *(ES=0.68)*	0.6905	0.5476	0.6905	0.1508	0.2222	0.2222
80	1	0.2222	0.8413	0.4206	0.3095	0.3095	0.8413	**0.03175** ***(75.42, 53.55)*** ***(ES=0.84)***	0.8413
90	0.5476	0.8413	0.6905	0.4206	0.5476	0.4034	0.5476	0.6905	0.6905
100	0.8413	0.5476	0.6905	1	0.8413	1	0.3095	0.5309	0.5476

Findings: Performance metrics typically do not follow the same distribution in OpenShift, Kubernetes, and Docker Swarms environments.
Actionable implications: Practitioners should consider deploying Hono on top of Docker Swarm for better performance.

- **RQ8.2.: Do serverless applications deployed using Eclipse Hono display similar performance(s) in Kubernetes, OpenShift and Docker Swarm setups?**

Motivation Kubernetes, OpenShift and Docker Swarm are three leading containers technologies in the Industry nowadays. This research question aims to evaluate and inspect potential performance differences when serverless applications are deployed in the three different container setups.

Approach After running and collecting the performance metrics, we compare every single performance metric between the container environments. Since the performance tests are carried out in different containers' behavior, intuitively the scales of performance metrics are not the same. For example, the Kubernetes environment may have higher memory usage than in OpenShift. Therefore, instead of comparing the values of each performance metric in each environment, we study whether the distributions of the performance metric follows the same shape and the same trend across the different container environments. We perform Wilcoxon tests and compute Effect Size (ES) metrics, to capture differences between the distributions of different corresponding performance metrics.

Results Tables 7.4 and 7.5 show differences between CPU/memory usages and network consumption when our serverless application is deployed on top of the scaled out Hono in the container setups, respectively. There is a statistically significant difference between the amount of consumed **memory** for OpenShift. Therefore, we reject HR_x^{1-2} for KO-x & DS-x ($x \in \{0..1\}$). We explain this difference in memory consumption as follows. When OpenShift nodes are memory overcommitted, they run out of resources that successful deployment is hindered, such situation is called resource pressure. We experienced such behavior several times while experimenting our scenario. A number of Hono pods[2] where stuck being created leading to the "freeze" state of the whole experiment, hence, we had to destroy such deployment and start a new creation to heal such phenomenon. When OpenShift node service

[2] *hono-adapter-http-vertx-1-deploy*
hono-adapter-kura-1-deploy
hono-service-device-registry-1-deploy
hono-service-messaging-1-deploy

Table 7.4 *p-value of Wilcoxon Test (p-VAL) for Serverless Apps – (Median Kubernetes, Median OpenShift) – (Cliff δ Effect Size (ES))*

HONO-Serverless-App KO-DS									
# Devices	CPU cores usage (*p*-value)			Memory usage (*p*-value)			Network consumption (*p*-value)		
	min	max	avg	min	max	avg	min	max	avg
10	0.8345	0.6761	1	0.6761	0.8345	1	0.2963	0.2963	0.2101
20	0.2101	0.1437	0.2101	1	0.8345	1	1	1	1
30	0.2963	0.6761	0.5309	0.5309	0.4034	0.5309	0.1437	0.09469	0.09469
40	0.1437	0.09469	0.1437	0.4034	0.5309	0.4034	0.5309	0.8345	0.8345
50	0.1437	0.2101	0.1437	1	1	1	0.2101	0.2101	0.2101
60	0.1437	0.09469	0.09469	0.5309	0.6761	0.5309	1	0.6761	0.8345
70	0.09469	0.2101	0.09469	0.6761	0.6761	0.6761	0.5309	0.345	0.6761
80	0.09469	0.5309	0.4034	**0.03671** *(19.06, 59.14)*–***(ES=-0.84)***	**0.03671** *(38.9, 73.29)*–***(ES=-0.84)***	**0.03671** *(28.26, 66.43)*–***(ES=-0.84)***	0.2963	0.2101	0.2963
90	0.1437	0.09469	0.2101	0.1437	0.09469	0.09469	0.8345	0.6761	0.8345
100	0.8345	0.8345	1	0.8345	0.6761	1	0.6761	0.8345	0.6761

Table 7.5 *p-value of Wilcoxon Test (p-VAL) – (Median Kubernetes/OpenShift, Median Docker Swarm) – (Cliff δ Effect Size (ES))*

HONO-DS-AWS-Serverless simulating workload app DS-1									
# Devices	CPU cores usage (p-value)			Memory usage (p-value)			Network consumption (p-value)		
	min	max	avg	min	max	avg	min	max	avg
10	1	1	1	0.8413	0.8413	0.8413	0.2222	0.1508	0.2222
20	0.6905	0.5476	0.4206	0.8413	0.6905	0.8413	**0.01587** ***(16.9, 47.48)*** ***(ES=0.36)***	**0.03175** ***(35.83, 62.02)*** ***(ES=0.36)***	**0.03175** ***(20.75, 53.19)*** ***(ES=0.36)***
30	0.8413	0.6905	0.6905	0.05556 *(76.22, 22.01)* *(ES=0.36)*	0.1508	0.05556 *(88.83, 26.76)* *(ES=0.36)*	0.8413	0.8413	0.8413
40	**0.007937** ***(64.244, 44.92)*** ***(ES=0.68)***	0.09524	**0.01587** ***(73.242, 57.67)*** ***(ES=0.68)***	1	0.6905	0.6905	0.1508	0.6905	0.6905
50	0.6905	**0.007937** ***(92.953, 68.76)*** ***(ES=0.68)***	**0.007937** ***(83.413, 63.33)*** ***(ES=0.68)***	1	1	0.8413	0.05556 *(73.71, 24.14)* *(ES=0.36)*	0.05556 *(87.64, 36.31)* *(ES=0.36)*	0.09524
60	0.6905	0.6905	0.6905	0.6905	0.8413	0.6905	0.3095	0.4206	0.4206
70	0.1508	0.1508	0.1508	0.05556 *(72.62, 41.93)* *(ES=0.36)*	**0.03175** ***(96.27, 56.95)*** ***(ES=0.36)***	**0.03175** ***(84.59, 47.45)*** ***(ES=0.36)***	0.8413	0.4206	0.8413
80	0.3095	0.3095	0.3095	0.3095	0.3095	0.3095	0.2222	0.3095	0.2222
90	0.6905	0.1508	0.6905	0.6905	1	1	0.8413	1	1
100	**0.007937** ***(64.775, 90.19)*** ***(ES=0.68)***	**0.007937** ***(108.31, 82.725)*** ***(ES=0.68)***	**0.007937** ***(74.732, 100.92)*** ***(ES=0.68)***	0.8413	0.8413	0.8413	0.2222	0.5476	0.3095

realizes that it is under resource pressure, it then stops accepting new pods formulations requests to try complete what is already there, in terms of processes.

In the case of CPU cores usage and Network consumption, we did not notice any significant difference between Kubernetes and OpenShift. Hence, in these cases, we cannot reject HR_x^{1-1} and HR_x^{1-3} for the CPU cores usage and network consumption for such environments. A similar behaviour was also observed in our previous study, chapter 6, for OpenShift; it also consumed more **memory** than Kubernetes, when regular applications were deployed on top of the scaled Hono.

In addition to the previous observation, the trend of the Memory usage, in both container setups, are slightly larger than that of the Docker Swarm environment. The trends tend to fall down, yet maintaining a statistically significant difference with the distribution of memory consumed by the Docker Swarm environment. Both Kubernetes and OpenShift need larger amounts of memory to allow the proper deployments of the serverless application.

Findings: Performance metrics typically do not follow the same distribution in container environments when deploying serverless applications on top of the framework and after scaling it out using the Eclipse Hono messaging infrastructure.
Actionable implications: Developers should be aware of the performance implications of choosing Kubernetes/OpenShift containers to deploy their IoT serverless applications. They should ensure that enough memory is available for proper deployments. Moreover, they should consider deploying their IoT serverless applications on top of Docker Swarm, for better performance.

- **RQ8.3.: To what extent does the relationship between the performance metrics change across Kubernetes, OpenShift, and Docker Swarm setups?**

Motivation The relationship between performance metrics may significantly change and be different between environments, which may be a glimpse of system regression or performance issues. According to [412], combinations of performance metrics are more predictive of performance issues than a single metric. A change in correlation values between a group of performance metrics often reveals performance issues in an application and–or discrepancies between its performance across different platforms. Practitioners often rely on correlation analysis of performance indicators to identify the behavioral changes of a system between different environments. For instance, in one system, the CPU may be correlated to a great extent with network usage (e.g., when network's operations are high due to the workload being generated, eventually CPU is high to accommodate such increase); on the other hand,

in the same system, the correlation between CPU and memory may become low. Such change identified may expose performance issues (i.e., high CPU without memory and–or network I/O operations might be due to a performance failure).

Also, if there is a significant difference in the correlations simply because of the platform being used, i.e., Kubernetes vs. OpenShift vs. Docker Swarm, then practitioners may need to be warned about the potential implications of selecting one platform over another.

In this research question, we examine whether the relationship between performance metrics varies across our three studied environments, i.e., Kubernetes, OpenShift, and Docker Swarm.

Approach We calculated the Spearman's rank correlation coefficient between all performance metrics, i.e., in the container setups and Docker Swarm stack, and studied whether they are different. For instance, in one environment, the CPU cores usage may be highly correlated with the network I/O; whereas in another environment, such correlation could be low. In such a case, we consider that there is a discrepancy in the correlation coefficient between the CPU and the network I/O.

Spearman's Correlation is an effect size (ES) measure and the strength of the correlation for the r_s is described as: 0.00-0.19 "very weak", 0.20-0.39 "weak", 0.40-0.59 "moderate", 0.60-0.79 "strong", and 0.80-1.0 "very strong".

We chose the Spearman's rank correlation coefficient because it does not require any assumptions about the distribution of the variables. This is necessary because load test data contains traces that do not follow a normal distribution.

Results There exists differences in correlation between the performance metrics in each of Kubernetes, OpenShift and Docker Swarm. Tables 7.6, 7.7, and 7.8 show the changes in the correlation coefficient among the resources utilized in each environment. By closely looking at them, we find that (CPU–Network) in Kubernetes and Docker Swarm have a stronger correlation than that of OpenShift (i.e., noticeable network's operations due to Hono's deployment workload); whereas (Memory–Network) coefficients in OpenShift are stronger than the other two environments[3].

When the serverless application was added on top of Kubernetes, the (Memory–Network)

[3]A detailed statistical analysis for Hono's bare deployment and other serverful applications on top of both Kubernetes and OpenShift has been carefully studied in the previous chapter 6.

Table 7.6 *Spearman's rank correlation summary of performance metrics in Docker Swarm on top of Amazon (AWS)*

# Devices	**CPU–Memory**			**CPU–Network**			**Memory–Network**		
	min	max	avg	min	max	avg	min	max	avg
HONO AWS-Swarm Deployment DS-0									
10 –> 100	0.1	0.5	0.2	0.1	**0.6 (strong)**	-0.3	0.2	0.3	0.3
HONO-DS-serverless app DS-1									
10 –> 100	**0.9 (v. strong)**	-0.4	0.4	-0.6	**0.6 (strong)**	-0.9	-0.7	-0.2	-0.7

Table 7.7 *Spearman's rank correlation summary of serverless performance metrics in Kubernetes deployments*

# Devices	**CPU–Memory**			**CPU–Network**			**Memory–Network**		
	min	max	avg	min	max	avg	min	max	avg
HONO Bare metal Deployment									
10 –> 100	-0.1025978	-0.1	-0.3077935	**0.6 (strong)**	**0.7 (strong)**	**0.6 (strong)**	0.2051957	0.1	0.2051957
HONO-Serverless-App KO									
10 –> 100	-0.9	-0.7	-0.9	-0.6	-0.4	-0.6	**0.8 (v. strong)**	**0.9 (v. strong)**	**0.8 (v. strong)**

coefficient has showed a very strong bond rather than the other two setups. Whereas, while deploying the serverless application on top of the Docker Swarm platform on AWS, both the (CPU–Memory) and (CPU-Network) have shown very strong and strong relationships, respectively. This is becuase the workload simulator serverless app is used to determine system performance and response time to evaluate network design, and to simulate the actions of a number of different events. OpenShift was neutral and has not shown any bond between it's metrics. For the sake of brevity, we do not show the detailed analysis here, but it is available in our replication package indicated in the next section, i.e., Section 7.6. Box-plots, as well as Spearman's correlation trend-lines, are depicted therein to show the correlation changes among related metrics.

Findings: The correlations between performance metrics may change considerably between container environments when serverless apps are deployed on top of them using Eclipse Hono.
Actionable implications: Practitioners should pay attention to these performance deviations across the three studied platforms when planning the migration of their IoT apps. They should also avoid reusing blindly any performance benchmarking result obtained in a different environment.

7.6 Threats to Validity

This section discusses the threats to the validity of our two conducted studies, in chapters 6 and 7, taking into account the guidelines suggested by Wohlin et al. [413].

Construct validity threats concern the relation between both theory and observations, such as the measurements errors. We instrumented the different versions of deployments, described in Section 7.4, to generate execution readings from which we computed min, max and average values of the performance metrics. We repeated every single experiment five times and computed the median values to mitigate the potential biases that could be induced by perturbations on the network, hardware and our tracing. We are confident that such repeated measurements increased the quality of our calibration and measurements. We monitored the performance of the containers since their creation until their destruction and combined the performance metrics for every minute together as a median value.

Internal validity threats concern our analysis method. Our empirical study is based on the performance testing results obtained from the studied subject systems. The way of conducting the performance tests and its quality may introduce threats to the validity of our findings. In particular, our approach is based on the recorded performance metrics, and their quality can have an impact on the internal validity of our conducted study. Our performance evaluations lasted for a duration of, roughly, six $\rightarrow$ eight months, while the length of the evaluations may impact the findings of the conducted case study, we believe that we have observed and analyzed performance readings over a realistic amount of time. The statistical analysis that is carried out is considered to be another internal validity threat. To mitigate it, we paid attention not to violate the assumptions of the statistical tests. Specifically, we applied non-parametric tests that do not require making assumptions on the distribution of our dataset.

External validity threats concern the possibility of generalizing our findings. Further validation with different configurations is desirable to broaden our understanding of the impact of Eclipse Hono deployment strategies on the resources utilization and to provide guidelines to practitioners about the usage of such platform when developing and deploying IoT serverless applications.

Reliability validity threats concern the possibility to replicate this study. We attempt to provide all the necessary information and details to replicate our studies. All the data used in the two conducted studies, in chapters 6 and 7, are available online in our replication package[4].

[4]Complete results, box plots, data and scripts are shared online on *osf.io* scientific data repository: `http:`

Table 7.8 *Spearman's rank correlation summary of serverless performance metrics in OpenShift deployments*

# Devices	**CPU–Memory**			**CPU–Network**			**Memory–Network**		
	min	max	avg	min	max	avg	min	max	avg
HONO Bare metal Deployment									
10 –> 100	**0.6 (strong)**	0.1	0.3	-0.1	0.1	0.1	-0.3	**0.8 (v. strong)**	**0.6 (strong)**
HONO-Serverless-App KO									
10 –> 100	0.3	0.3	0.3	-0.9	-0.7	-0.6	-0.4	-0.3	-0.1

Last but not least, the **conclusion validity** threats which refer to the relation between the treatment and the outcome do not affect this study since we paid attention to avoid violating the assumptions of the statistical tests used in our analysis. Nevertheless, replications of this study on more complex clusters using different applications are desirable to make our findings more generic.

7.7 Chapter Summary

Performance assurance activities are crucial in ensuring platform–software reliability. Virtual environments nowadays are used to conduct performance tests. However, the discrepancy between testing results between different virtual environments, including container setups, are still under evaluated. We aimed at investigating whether a discrepancy present between different container environments will impact the studies and tests carried out in the IoT domain. In this chapter, we tested and evaluated such discrepancy by conducting performance tests on the open-sourced IoT platform, Eclipse IoT, in different container/stack environments. By examining the performance testing results, we find that there exists discrepancies between performance testing results in each of the environment involved when examining single performance metric, the relationship, and building statistical performance models among different performance metrics.

`//bit.ly/benchmarking-hono-FaaS-deployments`

The main contributions of this chapter include:

- This empirical study is one of the first research works that attempts to evaluate the discrepancy in the context of analyzing the performance testing results in container environments.
- We find that relationships among related performance metrics have large differences between different setups. Practitioners, including developers, cannot always assume a straightforward overhead from deploying an open-sourced IoT platform, specifically IoT–Hono, with serverless applications on top of different container/Cloud technologies.
- We evaluated serverless computing environment invoking functions in parallel to demonstrate the performance and throughput of serverless computing for open-sourced framework. We compared their performance regarding CPU, memory and network I/O between a sequential and a concurrence invocations that helps understanding performance and function behaviors on serverless computing environment.
- We distributed the deployment of the Hono platform on top of AWS Cloud infrastructure and assessed such performance implications while adding an additional layer of a serverless app on top of it.

Our results stress the need to be aware of and to reduce the discrepancy between performance testing results in container environments, for both practitioners and researchers.

CHAPTER 8 CONCLUSION AND RECOMMENDATIONS

8.1 Summary

Newly discovered pillars within the Internet of Things (IoT) paradigm, such as smart cities, Industry 4.0, Industrial IoT (IIoT), etc., require specific platform(s) to allow different components to integrate and communicate. The value of IoT systems often correlates directly with the ability of those platforms to connect different devices efficiently and integrate them into higher-level solutions. The paradigm aims at bringing connectivity to almost everyday objects found in the physical and ambient spaces. Industry nowadays incorporates different technologies and mechanisms to help organizations improve their efficiency and reduce costs whenever possible. Nonetheless, architecting, designing, and implementing an IoT network can still be a challenging task nowadays.

Since its inception, IoT services and applications have been developed to be adapted in a vertical service model, i.e., in which every layer has been designed by a company or organization. Interoperability and fragmentation among various service domains are a major challenge. As a consequence, the latest trends of IoT technologies are to adapt horizontal service domain to achieve interoperability between participating things. Various standards and protocols have been proposed by a number of IoT consortia to tackle the integration issues. However, current IoT technologies do not fully make an inter-operation between various devices in different networks. Thus, we investigated the integration, interoperability and fragmentation issues in IoT and proposed viable solutions as countermeasures.

In addition, the emergence of the paradigm itself has led and will continue to lead, to a number of threats and possible breaches against its security. Unfortunately, security risks are not well-recognized in this domain until now. Therefore, we studied the specific properties of the IoT, devised a detailed security and privacy requirements list, summarized several IoT attacks and their countermeasures at each level to establish a secured IoT framework. Given the wide applicability of the IoT, security threats should be continuously addressed by the academic research communities, industry, as well as manufacturers in a very proactive way.

Last but not least, performance assurance activities are crucial in ensuring platform–software reliability. Virtual environments today are used to conduct performance tests. However, the discrepancy between testing results between different virtual environments, including container setups, are still under evaluated. We aimed at investigating whether a discrepancy exist between different container environments will impact the studies and tests carried out

in the IoT domain – specifically when a number of applications, varied between serverful and serverless, have been deployed on top of the architecture. We tested and evaluated such discrepancy by conducting performance tests on the open-sourced IoT platform, Eclipse IoT, in different container/stack environments. By examining the performance testing results, we find that there exists discrepancies between performance testing results in the different setups involved when examining single performance metric, the relationship, and building statistical performance models among different performance metrics. Our results emphasize the need to be aware of and to reduce the discrepancy between performance testing results in container environments, for both practitioners and researchers.

8.2 Limitations of this Thesis

Beyond the empirical aspects on the various IoT directions, such as the identification of the challenges faced by IoT developers, interoperability, security and application deployment issues in the industrial IoT frameworks, in this thesis:

- A part of our discussions identification on online Stack Exchange forums relies on manual labelling for the different topics extracted. Future automated identification using larger discussion sizes along with more volunteering coders would be needed; not only for more consistent identified topic and accuracy, but also to make real consensus more deterministic.
- We only used container technologies as subjective systems for the open-source IoT platform we dealt with. Although the aforementioned utilized containers are large-scale systems that allow practitioners to access its docker images repositories to build environments, we cannot guarantee that our findings are generalizable to other types of containerized DevOps tools, e.g., Jaeger Tracing and Azure Kubernetes Service (AKS).
- We only analyzed the performance of three utilized resources at the container levels (i.e., CPU cores usage, memory usage, and network I/O throughput) for the Telemetry messages sent from the participating devices. This is a concern for system optimization practitioners and research into the impact of the whole performance of the ecosystem. Hence, additional consideration for the "Command&Control" messages (i.e., full interaction between the Hono Sandbox and devices) should be considered alongside with various utilized resources should be taken into consideration so that the comprehension of the whole performance is more generalized.

8.3 Future Work and Research

We plan to study IoT/Industry 4.0-related questions by filtering more information of their corresponding answers on Stack Exchange communities. In addition, we will consider and investigate IoT/Industry 4.0-related posts on other software Q&A websites. Moreover, we will further develop our conducted studies and–or reviews in two directions: (1) including more papers to cover more IoT devices, protocols, and standards, in particular networking protocols. (2) defining and assessing various aspects of IoT devices, in particular security, energy consumption, and usability in addition to interoperability. This can also include developing means for users and developers of IoT devices to assess possible integration issues before using/releasing the devices into action.

Also, as an extension to such a study, we intend to investigate practitioners' perspective of IoT applications in industry 4.0 by exploring website data. This would enable us to determine how compatible academic research is with state-of-the-art in industry. We believe this is a small but important step towards understanding what research has to be undertaken to transform the delivery and quality of industry 4.0 through IoTs.

Conventionally, this dissertation is the first step to lay a concrete ground for a deeper understanding of the discrepancy between performance test results in different container setups while adopting open-sourced IoT platforms, deploying serverful and serverless apps on top of them, and the impact of detecting performance issues with such kind of discrepancies. Having such knowledge of the discrepancies, we can better comprehend the existence and magnitude of impact on detecting real-world performance degradation in the future. But still, we need to focus our research efforts on generating comparable performance testing results from different setups with different workloads as well as to discuss when to go for either deployments, i.e., serverful and serverless, in the context of specific cases of IoT application deployments. Such guidelines are utterly needed in the industry nowadays.

Beyond the above-mentioned deployment considerations, we further plan to investigate the uni-kernel application deployments and associate and–or compare it with the empirical evaluations conducted earlier to get a complete picture the possible paradigms that could be considered for IoT applications designs and deployments. It is an another interesting issue which has not been discussed in the scope of this thesis and it would be beneficial to be assessed to provide practitioners with all possible deployment environments to choose the best fit for them.

REFERENCES

[1] O. Vermesan and P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[2] O. Vermesan, P. Friess *et al.*, *Internet of things-from research and innovation to market deployment*. River publishers Aalborg, 2014, vol. 29.

[3] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things: Taxonomies and open challenges," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 796–809, 2019.

[4] R. H. Weber, "Internet of things–new security and privacy challenges," *Computer law & security review*, vol. 26, no. 1, pp. 23–30, 2010.

[5] P. Patel and D. Cassou, "Enabling high-level application development for the internet of things," *Journal of Systems and Software*, vol. 103, pp. 62–84, 2015.

[6] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[8] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 53–56.

[9] S. Azad, P. C. Rigby, and L. Guerrouj, "Generating api call rules from version history and stack overflow posts," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 4, p. 29, 2017.

[10] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[11] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[12] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?" in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 804–807.

[13] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 112–121.

[14] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[15] J. Manyika, *The Internet of Things: Mapping the value beyond the hype*. McKinsey Global Institute, 2015.

[16] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[17] S. Agrawal and M. L. Das, "Internet of things—a paradigm shift of future internet applications," in *2011 Nirma University International Conference on Engineering*. IEEE, 2011, pp. 1–7.

[18] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[19] A. Giaretta, S. Balasubramaniam, and M. Conti, "Security vulnerabilities and countermeasures for target localization in bio-nanothings communication networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 665–676, 2015.

[20] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[21] M. Hepp, K. Siorpaes, and D. Bachlechner, "Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management," *IEEE Internet Computing*, vol. 11, no. 5, 2007.

[22] G. P. Joshi and S. W. Kim, "Survey, nomenclature and comparison of reader anticollision protocols in rfid," *IETE Technical Review*, vol. 25, no. 5, pp. 234–243, 2008.

[23] K. Pretz, "The next evolution of the internet," *IEEE Magazine The institute*, vol. 50, no. 5, 2013.

[24] D. Kiritsis, "Closed-loop plm for intelligent products in the era of the internet of things," *Computer-Aided Design*, vol. 43, no. 5, pp. 479–501, 2011.

[25] L. Li, "Effects of enterprise technology on supply chain collaboration: analysis of china-linked supply chain," *Enterprise Information Systems*, vol. 6, no. 1, pp. 55–77, 2012.

[26] L. Li and J. Liu, "An efficient and flexible web services-based multidisciplinary design optimisation framework for complex engineering systems," *Enterprise Information Systems*, vol. 6, no. 3, pp. 345–371, 2012.

[27] M. Köhler, D. Wörner, and F. Wortmann, "Platforms for the internet of things–an analysis of existing solutions," in *5th Bosch Conference on Systems and Software Engineering (BoCSE)*, 2014.

[28] ETSI, "European telecommunications standards institute," Available at http://www.etsi.org/, accessed (2018/01/08), 2013.

[29] N. Nic, "Disruptive civil technologies: Six technologies with potential impacts on us interests out to 2025," *Tech. Rep.*, 2008.

[30] L. Jiang, L. Da Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An iot-oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[31] M. Y. Kataev, L. Bulysheva, A. Emelyanenko, and V. Emelyanenko, "Enterprise systems in russia: 1992—2012//enterprise information systems," *Volume*, vol. 7, pp. 169–186, 2012.

[32] Q. Li, Z.-y. Wang, W.-h. Li, J. Li, C. Wang, and R.-y. Du, "Applications integration in a hybrid cloud computing environment: Modelling and platform," *Enterprise Information Systems*, vol. 7, no. 3, pp. 237–271, 2013.

[33] L. Ren, L. Zhang, F. Tao, X. Zhang, Y. Luo, and Y. Zhang, "A methodology towards virtualisation-based high performance simulation platform supporting multidisciplinary design of complex products," *Enterprise Information Systems*, vol. 6, no. 3, pp. 267–290, 2012.

[34] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, and B. H. Li, "Cciot-cmfg: cloud computing and internet of things-based cloud manufacturing service system," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1435–1442, 2014.

[35] C. Wang, Z. Bi, and L. Da Xu, "Iot and cloud computing in automation of assembly modeling systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1426–1434, 2014.

[36] R. H. Deng, Y. Li, M. Yung, and Y. Zhao, "A new framework for rfid privacy." in *ESORICS*, vol. 6345. Springer, 2010, pp. 1–18.

[37] R. Van Kranenburg, E. Anzelmo, A. Bassi, D. Caprio, S. Dodson, and M. Ratto, "The internet of things," in *1st Berlin Symposium on Internet and society*, 2011, pp. 25–27.

[38] S. Li, L. Xu, X. Wang, and J. Wang, "Integration of hybrid wireless networks in cloud services oriented enterprise information systems," *Enterprise Information Systems*, vol. 6, no. 2, pp. 165–187, 2012.

[39] A. Malatras, A. Asgari, and T. Baugé, "Web enabled wireless sensor networks for facilities management," *IEEE systems journal*, vol. 2, no. 4, pp. 500–512, 2008.

[40] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[41] C. Pautasso and E. Wilde, "Why is the web loosely coupled?: a multi-faceted metric for service design," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 911–920.

[42] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "M^ 2ap: A minimalist mutual-authentication protocol for low-cost rfid tags," *UIC*, vol. 6, pp. 912–923, 2006.

[43] C. Wang, "Advances in information integration infrastructures supporting multidisciplinary design optimisation," 2012.

[44] R. Kolcun, D. Boyle, and J. A. McCann, "Efficient in-network processing for a hardware-heterogeneous iot," in *Proceedings of the 6th International Conference on the Internet of Things*. ACM, 2016, pp. 93–101.

[45] R. F. Brena and E. Garcia-Ceja, "A crowdsourcing approach for personalization in human activities recognition," *Intelligent Data Analysis*, vol. 21, no. 3, pp. 721–738, 2017.

[46] A. M. Nia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[47] M. Weyrich and C. Ebert, "Reference architectures for the internet of things," *IEEE Software*, vol. 33, no. 1, pp. 112–116, 2016.

[48] H. Panetto and J. Cecil, "Information systems for enterprise integration, interoperability and networking: theory and applications," 2013.

[49] R. Jardim-Goncalves, A. Grilo, C. Agostinho, F. Lampathaki, and Y. Charalabidis, "Systematisation of interoperability body of knowledge: the foundation for enterprise interoperability as a science," *Enterprise Information Systems*, vol. 7, no. 1, pp. 7–32, 2013.

[50] X. V. Wang and X. W. Xu, "Dimp: an interoperable solution for software integration and product data exchange," *Enterprise Information Systems*, vol. 6, no. 3, pp. 291–314, 2012.

[51] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: the internet of things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on*. IEEE, 2012, pp. 257–260.

[52] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang, and W. Liu, "Study and application on the architecture and key technologies for iot," in *Multimedia Technology (ICMT), 2011 International Conference on*. IEEE, 2011, pp. 747–751.

[53] M. A. Chaqfeh and N. Mohamed, "Challenges in middleware solutions for the internet of things," in *Collaboration Technologies and Systems (CTS), 2012 International Conference on*. IEEE, 2012, pp. 21–26.

[54] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[55] D. Minoli, K. Sohraby, and B. Occhiogrosso, "Iot security (iotsec) mechanisms for e-health and ambient assisted living applications," in *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017 IEEE/ACM International Conference on*. IEEE, 2017, pp. 13–18.

[56] D. M. Blei and J. Lafferty, "Topic models. text mining: theory and applications," 2009.

[57] T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum, "Topics in semantic representation." *Psychological review*, vol. 114, no. 2, p. 211, 2007.

[58] D. Newman, C. Chemudugunta, P. Smyth, and M. Steyvers, "Analyzing entities and topics in news articles using statistical topic models," in *International conference on intelligence and security informatics*. Springer, 2006, pp. 93–104.

[59] A. Kuhn, S. Ducasse, and T. Gírba, "Semantic clustering: Identifying topics in source code," *Information and Software Technology*, vol. 49, no. 3, pp. 230–243, 2007.

[60] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Source code retrieval for bug localization using latent dirichlet allocation," in *2008 15th Working Conference on Reverse Engineering*. IEEE, 2008, pp. 155–164.

[61] G. Mathew, A. Agrawal, and T. Menzies, "Finding trends in software research," *IEEE Transactions on Software Engineering*, 2018.

[62] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, "Modeling the evolution of topics in source code histories," in *Proceedings of the 8th working conference on mining software repositories*. ACM, 2011, pp. 173–182.

[63] G. Heinrich, "Parameter estimation for text analysis," Technical report, Tech. Rep., 2005.

[64] E. Hono, "Eclipse hono," Available at https://www.eclipse.org/hono/, accessed (2018/02/01), 2017.

[65] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," in *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.

[66] N. Kratzke, "A brief history of cloud application architectures," *Applied Sciences*, vol. 8, no. 8, p. 1368, 2018.

[67] M. Roberts and J. Chapin, *What is Serverless?* O'Reilly Media, Incorporated, 2017.

[68] H. Li, Z. Xing, X. Peng, and W. Zhao, "What help do developers seek, when and how?" in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 142–151.

[69] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming q&a in stackoverflow," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 25–34.

[70] W. Wang and M. W. Godfrey, "Detecting api usage obstacles: A study of ios and android developer questions," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 61–64.

[71] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of stack overflow," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 97–100.

[72] S. Beyer and M. Pinzger, "A manual categorization of android app development issues on stack overflow," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 531–535.

[73] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 93–96.

[74] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, "Jumping through hoops: Why do java developers struggle with cryptography apis?" in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 935–946.

[75] X. Xia, D. Lo, X. Wang, and B. Zhou, "Tag recommendation in software information sites," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 287–296.

[76] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik, "Entagrec++: An enhanced tag recommendation system for software information sites," *Empirical Software Engineering*, vol. 23, no. 2, pp. 800–832, 2018.

[77] L. Nie, H. Jiang, Z. Ren, Z. Sun, and X. Li, "Query expansion based on crowd knowledge for code search," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 771–783, 2016.

[78] H. Jiang, J. Zhang, X. Li, Z. Ren, and D. Lo, "A more accurate model for finding tutorial segments explaining apis," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 157–167.

[79] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, "Multi-factor duplicate question detection in stack overflow," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 981–997, 2015.

[80] X. Xia, D. Lo, D. Correa, A. Sureka, and E. Shihab, "It takes two to tango: Deleted stack overflow question prediction with text and meta features," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2016, pp. 73–82.

[81] X.-Y. Wang, X. Xia, and D. Lo, "Tagcombine: Recommending tags to contents in software information sites," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 1017–1035, 2015.

[82] B. Xu, Z. Xing, X. Xia, D. Lo, and S. Li, "Domain-specific cross-language relevant question retrieval," *Empirical Software Engineering*, vol. 23, no. 2, pp. 1084–1122, 2018.

[83] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016, pp. 51–62.

[84] S. W. Thomas, A. E. Hassan, and D. Blostein, "Mining unstructured software repositories," in *Evolving Software Systems*. Springer, 2014, pp. 139–162.

[85] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in *2009 IEEE International Conference on Software Maintenance*. IEEE, 2009, pp. 339–348.

[86] S. Neuhaus and T. Zimmermann, "Security trend analysis with cve topic models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 2010, pp. 111–120.

[87] J. Garcia, D. Popescu, C. Mattmann, N. Medvidovic, and Y. Cai, "Enhancing architectural recovery using concerns," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 552–555.

[88] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 522–531.

[89] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen, "A topic-based approach for narrowing the search space of buggy files from a bug report,"

in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2011, pp. 263–272.

[90] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2012, pp. 70–79.

[91] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Bug localization using latent dirichlet allocation," *Information and Software Technology*, vol. 52, no. 9, pp. 972–990, 2010.

[92] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *IEEE Wireless Communications*, vol. 20, no. 6, pp. 91–98, 2013.

[93] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.

[94] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the internet of things: a survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[95] J. Granjal, R. Silva, E. Monteiro, J. S. Silva, and F. Boavida, "Why is ipsec a viable option for wireless sensor networks," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*. IEEE, 2008, pp. 802–807.

[96] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147–159, 2011.

[97] S. Cirani, G. Ferrari, and L. Veltri, "Enforcing security mechanisms in the ip-based internet of things: An algorithmic overview," *Algorithms*, vol. 6, no. 2, pp. 197–226, 2013.

[98] R. Mitchell and R. Chen, "A survey of intrusion detection in wireless network applications," *Computer Communications*, vol. 42, pp. 1–23, 2014.

[99] A. Abduvaliyev, A.-S. K. Pathan, J. Zhou, R. Roman, and W.-C. Wong, "On the vital areas of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1223–1237, 2013.

[100] Y. Wang, T. Uehara, and R. Sasaki, "Fog computing: Issues and challenges in security and forensics," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 3. IEEE, 2015, pp. 53–59.

[101] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.

[102] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[103] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[104] V. Oleshchuk, "Internet of things and privacy preserving technologies," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*. IEEE, 2009, pp. 336–340.

[105] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based iot: challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.

[106] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 230–234.

[107] D. Miessler, "Securing the internet of things: Mapping attack surface areas using the owasp iot top 10," in *RSA Conference*, 2015.

[108] D. Kozlov, J. Veijalainen, and Y. Ali, "Security and privacy threats in iot architectures," in *Proceedings of the 7th International Conference on Body Area Networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 256–262.

[109] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the internet of things (iot)," *Recent Trends in Network Security and Applications*, pp. 420–429, 2010.

[110] A. Dujin, C. Geissler, and D. Horstkötter, "Think act industry 4.0. the new industrial revolution: How europe will succeed," *Ronald Berger Strategy Consultants GmbH: Munich, 24p*, 2014.

[111] M. Mittermair, "Industry 4.0 initiatives," *SMT: Surface Mount Technology*, vol. 30, no. 3, pp. 58–63, 2015.

[112] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios: a literature review," *Technische Universität Dortmund, Dortmund*, 2015.

[113] ——, "Design principles for industrie 4.0 scenarios," in *2016 49th Hawaii international conference on system sciences (HICSS)*. IEEE, 2016, pp. 3928–3937.

[114] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, "How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective," *International journal of mechanical, industrial science and engineering*, vol. 8, no. 1, pp. 37–44, 2014.

[115] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[116] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pp. 18–23, 2015.

[117] S. Parvin, F. K. Hussain, O. K. Hussain, T. Thein, and J. S. Park, "Multi-cyber framework for availability enhancement of cyber physical systems," *Computing*, vol. 95, no. 10-11, pp. 927–948, 2013.

[118] P. Andersson and L.-G. Mattsson, "Service innovations enabled by the "internet of things"," *Imp Journal*, 2015.

[119] W. Wahlster, H.-J. Grallert, S. Wess, H. Friedrich, and T. Widenka, *Towards the internet of services: The THESEUS research program*. Springer, 2014.

[120] A. Barros and D. Oberle, "Handbook of service description," *USDL and Its Methods*, 2012.

[121] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.

[122] D. Spath, O. Ganschar, S. Gerlach, M. Hämmerle, T. Krause, and S. Schlund, *Produktionsarbeit der Zukunft-Industrie 4.0*. Fraunhofer Verlag Stuttgart, 2013, vol. 150.

[123] L. Da Xu, "Enterprise systems: state-of-the-art and future trends," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 630–640, 2011.

[124] C. Flügel and V. Gehrmann, "Scientific workshop 4: Intelligent objects for the internet of things: Internet of things–application of sensor networks in logistics," in *European Conference on Ambient Intelligence*. Springer, 2008, pp. 16–26.

[125] Z. Pang, Q. Chen, J. Tian, L. Zheng, and E. Dubrova, "Ecosystem analysis in the design of open platform-based in-home healthcare terminals towards the internet-of-things," in *2013 15th international conference on advanced communications technology (ICACT)*. IEEE, 2013, pp. 529–534.

[126] M. C. Domingo, "An overview of the internet of things for people with disabilities," *journal of Network and Computer Applications*, vol. 35, no. 2, pp. 584–596, 2012.

[127] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[128] Z. Ji and Q. Anwen, "The application of internet of things (iot) in emergency management system in china," in *2010 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE, 2010, pp. 139–142.

[129] W. Qiuping, Z. Shunbing, and D. Chunquan, "Study on key technologies of internet of things perceiving mine," *Procedia Engineering*, vol. 26, pp. 2326–2333, 2011.

[130] M. Woodside, G. Franks, and D. C. Petriu, "The future of software performance engineering," in *2007 Future of Software Engineering*. IEEE Computer Society, 2007, pp. 171–187.

[131] M. D. Syer, W. Shang, Z. M. Jiang, and A. E. Hassan, "Continuous validation of performance test workloads," *Automated Software Engineering*, vol. 24, no. 1, pp. 189–231, 2017.

[132] M. D. Syer, Z. M. Jiang, M. Nagappan, A. E. Hassan, M. Nasser, and P. Flora, "Leveraging performance counters and execution logs to diagnose memory-related performance issues," in *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*. IEEE, 2013, pp. 110–119.

[133] H. Malik, B. Adams, and A. E. Hassan, "Pinpointing the subsystems responsible for the performance deviations in a load test," in *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*. IEEE, 2010, pp. 201–210.

[134] H. Malik, Z. M. Jiang, B. Adams, A. E. Hassan, P. Flora, and G. Hamann, "Automatic comparison of load tests to support the performance analysis of large enterprise systems," in *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*. IEEE, 2010, pp. 222–231.

[135] H. Malik, H. Hemmati, and A. E. Hassan, "Automatic detection of performance deviations in the load testing of large scale systems," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 1012–1021.

[136] T. H. Nguyen, B. Adams, Z. M. Jiang, A. E. Hassan, M. Nasser, and P. Flora, "Automated detection of performance regressions using statistical process control techniques," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. ACM, 2012, pp. 299–310.

[137] W. A. Shewhart, *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[138] C. Heger, J. Happe, and R. Farahbod, "Automated root cause isolation of performance regressions during software development," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ACM, 2013, pp. 27–38.

[139] M. Jiang, M. A. Munawar, T. Reidemeister, and P. A. Ward, "System monitoring with metric-correlation models: problems and solutions," in *Proceedings of the 6th international conference on Autonomic computing*. ACM, 2009, pp. 13–22.

[140] K. C. Foo, Z. M. Jiang, B. Adams, A. E. Hassan, Y. Zou, and P. Flora, "Mining performance regression testing repositories for automated performance analysis," in *Quality Software (QSIC), 2010 10th International Conference on*. IEEE, 2010, pp. 32–41.

[141] M. Jiang, M. A. Munawar, T. Reidemeister, and P. A. Ward, "Automatic fault detection and diagnosis in complex software systems by information-theoretic monitoring," in *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. IEEE, 2009, pp. 285–294.

[142] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, and P. Kilpatrick, "Io performance prediction in consolidated virtualized environments," in *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5. ACM, 2011, pp. 295–306.

[143] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments." in *CLOSER*, 2011, pp. 563–573.

[144] Y. Cui, "How does language, memory and package size affect cold starts of aws lambda?" Available at https://read.acloud.guru/does-coding-language-memory-or-package-size-affect-cold-starts-of-aws-lambda-a15e26d12c76, accessed (2019/03/26), 2017.

[145] C. Smith, "Understanding aws lambda performance—how much do cold starts really matter?" Available at https://blog.newrelic.com/technology/aws-lambda-cold-start-optimization/, accessed (2019/03/26), 2017.

[146] E. Windisch, "Understanding aws lambda coldstarts," Available at https://read.iopipe.com/understanding-aws-lambda-coldstarts-49350662ab9e, accessed (2019/03/26), 2017.

[147] Y. Cui, "How long does aws lambda keep your idle functions around before a cold start?" Available at https://read.acloud.guru/how-long-does-aws-lambda-keep-your-idle-functions-around-before-a-cold-start-bf715d3b810, accessed (2019/03/26), 2017.

[148] M. Akin, "How does proportional cpu allocation work with aws lambda?" Available at https://engineering.opsgenie.com/how-does-proportional-cpu-allocation-work-with-aws-lambda-41cd44da3cac, accessed (2019/03/26), 2018.

[149] J. Chapin, "The occasional chaos of aws lambda runtime performance," Available at https://blog.symphonia.io/the-occasional-chaos-of-aws-lambda-runtime-performance-880773620a7e, accessed (2019/03/26), 2017.

[150] F. Willaert, "Aws lambda container lifetime and config refresh," Available at https://www.linkedin.com/pulse/aws-lambda-container-lifetime-config-refresh-frederik-willaert/, accessed (2019/03/26), 2016.

[151] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in *8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[152] G. McGrath and P. R. Brenner, "Serverless computing: Design, implementation, and performance," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2017, pp. 405–410.

[153] W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, and S. Pallickara, "Serverless computing: An investigation of factors influencing microservice performance," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 159–169.

[154] P. M. F, "Snowball: A language for stemming algorithms." Available at http://snowball.tartarus.org/texts/introduction.html, accessed (2019/10/12), 2016.

[155] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "Api change and fault proneness: a threat to the success of android apps," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*. ACM, 2013, pp. 477–487.

[156] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2011, pp. 2857–2866.

[157] S. R. Services, "Schlumberger limited 2018 annual report," Available Online: https://investorcenter.slb.com/static-files/26a0c7a1-58ce-4903-b10f-41af7ea18169, accessed (2019/11/13), 2018.

[158] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[159] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[160] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu, "Snow: Sensor network over white spaces," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 2016, pp. 272–285.

[161] 3GPP, "3gpp the mobile broadband standard," Available Online: https://www.3gpp.org/news-events/3gpp-news/1785-nb_, accessed (2019/11/14), 2016.

[162] P. Ferrari, A. Flammini, E. Sisinni, S. Rinaldi, D. Brandão, and M. S. Rocha, "Delay estimation of industrial iot applications based on messaging protocols," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 9, pp. 2188–2199, 2018.

[163] T. Zheng, M. Gidlund, and J. Åkerberg, "Wirarb: A new mac protocol for time critical industrial wireless sensor network applications," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2127–2139, 2015.

[164] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2011, pp. 3–12.

[165] Q. Leng, Y.-H. Wei, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "Improving control performance by minimizing jitter in rt-wifi networks," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 63–73.

[166] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for wirelesshart networks," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 150–159.

[167] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "Wirelesshart: Applying wireless technology in real-time industrial process control," in *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS'08. IEEE*. IEEE, 2008, pp. 377–386.

[168] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "Rt-wifi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 140–149.

[169] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end communication delay analysis in industrial wireless networks," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1361–1374, 2014.

[170] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability analysis under graph routing in wirelesshart networks," in *2015 IEEE Real-Time Systems Symposium*. IEEE, 2015, pp. 165–174.

[171] A. Saifullah, S. Sankar, J. Liu, C. Lu, R. Chandra, and B. Priyantha, "Capnet: A real-time wireless management network for data center power capping," in *2014 IEEE Real-Time Systems Symposium*. IEEE, 2014, pp. 334–345.

[172] L. Ascorti, S. Savazzi, G. Soatti, M. Nicoli, E. Sisinni, and S. Galimberti, "A wireless cloud network platform for industrial process automation: Critical data publishing and distributed sensing," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 4, pp. 592–603, 2017.

[173] S. M. Kim and T. He, "Freebee: Cross-technology communication via free side-channel," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 317–330.

[174] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011.

[175] P. H. Cole and D. C. Ranasinghe, "Networked rfid systems and lightweight cryptography," *London, UK: Springer*, vol. 10, pp. 978–3, 2008.

[176] H. C. Pöhls, V. Angelakis, S. Suppan, K. Fischer, G. Oikonomou, E. Z. Tragos, R. D. Rodriguez, and T. Mouroutis, "Rerum: Building a reliable iot upon privacy-and security-enabled smart objects," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2014, pp. 122–127.

[177] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[178] R. K. Yin, *Case study research and applications: Design and methods*. Sage publications, 2017.

[179] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[180] M. Kuhrmann, D. M. Fernández, and M. Daneva, "On the pragmatic design of literature studies in software engineering: an experience-based guideline," *Empirical software engineering*, vol. 22, no. 6, pp. 2852–2891, 2017.

[181] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9, pp. 833–859, 2008.

[182] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 38.

[183] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. CRC Press, 2015, vol. 4.

[184] D. Johnson, Y.-c. Hu, and D. Maltz, "The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," Tech. Rep., 2007.

[185] C. Perkins, S. Ratliff, and J. Dowdell, "Dynamic manet on-demand (aodvv2) routing draft-ietf-manet-dymo-26," *IETF, Feb*, 2013.

[186] M. Jerbi, S.-M. Senouci, T. Rasheed, and Y. Ghamri-Doudane, "Towards efficient geographic routing in urban vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5048–5059, 2009.

[187] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl)," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*. IEEE, 2011, pp. 365–372.

[188] B. Pavković, F. Theoleyre, and A. Duda, "Multipath opportunistic rpl routing over ieee 802.15. 4," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2011, pp. 179–186.

[189] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the performance of rpl and 6lowpan in tinyos," in *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN)*, vol. 80, 2011, pp. 85–90.

[190] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6lowpan protocol on micaz and telosb motes," in *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2009, pp. 25–30.

[191] S. Dawans, S. Duquennoy, and O. Bonaventure, "On link estimation in dense rpl deployments," in *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*. IEEE, 2012, pp. 952–955.

[192] E. Baccelli, M. Philipp, and M. Goyal, "The p2p-rpl routing protocol for ipv6 sensor networks: Testbed experiments," in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*. IEEE, 2011, pp. 1–6.

[193] G. Oikonomou and I. Phillips, "Stateless multicast forwarding with rpl in 6lowpan sensor networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 272–277.

[194] S. K. Hammerseth, "Implementing rpl in a mobile and fixed wireless sensor network with omnet++." Master's thesis, 2011.

[195] L. B. Saad, C. Chauvenet, and B. Tourancheau, "Simulation of the rpl routing protocol for ipv6 sensor networks: two cases studies," in *International Conference on Sensor Technologies and Applications SENSORCOMM 2011*. IARIA, 2011.

[196] U. Herberg and T. Clausen, "A comparative performance study of the routing protocols load and rpl with bi-directional traffic in low-power and lossy networks (lln)," in *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 2011, pp. 73–80.

[197] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[198] Y. Chen, J.-P. Chanet, and K. M. Hou, "Rpl routing protocol a case study: Precision agriculture," in *First China-France Workshop on Future Computing Technology (CFWoFUCT 2012)*, 2012, pp. 6–p.

[199] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg, "Implementation of coap and its application in transport logistics," *Proc. IP+ SN, Chicago, IL, USA*, 2011.

[200] M. Becker, T. Pötsch, K. Kuladinithi, and C. Goerg, "Deployment of coap in transport logistics," in *Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN). Bonn Germany 2011*, 2011.

[201] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability," in *Mobile Services (MS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 313–319.

[202] J. J. Rodrigues and P. A. Neves, "A survey on ip-based wireless sensor network solutions," *International Journal of Communication Systems*, vol. 23, no. 8, pp. 963–981, 2010.

[203] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[204] J. Kim, J. Lee, J. Kim, and J. Yun, "M2m service platforms: Survey, issues, and enabling technologies." *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 61–76, 2014.

[205] H. Kagermann, W.-D. Lukas, and W. Wahlster, "Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution," *VDI nachrichten*, vol. 13, p. 11, 2011.

[206] H. Kagermann, W. Lukas, and W. Wahlster, "Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution. vdi nachrichten 13," 2014.

[207] R. Drath and A. Horch, "Industrie 4.0: Hit or hype?[industry forum]," *IEEE industrial electronics magazine*, vol. 8, no. 2, pp. 56–58, 2014.

[208] I. I. Consortium *et al.*, "Introductory white paper," *IIC, White paper, March*, 2014.

[209] I. I. Consortium, "Industrial internet reference architecture," *Industrial Internet Consortium, Tech. Rep., June*, 2015.

[210] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[211] D. Locke, "Mqtt v3. 1 protocol specification," Technical Report, International Business Machines Corporation (IBM) and Eurotech, 2010, 42p, Tech. Rep., 2010.

[212] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," 2014.

[213] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks," in *Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*. IEEE, 2011, pp. 1–6.

[214] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating wireless sensor networks with the web," *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*, 2011.

[215] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007, pp. 78–82.

[216] D. Culler and S. Chakrabarti, "6lowpan: Incorporating ieee 802.15. 4 into the ip architecture," *IPSO Alliance White Paper*, 2009.

[217] D. Law, D. Dove, J. D'Ambrosia, M. Hajduczenia, M. Laubach, and S. Carlson, "Evolution of ethernet standards in the ieee 802.3 working group," *IEEE Communications Magazine*, vol. 51, no. 8, pp. 88–96, 2013.

[218] I. Adam Healey, "Ieee: Ieee 802.3 ethernet working group," Available at http://www.ieee802.org/3/, accessed (2018/01/18), 2018.

[219] S. Mangold, L. Berlemann, M. Siebert, and B. H. Walke, "Ieee 802.11 wireless local area networks," *IEEE 802 Wireless Systems: Protocols, Multi-Hop Mesh/Relaying, Performance and Spectrum Coexistence*, pp. 77–117, 2006.

[220] I. Adrian Stephens, "Ieee: Ieee 802.11tm wireless local area networks," Available at http://www.ieee802.org/11/, accessed (2018/01/18), 2018.

[221] T. Group, "Thread for internet of things," Available at http://threadgroup.org, accessed (2018/01/22), 2015.

[222] P. Beecher, "Wi-sun alliance," *Policy*, vol. 2, p. 3, 2013.

[223] ——, "Wi-sun alliance-interoperable communications solutions," 2016.

[224] S. Lawson, "Zigbee 3.0 promises one smart home standard for many uses. pcworld," 2014.

[225] C. Links, "The power of zigbee 3.0 all about the new and improved zigbee 3.0," 2015.

[226] TSN, "Ieee: Ieee 802.1 time-sensitive networking task group," Available at http://www.ieee802.org/1/pages/tsn.html, accessed (2018/01/22), 2012.

[227] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[228] Z.-W. Alliance, "Z-wave," 2015.

[229] ITU, "Short range narrow-band digital radiocommunication transceivers - phy and mac layer specifications," Available at https://www.itu.int/rec/T-REC-G.9959, accessed (2018/01/22), 2012.

[230] J. A. Gutierrez, E. H. Callaway, and R. L. Barrett, *Low-rate wireless personal area networks: enabling wireless sensors with IEEE 802.15. 4*. IEEE Standards Association, 2004.

[231] 3GPP, "3rd generation partnership project; technical specification group radio access network; evolved universal terrestrial radio access (e-utra); user equipment (ue) radio transmission and reception (release 10)," *Technical Specification*, vol. 36, 2010.

[232] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.

[233] "M2m and iot redefined through cost effective and energy optimized connectivity," Available at https://www.sigfox.com, SIGFOX, 2014.

[234] G. E. S. Specification, "Provisioning for m2m," *Presentation, GSMA Connected Living*, 2014.

[235] G. Wachob and D. Reed, "Extensible resource identifier (xri) resolution version 2.0, 2008," 2008.

[236] E. Hammer-Lahav and W. Norris, "Extensible resource descriptor (xrd) version 1.0," *Working Draft*, vol. 1, no. 09, 2009.

[237] M. Dürst and M. Suignard, "Internationalized resource identifiers (iris)," Tech. Rep., 2004.

[238] P. Fran O'Brien, Omar Elloumi, "Onem2m interoperability for m2m and iot technologies," Available at http://www.onem2m.org/, accessed (2018/01/16), 2012.

[239] V. T. Dominique Guinard, "Architecting the web of things for techies and thinkers," Available at https://webofthings.org, accessed (2018/01/16), 2007.

[240] S. Kim, "Ieee: P2413 - standard for an architectural framework for the internet of things (iot)," Available at https://standards.ieee.org/develop/project/2413.html, accessed (2018/01/16), 2014.

[241] K. Naito, "A survey on the internet-of-things: Standards, challenges and future prospects," *Journal of Information Processing*, vol. 25, pp. 23–31, 2017.

[242] R. van der Meulen, "5 steps to address iot integration challenges," Available at https://www.gartner.com/smarterwithgartner/five-steps-to-address-iot-integration-challenges/, accessed (2018/01/15), 2017.

[243] P. Schone, "Apis in the world of iot," Available at https://apifriends.com/2017/08/14/iot-api/, accessed (2018/01/30), 2017.

[244] M. Villari, A. Al-Anbuky, A. Celesti, and K. Moessner, "Leveraging the internet of things: Integration of sensors and cloud computing systems," 2016.

[245] G. Eastwood, "Iot's interoperability challenge, , howpublished = Available at https://www.networkworld.com/article/3205207/internet-of-things/iots-interoperability-challenge.html, accessed (2018/01/30)," 2017.

[246] A. John Thielens, "Without api management, the internet of things is just a big thing," Available at https://www.wired.com/insights/2013/07/without-api-management-the-internet-of-things-is-just-a-big-thing/, accessed (2018/01/30), 2017.

[247] H. Ning and Z. Wang, "Future internet of things architecture: like mankind neural system or social organization framework?" *IEEE Communications Letters*, vol. 15, no. 4, pp. 461–463, 2011.

[248] N. Dmitry and S.-S. Manfred, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.

[249] I. The Eclipse Foundation, "Open source software for industry 4.0, an eclipse iot working group collaboration," Made available under the Eclipse Public License 2.0 (EPL-20), Online: https://iot.eclipse.org/resources/white-papers/Eclipse%20IoT%20White%20Paper%20-%20Open%20Source%20Software%20for%20Industry%204.0.pdf, accessed (2017/10/20), 2017.

[250] E. Foundation, "Eclipse kuksa," Made available under the Eclipse Public License 2.0 (EPL-20), Online: https://projects.eclipse.org/projects/iot.kuksa, accessed (2018/06/20), 2017.

[251] B. Data, "for better or worse: 90% of world's data generated over last two years," *SCIENCE DAILY, May*, vol. 22, 2013.

[252] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[253] R. Balasubramonian, J. Chang, T. Manning, J. H. Moreno, R. Murphy, R. Nair, and S. Swanson, "Near-data processing: Insights from a micro-46 workshop," *IEEE Micro*, vol. 34, no. 4, pp. 36–42, 2014.

[254] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," *Computer networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[255] L. Atzori, D. Carboni, and A. Iera, "Smart things in the social loop: Paradigms, technologies, and potentials," *Ad Hoc Networks*, vol. 18, pp. 121–132, 2014.

[256] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things: Resource-oriented architecture and best practices," *Architecting the Internet of things*, pp. 97–129, 2011.

[257] S. Wang, Z. Zhang, Z. Ye, X. Wang, X. Lin, and S. Chen, "Application of environmental internet of things on water quality management of urban scenic river," *International Journal of Sustainable Development & World Ecology*, vol. 20, no. 3, pp. 216–222, 2013.

[258] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer *et al.*, "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, pp. 9–52, 2011.

[259] I. F. Akyildiz and J. M. Jornet, "The internet of nano-things," *IEEE Wireless Communications*, vol. 17, no. 6, 2010.

[260] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 38.

[261] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[262] S. Li and L. Da Xu, *Securing the Internet of Things*. Syngress, 2017.

[263] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation computer systems*, vol. 28, no. 3, pp. 583–592, 2012.

[264] M. Zhang, A. Raghunathan, and N. K. Jha, "Trustworthiness of medical devices and body area networks," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1174–1188, 2014.

[265] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*. IEEE, 2011, pp. 150–156.

[266] B. Darrow, "Amazon explains database glitch that impacted big customers," Available at http://fortune.com/2015/09/23/amazon-database-glitch/, accessed (2017/11/28), 2015.

[267] C. Brooks, "Amazon ec2 attack prompts customer support changes," Available at http://searchaws.techtarget.com/news/1371090/Amazon-EC2-attack-prompts-customer-support-changes, accessed (2017/11/28), 2009.

[268] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*. IEEE, 2015, pp. 336–341.

[269] H. Cai, L. Da Xu, B. Xu, C. Xie, S. Qin, and L. Jiang, "Iot-based configurable information service platform for product lifecycle management," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1558–1567, 2014.

[270] E. Vasilomanolakis, J. Daubert, M. Luthra, V. Gazis, A. Wiesmaier, and P. Kikiras, "On the security and privacy of internet of things architectures and systems," in *Secure Internet of Things (SIoT), 2015 International Workshop on*. IEEE, 2015, pp. 49–57.

[271] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014.

[272] X. Wang, S. Chellappan, W. Gu, W. Yu, and D. Xuan, "Search-based physical attacks in sensor networks," in *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*. IEEE, 2005, pp. 489–496.

[273] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart nest thermostat: A smart spy in your home," *Black Hat USA*, 2014.

[274] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey," *Security in distributed, grid, mobile, and pervasive computing*, vol. 1, p. 367, 2007.

[275] D. G. Padmavathi, M. Shanmugapriya *et al.*, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," *arXiv preprint arXiv:0909.0576*, 2009.

[276] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 49–63.

[277] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on.* IEEE, 2003, pp. 197–213.

[278] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[279] D. M. Shila and V. Venugopal, "Design, implementation and security analysis of hardware trojan threats in fpga," in *Communications (ICC), 2014 IEEE International Conference on.* IEEE, 2014, pp. 719–724.

[280] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & test of computers*, vol. 27, no. 1, 2010.

[281] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of ddos-capable iot malwares," in *Proceedings of the 1st International Conference on Security, Privacy, and Trust (INSERT), IEEE*, 2017.

[282] S. Mansfield-Devine, "Ddos goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare," *Network Security*, vol. 2016, no. 11, pp. 7–13, 2016.

[283] J. Scott and D. Spaniel, "Rise of the machines the dyn attack was just a practice run," *Institute for Critical Infrastructure Technology (ICIT)*, p. 60, 2016.

[284] A. Brandt and G. Porcu, "Home automation routing requirements in low power and lossy networks," 2010.

[285] S. Seys and B. Preneel, "Authenticated and efficient key management for wireless ad hoc networks," in *Proceedings of the 24th symposium on information theory in the Benelux, Werkgemeenschap voor Informatie-en Communicatietheorie*, 2003, pp. 195–202.

[286] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on.* IEEE, 2004, pp. 309–318.

[287] E. Y. Vasserman and N. Hopper, "Vampire attacks: draining life from wireless ad hoc sensor networks," *IEEE transactions on mobile computing*, vol. 12, no. 2, pp. 318–332, 2013.

[288] F. Stajano, "The resurrecting duckling," in *International workshop on security protocols*. Springer, 1999, pp. 183–194.

[289] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the microscope," *ESET LLC (September 2010)*, 2010.

[290] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*. ACM, 2011, pp. 355–366.

[291] G. Noubir and G. Lin, "Low-power dos attacks in data wireless lans and countermeasures," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 29–30, 2003.

[292] Q. I. Sarhana, "Security attacks and countermeasures for wireless sensor networks: Survey," *International Journal of Current Engineering and Technology*, vol. 3, no. 2, pp. 628–635, 2013.

[293] A. D. Wood, J. A. Stankovic, and G. Zhou, "Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE, 2007, pp. 60–69.

[294] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short paper: reactive jamming in wireless networks: how realistic is the threat?" in *Proceedings of the fourth ACM conference on Wireless network security*. ACM, 2011, pp. 47–52.

[295] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in wsns," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, 2009.

[296] A. Mukherjee, "Physical-layer security in the internet of things: Sensing and communication confidentiality under resource constraints," *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1747–1761, 2015.

[297] Z. Karakehayov, "Using reward to detect team black-hole attacks in wireless sensor networks," *Wksp. Real-World Wireless Sensor Networks*, pp. 20–21, 2005.

[298] B. Revathi and D. Geetha, "A survey of cooperative black and gray hole attack in manet," *International Journal of Computer Science and Management Research*, vol. 1, no. 2, pp. 205–208, 2012.

[299] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, p. 794326, 2013.

[300] V. P. Singh, S. Jain, and J. Singhai, "Hello flood attack and its countermeasures in wireless sensor networks," *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 11, pp. 23–27, 2010.

[301] O. Garcia-Morchon, S. Kumar, S. Keoh, R. Hummen, and R. Struik, "Security considerations in the ip-based internet of things draft-garciacore-security-06," *Internet Engineering Task Force*, 2013.

[302] J. R. Douceur, "The sybil attack," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 251–260.

[303] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. IEEE, 2014, pp. 1–8.

[304] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[305] S. Boyd and A. Keromytis, "Sqlrand: Preventing sql injection attacks," in *Applied Cryptography and Network Security*. Springer, 2004, pp. 292–302.

[306] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 50–57, 2011.

[307] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and internet of things," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 23–30.

[308] S. Aguzzi, D. Bradshaw, M. Canning, M. Cansfield, P. Carter, G. Cattaneo, S. Gusmeroli, G. Micheletti, D. Rotondi, and R. Stevens, "Definition of a research and innovation policy leveraging cloud computing and iot combination," *Final Report, European Commission, SMART*, vol. 37, p. 2013, 2013.

[309] Z. Bi, L. Da Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," *IEEE Transactions on industrial informatics*, vol. 10, no. 2, pp. 1537–1546, 2014.

[310] Y. Chen, F. Han, Y.-H. Yang, H. Ma, Y. Han, C. Jiang, H.-Q. Lai, D. Claffey, Z. Safar, and K. R. Liu, "Time-reversal wireless paradigm for green internet of things: An overview," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 81–98, 2014.

[311] S. Raza, T. Voigt, and V. Jutvik, "Lightweight ikev2: a key management solution for both the compressed ipsec and the ieee 802.15. 4 security," in *Proceedings of the IETF workshop on smart object security*, vol. 23, 2012.

[312] H.-Y. Tsai, M. Siebenhaar, A. Miede, Y. Huang, and R. Steinmetz, "Threat as a service?: Virtualization's impact on cloud security," *IT professional*, vol. 14, no. 1, pp. 32–37, 2012.

[313] Y. Xia, Y. Liu, H. Chen, and B. Zang, "Defending against vm rollback attack," in *Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on*. IEEE, 2012, pp. 1–5.

[314] M. Lindner, F. McDonald, B. McLarnon, and P. Robinson, "Towards automated business-driven indication and mitigation of vm sprawl in cloud supply chains," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 1062–1065.

[315] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 17, 2013.

[316] C. Vulnerabilities, "Common vulnerabilities and exposures," 2005.

[317] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in multi-tenancy cloud," in *Security Technology (ICCST), 2010 IEEE International Carnahan Conference on*. IEEE, 2010, pp. 35–41.

[318] W. Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a service security: Challenges and solutions," in *Informatics and Systems (INFOS), 2010 the 7th International Conference on*. IEEE, 2010, pp. 1–8.

[319] A. Aviram, S. Hu, B. Ford, and R. Gummadi, "Determinating timing channels in compute clouds," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*. ACM, 2010, pp. 103–108.

[320] H. AlJahdali, A. Albatli, P. Garraghan, P. Townend, L. Lau, and J. Xu, "Multi-tenancy in cloud computing," in *Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on*. IEEE, 2014, pp. 344–351.

[321] A. Carlin, M. Hammoudeh, and O. Aldabbas, "Defence for distributed denial of service attacks in cloud computing," *Procedia Computer Science*, vol. 73, pp. 490–497, 2015.

[322] D. Catteddu and G. Hogben, "Cloud computing risk assessment," *European Network and Information Security Agency (ENISA)*, pp. 583–592, 2009.

[323] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[324] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[325] W. Du and Z. Zhan, "Building decision tree classifier on private data," in *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*. Australian Computer Society, Inc., 2002, pp. 1–8.

[326] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[327] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013.

[328] K. Goeschel, "Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive bayes for off-line analysis," in *SoutheastCon, 2016*. IEEE, 2016, pp. 1–6.

[329] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[330] S. Alharbi, P. Rodriguez, R. Maharaja, P. Iyer, N. Subaschandrabose, and Z. Ye, "Secure the internet of things with challenge response authentication in fog computing," in *Performance Computing and Communications Conference (IPCCC), 2017 IEEE 36th International*. IEEE, 2017, pp. 1–2.

[331] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[332] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[333] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[334] W. Hu, Y. Liao, and V. R. Vemuri, "Robust support vector machines for anomaly detection in computer security." in *ICMLA*, 2003, pp. 168–174.

[335] Y. Liu and D. Pi, "A novel kernel svm algorithm with game theory for network intrusion detection." *KSII Transactions on Internet & Information Systems*, vol. 11, no. 8, 2017.

[336] C. Wagner, J. François, T. Engel *et al.*, "Machine learning approach for ip-flow record anomaly detection," in *International Conference on Research in Networking*. Springer, 2011, pp. 28–39.

[337] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear svm-based android malware detection for reliable iot services," *Journal of Applied Mathematics*, vol. 2014, 2014.

[338] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, pp. 1773–1786, 2016.

[339] G. D'Agostini, "A multidimensional unfolding method based on bayes' theorem," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 362, no. 2-3, pp. 487–498, 1995.

[340] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[341] M. Swarnkar and N. Hubballi, "Ocpad: One class naive bayes classifier for payload based anomaly detection," *Expert Systems with Applications*, vol. 64, pp. 330–339, 2016.

[342] M. Panda and M. R. Patra, "Network intrusion detection using naive bayes," *International journal of computer science and network security*, vol. 7, no. 12, pp. 258–263, 2007.

[343] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012.

[344] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011, vol. 40.

[345] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.

[346] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[347] Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[348] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[349] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[350] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in neural information processing systems*, 2013, pp. 190–198.

[351] H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*, 2018.

[352] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[353] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for botnet detection behavior," in *Biennial Congress of Argentina (ARGENCON), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[354] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619.

[355] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013.

[356] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb *et al.*, "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices." in *HealthTech*, 2013.

[357] M. Msgna, K. Markantonakis, and K. Mayes, "The b-side of side channel leakage: Control flow security in embedded systems." in *SecureComm*. Springer, 2013, pp. 288–304.

[358] A. Nejat, S. M. H. Shekarian, and M. S. Zamani, "A study on the efficiency of hardware trojan detection based on path-delay fingerprinting," *Microprocessors and Microsystems*, vol. 38, no. 3, pp. 246–252, 2014.

[359] N. Yoshimizu, "Hardware trojan detection by symmetry breaking in path delays," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 107–111.

[360] A. N. Nowroz, K. Hu, F. Koushanfar, and S. Reda, "Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1792–1805, 2014.

[361] M. Tehranipoor, H. Salmani, and X. Zhang, "Hardware trojan detection: Untrusted manufactured integrated circuits," in *Integrated Circuit Authentication*. Springer, 2014, pp. 31–38.

[362] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar, "High-sensitivity hardware trojan detection using multimodal characterization," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 2013, pp. 1271–1276.

[363] N. Lesperance, S. Kulkarni, and K.-T. Cheng, "Hardware trojan detection using exhaustive testing of k-bit subspaces," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*. IEEE, 2015, pp. 755–760.

[364] X. Ye, J. Feng, H. Gong, C. He, and W. Feng, "An anti-trojans design approach based on activation probability analysis," in *Electron Devices and Solid-State Circuits (EDSSC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 443–446.

[365] R. S. Chakraborty, F. G. Wolff, S. Paul, C. A. Papachristou, and S. Bhunia, "Mero: A statistical approach for hardware trojan detection." in *CHES*, vol. 5747. Springer, 2009, pp. 396–410.

[366] S. S. Doumit and D. P. Agrawal, "Self-organized criticality and stochastic learning based intrusion detection system for wireless sensor networks," in *Military Communications Conference, 2003. MILCOM'03. 2003 IEEE*, vol. 1. IEEE, 2003, pp. 609–614.

[367] M. S. I. Mamun, A. Kabir, M. Hossen, M. Khan, R. Hayat *et al.*, "Policy based intrusion detection and response system in hierarchical wsn architecture," *arXiv preprint arXiv:1209.1678*, 2012.

[368] A. P. R. da Silva, M. H. Martins, B. P. Rocha, A. A. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*. ACM, 2005, pp. 16–23.

[369] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[370] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc networks*, vol. 1, no. 2, pp. 293–315, 2003.

[371] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 78–87.

[372] M. Cagalj, S. Capkun, and J.-P. Hubaux, "Wormhole-based antijamming techniques in sensor networks," *IEEE transactions on Mobile Computing*, vol. 6, no. 1, 2007.

[373] S. Misra and A. Vaish, "Reputation-based role assignment for role-based access control in wireless sensor networks," *Computer Communications*, vol. 34, no. 3, pp. 281–294, 2011.

[374] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[375] M. Katagi and S. Moriai, "Lightweight cryptography for the internet of things," *Sony Corporation*, pp. 7–10, 2008.

[376] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher clefia," in *FSE*, vol. 4593. Springer, 2007, pp. 181–195.

[377] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *CHES*, vol. 4727. Springer, 2007, pp. 450–466.

[378] S. Son, K. S. McKinley, and V. Shmatikov, "Diglossia: detecting code injection attacks with precision and efficiency," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1181–1192.

[379] W. G. Halfond and A. Orso, "Amnesia: analysis and monitoring for neutralizing sql-injection attacks," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, 2005, pp. 174–183.

[380] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.

[381] H. Mouratidis and P. Giorgini, "Security attack testing (sat)—testing the security of information systems at design time," *Information systems*, vol. 32, no. 8, pp. 1166–1183, 2007.

[382] R. V. Deshmukh and K. K. Devadkar, "Understanding ddos attack & its effect in cloud environment," *Procedia Computer Science*, vol. 49, pp. 202–210, 2015.

[383] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*. IEEE, 2009, pp. 1–11.

[384] D. Sun, G. Chang, L. Sun, and X. Wang, "Surveying and analyzing security, privacy and trust issues in cloud computing environments," *Procedia Engineering*, vol. 15, pp. 2852–2856, 2011.

[385] C. Alliance, "Security guidance for critical areas of focus in cloud computing v3. 0," *Cloud Security Alliance*, p. 15, 2011.

[386] Y. Tang, P. P. Lee, J. C. Lui, and R. Perlman, "Fade: Secure overlay cloud storage with file assured deletion," *Security and Privacy in Communication Networks*, pp. 380–397, 2010.

[387] M. Aslam, C. Gehrmann, and M. Björkman, "Security and trust preserving vm migrations in public clouds," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 869–876.

[388] B. Ding, Y. He, Y. Wu, and Y. Lin, "Hyperverify: a vm-assisted architecture for monitoring hypervisor non-control data," in *Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on*. IEEE, 2013, pp. 26–34.

[389] F. Rocha, T. Gross, and A. van Moorsel, "Defense-in-depth against malicious insiders in the cloud," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. IEEE, 2013, pp. 88–97.

[390] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[391] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[392] I. Friese, J. Heuer, and N. Kong, "Challenges from the identities of things: Introduction of the identities of things discussion group within kantara initiative," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 1–4.

[393] P. N. Mahalle, B. Anggorojati, N. R. Prasad, R. Prasad *et al.*, "Identity authentication and capability based access control (iacac) for the internet of things," *Journal of Cyber Security and Mobility*, vol. 1, no. 4, pp. 309–348, 2013.

[394] P. Otte, M. de Vos, and J. Pouwelse, "Trustchain: A sybil-resistant scalable blockchain," *Future Generation Computer Systems*, 2017.

[395] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of things security and forensics: Challenges and opportunities," 2018.

[396] S. Zawoad and R. Hasan, "Faiot: Towards building a forensics aware eco system for the internet of things," in *Services Computing (SCC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 279–284.

[397] M. B. Barcena and C. Wueest, "Insecurity in the internet of things," *Security Response, Symantec*, 2015.

[398] E. McKenna, I. Richardson, and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," *Energy Policy*, vol. 41, pp. 807–814, 2012.

[399] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis." in *USENIX Security Symposium*, 2015, pp. 785–800.

[400] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*. IEEE, 2012, pp. 1–9.

[401] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion." *HotSec*, vol. 11, pp. 9–9, 2011.

[402] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "Deepx: A software accelerator for low-power deep learning inference on mobile devices," in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*. IEEE Press, 2016, p. 23.

[403] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017.

[404] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked aes," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, 2015.

[405] A. Heuser and M. Zohner, "Intelligent machine homicide," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 249–264.

[406] S. Greydanus, "Learning the enigma with recurrent neural networks," *arXiv preprint arXiv:1708.07576*, 2017.

[407] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Mining energy-greedy api usage patterns in android apps: an empirical study," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 2–11.

[408] N. Nikzad, O. Chipara, and W. G. Griswold, "Ape: an annotation language and middleware for energy-efficient mobile application development," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 515–526.

[409] J. H. Stapleton, *Models for probability and statistical inference: theory and applications*. John Wiley & Sons, 2007, vol. 652.

[410] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys," in *annual meeting of the Florida Association of Institutional Research*, 2006, pp. 1–33.

[411] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.

[412] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating instrumentation data to system states: A building block for automated diagnosis and control." in *OSDI*, vol. 4, 2004, pp. 16–16.

[413] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.