FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Privacy-oriented Task Orchestration System for IoT Networks

Nuno Tiago Tavares Lopes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Hugo Sereno Ferreira, Assistant Professor

Second Supervisor: Tiago Boldt Sousa, Assistant Professor

FhP AICOS Supervisors: Marcos Liberal, Pedro Madureira

July 16, 2021

Privacy-oriented Task Orchestration System for IoT Networks

Nuno Tiago Tavares Lopes

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. João Pedro Correia dos Reis
External Examiner: Prof. Ângelo Manuel Rego e Silva Martins
Supervisor: Prof. Hugo José Sereno Lopes Ferreira

July 14, 2021

Abstract

The Internet-of-Things (IoT) represents a network of heterogeneous devices embedded with sensors and other technologies connected to the Internet. The usage of IoT systems in areas such as home automation, industry, and health scenarios has been rising over time. However, most of these existing IoT systems were built following a centralized approach where the main component manages and executes most of the computation on the data provided by edge devices (*e.g.*, sensors, actuators). These approaches neglect the edge devices' computation capabilities, and local data may be transferred across boundaries without need.

With Fog and Edge Computing's recent appearance, the storage and computation of data were moved to the edge of the network, thus closer to the users, making better use of the computational capabilities of edge devices. Herewith, orchestration mechanisms capable of decomposing the system into smaller tasks and allocating them to edge devices began to emerge. However, these devices are usually less secure since they have lower capabilities and energy constraints when compared to the execution of computation in cloud devices. Moreover, many of the edge devices generate, process, and exchange privacy-sensitive data, thus are more appealing targets for attacks that aim to access private data. Although several approaches allow the orchestration of distributed IoT systems, they lack of protection mechanisms that guarantee the privacy of data flowing in the system.

In this work, we propose mechanisms that can automatically guarantee a safe flow of privacy-sensitive data in IoT systems where computational tasks are orchestrated to devices at the edge of the network. To achieve this, we extend an existing platform, NoRDOr, which consists of a modified version of Node-RED capable of orchestrating distributed IoT systems in real-time. We address privacy concerns in this platform by (1) preventing privacy-sensitive data from being sent to possible unsafe devices, (2) making use of the computational power of the available devices without compromising private data, and (3) by allowing the orchestrator to prioritize the allocation of privacy-sensitive nodes to safer locations.

In order to validate and evaluate our solution, we performed virtual experiments with simulated IoT devices, where faults were introduced to simulate real-world scenarios. Several metrics were collected from the devices to better understand and analyze if our solution protected the privacy-sensitive data but at the same time used the computational power provided by the edge devices. We conclude that the proposed improvements would lead to a system where the privacy of private data is ensured. We further identified some limitations and research work to improve our solution in the future.

Keywords: IoT, Privacy, Orchestration, Edge Computing, Node-RED, Fog Computing

Resumo

A Internet-of-Things (IoT) representa uma rede de dispositivos heterogéneos que incorporam sensores e outras tecnologias que estão ligados à Internet. O uso de sistemas IoT em áreas como a domótica, a indústria e em cenários de saúde tem vindo a aumentar ao longo do tempo. No entanto, a maioria dos sistemas IoT existentes foram construídos seguindo uma abordagem centralizada, onde um componente principal gere e executa a maior parte da computação sobre os dados fornecidos por dispositivos *edge* (*p.e.*, sensores, atuadores). Estas abordagens negligenciam o poder computacional dos dispositivos *edge* que por sua vez leva a que os dados locais sejam transferidos através das fronteiras sem necessidade.

Com o recente surgimento dos paradigmas Fog e Edge Computing, o armazenamento e a computação dos dados começaram a ser movidos para a periferia da rede, ou seja, mais perto dos utilizadores, utilizando melhor as capacidades computacionais dos dispositivos *edge*. Com isto, começaram a surgir mecanismos capazes de decompor os sistemas em tarefas mais pequenas que são orquestradas e alocadas a dispositivos *edge*. No entanto, estes dispositivos são geralmente menos seguros pois têm menos recursos e mais restrições de energia quando comparados com os dispositivos *cloud*. Além disso, muitos dos dispositivos *edge* geram, processam e trocam dados privados, sendo assim alvos bastante atrativos para ataques que visam obter acesso a este tipo de dados. Embora existam várias abordagens que permitem a orquestração de sistemas IoT distribuídos, não existe um grande foco na utilização de mecanismos capazes de garantir a privacidade dos dados que circulam pelo sistema.

Neste trabalho, propomos mecanismos que permitem garantir automaticamente um fluxo seguro de dados sensíveis em sistemas IoT e que ao mesmo tempo aproveitem as capacidades dos dispositivos *edge* enviando tarefas para eles executarem. De forma a atingirmos o nosso objetivo, estendemos uma plataforma existente, NoRDOr, que consiste numa versão modificada do Node-RED que é capaz de orquestrar em tempo real tarefas em sistemas IoT distribuídos. Para abordar as questões de privacidade nesta plataforma, a nossa solução (1) evita que dados privados sejam enviados para dispositivos considerados inseguros, (2) utiliza o poder computacional dos dispositivos disponíveis sem comprometer a privacidade de dados sensíveis e (3) permite a prioritização de alocação de nós sensíveis em locais mais seguros durante a orquestração.

Para validar e avaliar nossa solução, foram realizas experiências virtuais com dispositivos IoT simulados, onde foram introduzidas falhas para simular cenários do mundo real. Durante as experiências foram recolhidas várias métricas dos dispositivos para melhor entender e analisar se a nossa solução protegia os dados privados e ao mesmo tempo usava o poder computacional fornecido pela rede. Concluímos que as melhorias sugeridas levariam a um sistema onde a privacidade dos dados privados é garantida quando comunicada entre dispositivos. Foram também identificadas algumas limitações à nossa solução que podem ser endereçadas em trabalho futuro.

Keywords: IoT, Privacy, Orchestration, Edge Computing, Node-RED, Fog Computing

Acknowledgements

I would like to start by thanking everyone that made it possible for me to finish this dissertation and my *Master's Degree*.

First of all, I would like to thank my supervisors from FEUP, Professors Hugo Sereno Ferreira and Tiago Boldt Sousa, and my supervisors from FhP AICOS, Marcos Liberal and Pedro Madureira, for the guidance and insights provided throughout these last months which helped me produce a better work.

Then, I would like to thank my parents and my sister, for their love and encouragement, for being a critical pillar of my life providing me with the best conditions possible to be successful. I would not be where I am today without you.

To Inês Bernardo, my girlfriend, for all the love and support, for always being there for me, motivating me to be better every day.

Lastly, but not least, a huge thank you to all my friends who accompanied me on this journey through university. Without you, it would not be as memorable as it was. Thank you for the memories that will persist for many years.

Nuno Lopes

"You can't put a limit on anything.
The more you dream, the farther you get."

Michael F. Phelps

Contents

List of Figures

List of Tables

Abbreviations

AAL	Ambient Assisted Living
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ES	Experimental Scenario
HTTP	Hypertext Transfer Protocol
IIoT	Industrial Internet-of-Things
IoT	Internet-of-Things
MQTT	Message Queuing Telemetry Transport
NoRDOr	Node-RED Distributed Orchestrator
REST	Representational State Transfer
RQ	Research Question
SA	Simulated Annealing
SC	Sanity Check
URL	Uniform Resource Locator
VPL	Visual Programming Language
WWW	World Wide Web

Chapter 1

Introduction

This chapter introduces the context and motivation behind this dissertation, along with a short definition of the problem this work aims to solve. Section 1.1 details the context of this project in the area of Internet-of-Things and its applications. Section 1.2 (p. 2) defines some of the issues of current systems and their consequences. Section 1.3 (p. 3) explains the problem we intend to tackle, motivated by the issues mentioned in the previous section, while Section 1.4 (p. 3) details the goals this dissertation aims to achieve and how they are validated and evaluated. Finally, Section 1.5 (p. 4) gives a short overview of this document structure and what should be expected from each chapter.

1.1 Context

The Internet-of-Things (IoT) is the name given to a unified network of intelligent objects capable of communicating with each other in a universal and ubiquitously way [15]. In 2014, the number of "things" that were connected was around 20 billion [3]. It is expected that within six years, that number will increase to 30 billion. Furthermore, the applications are endless and will increase day after day [3]. Some examples of where IoT systems can be applied are smart homes, smart cities [49], industry [20, 38], and health [74, 59].

In the industry scenario, the advances in IoT sensors made possible the development of embedded and connected systems, which aimed to monitor and control the equipment, transporters, and products through a feedback cycle that collects large amounts of data (big data). This data is used to update virtual models with the physical processes' information, resulting in a smart factory

[38]. However, Industrial Internet-of-Things (IIoT) networks can also be used to monitor factory workers' health and well-being [45]. The data collected from these sources is much more personal and consequently more privacy-sensitive since it mostly belongs to the operators. IIoT systems that deal with this kind of data must ensure that it is correctly secured and can only be accessed by authorized entities.

Internet-of-Things systems also began to be used in Ambient Assisted Living (AAL) scenarios helping older adults and people with special needs during their daily routine by fostering their autonomy and increasing their safety by monitoring them in real-time [59]. Similarly, the data collected in these systems is highly private, thus the systems should provide ways to guarantee the privacy of data without compromising the real-time flow of it, which is critical in this case.

For these referred scenarios, it is transversal that users require the protection of their personal information related to their movements, habits, and interactions with other people. Thus, the privacy of all the data generated, processed, and exchanged by smart objects should be ensured [66]. However, the implementation of such mechanisms comes with challenges as edge devices used to collect data are highly heterogeneous, have lower computational power and storage capacity, and are more vulnerable compared to fog or cloud servers [4].

1.2 Motivation

The usage of Internet-of-Things systems in areas such as home automation, industry, and health has been rising over time [16, 21]. Most of these existing IoT Systems were built following a centralized approach where the main component manages and executes most of the computation on the data provided by edge devices (*e.g.*, sensors, actuators) [67, 68]. Thus, these approaches neglect the edge devices' computation capabilities making local data to be transferred across certain boundaries, which increases the risk of it being compromised.

With the increase of the computational capabilities of even the smaller devices in the Internet-of-Things, the Fog and Edge computing paradigms started to emerge [41, 39, 37, 40]. The computation and storage of the collected data began to be performed at the edge of the network, thus closer to the users. Many of these edge devices began to generate, process, and exchange more security and safety-critical data as well as privacy-sensitive information, and hence are appealing targets of various attacks since they are less secure due to the low capabilities and energy constraints [2, 75].

The broad range of IoT application scenarios impulses the need for tools that enable non-technical users to setup and adjust their systems to their requirements [71]. As a result, several low-code programming solutions emerged to reduce programming and configuring complexity of these systems, leveraging both visual and conversational interfaces [13, 21, 27, 46]. One of such development solutions' is Node-RED, being one of the most popular and widespread visual programming tools for IoT [22].

The security and privacy of IoT systems are key factors for deploying new applications since people will only accept these deployments if they are based on secure, trustworthy, and privacy-preserving infrastructures [65, 49, 58, 24]. Therefore, it is crucial to ensure the privacy of data produced, processed, and exchanged between the devices in these systems.

1.3 Problem

The majority of solutions that leverage the computational capabilities of distributed edge devices in an IoT network by orchestrating custom tasks do not have mechanisms to protect possible privacy-sensitive data that flows in the system. According to Ni *et al*. [54], the four most relevant types of information that an IoT system should always protect are (1) identity, (2) sensitive data, (3) information usage, and (4) location data. In some use cases, other types of information also needs to be protected. Thus, it should be a users' choice what type of information they wish to keep private the system.

In this work, we intend to research privacy methods that can be used in IoT systems without compromising the system's ability to take advantage of the computational power of the network. However, when choosing the privacy mechanisms, we have to bear in mind that most edge devices have low capabilities and energy constraints.

Chapter 4 further analyses the problem under study in this dissertation, defining its scope, desiderata, and research questions that guide the development of our work and are later answered in the experiments.

1.4 General Goals

The main goal of this dissertation is to develop mechanisms that can automatically guarantee a safe flow of privacy-sensitive data in distributed IoT systems. To achieve our goal, we extend a previously developed platform, NoRDOr [19, 67], which is comprised of a modified version of Node-RED with the ability to leverage the computational capabilities of edge devices in an IoT network. In this platform, Silva *et al*. also developed a custom MicroPython firmware which allowed the devices to execute custom scripts of MicroPython that were sent by an orchestrator running in the enhanced version of Node-RED.

To ensure the privacy of data in the NoRDOr platform, our solution should (1) prevent privacy-sensitive from being sent to possible unsafe devices, (2) make use of the computational power of the available devices without compromising private data, and (3) be capable of prioritizing the allocation of privacy-sensitive *nodes* to safer locations.

To validate the proposed solutions, we have performed several experiments with virtual IoT devices, where faults were introduced to simulate real-world scenarios [26]. The results of these experiments were analyzed, providing answers to each one of the research questions that guided the development of this dissertation.

1.5 Document Structure

This chapter serves to introduce the motivation and scope of this project, as well as the problem it aims to solve. This document contains six more chapters, structured as follow:

- Chapter 2 (p. 5), **Background**, introduces some concepts necessary to fully understand this dissertation;
- Chapter 3 (p. 13), **State of the Art**, describes the state of the art regarding the scope of this project, including a literature review on orchestration of distributed IoT systems and a literature review on privacy techniques applied to the orchestration of Internet-of-Things systems;
- Chapter 4 (p. 27), **Problem Statement**, presents the current issues under study, the requirements our solution should fulfill and the research questions that will guide the development of this dissertation;
- Chapter 5 (p. 33), **Privacy-oriented Task Orchestration**, details the implementation of our solution and discusses the reasons behind every choice made;
- Chapter 6 (p. 41), **Experimentation and Evaluation**, presents the experiments performed and analysis of the respective results which are used to answer the research questions of this dissertation;
- Chapter 7 (p. 69), **Conclusions**, summarizes the work developed during this dissertation, outlines the main difficulties and contributions, and also suggests future directions to improve our work.

Chapter 2

Background

This chapter describes the key concepts and the relationships between them, which are necessary to fully understand this work. Section 2.1 defines the term Internet-of-Things while Section 2.1.1 (p. 6) describes the tiers and computing paradigms related to it. Industrial IoT and Ambient Assisted Living are common scenarios where IoT systems are used. For both cases, the IoT systems used in these areas can generate vast amounts of privacy-sensitive data thus they are good examples where data needs to be protected. In Section 2.1.2 (p. 8), we present a brief description of the concepts of Industrial IoT and Industry 4.0 while in Section 2.1.3 (p. 8) we define the term Ambient Assisted Living and explain how IoT systems are used in AAL scenarios. Finally, Section 2.2 (p. 9) explains the current concerns in privacy and security in IoT, presenting the model CIA Triad, which is commonly used as guidance in the efforts and policies aimed at keeping data secure.

2.1 Internet-of-Things

The term Internet-of-Things (IoT) does not have a unique definition acceptable by the whole community of users. Instead, this term was defined by different groups of people. Atzori *et al.* [8] define IoT as:

> *"A conceptual framework that leverages on the availability of heterogeneous devices and interconnection solutions, as well as augmented physical objects providing a shared information base on global scale, to support the design of applications involving at the same virtual level both people and representations of objects."*

However, Madakam *et al.* [51] believe the best description for the Internet-of-Things is:

> *"An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment."*

This network of smart objects appeared to solve people's problems — limited time, accuracy, and attention — when capturing data about things in the real world [5]. According to Atzori *et al*. [7], the applications of IoT can be grouped in the following domains: (1) *Transportation and logistics* (*e.g*., assisted driving, environment monitoring, augmented maps), (2) *Healthcare* (*e.g*., patients tracking, identification, and authentication of patients), (3) *Smart environment*, (*e.g*., comfortable homes and offices, industrial plants) and (4) *Personal and social*, (*e.g*., social networking, historical queries). The authors also define a *futuristic* domain since several applications rely on technologies that either do not exist yet or whose implementation is still too complex to accept them as practical applications (*e.g*., autonomous taxi, enhanced game room).

2.1.1 IoT Tiers

Most IoT systems manage a vast amount of data that needs to be processed efficiently by taking into account the computation power required and reducing costs, resources and time. These IoT systems are generally composed of three tiers [79]:

Cloud Tier: is characterized for having more computational power and storage capacity, however, higher latency. They are mostly composed of data centers and servers.

Fog Tier: is composed of gateways and devices which stand between the cloud and the edge tiers. It has lower computation capacities than the cloud tier. However, it has less latency. It is also characterized for being more geographically and logically distributed and for having a higher heterogeneity.

Edge Tier: provides less latency but also less computational power due to the low capabilities and energy constraints of edge devices (*e.g*., sensors, actuators, embedded systems).

Nowadays, most of the existing IoT systems follow a Cloud-based approach to process and store the system's vast data. This type of approach may introduce problems in terms of latency, network traffic management, and power consumption. Moreover, for the applications that rely on real-time processing of data, the delay caused by the communication between the devices that collect the data and the cloud where it is processed can negatively impact the system's performance [32]. With this in mind and the increase of computational capabilities of even the smallest devices, new paradigms have emerged, allowing to process data closer to where it was generated, reducing system latency and network traffic. These new computation paradigms are named: Fog Computing and Edge Computing.

2.1.1.1 Fog Computing

Fog Computing was introduced in 2012 by Bonomi *et al*. [14] in the following way:

> *"(...) a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network."*

Fog Computing brings the computation and storage services closer to the edge devices removing all the cloud responsibility. The computation can be powered by high-performance servers or even typical network devices — *e.g.*, routers, gateways — that are closer to the end devices. Fog Computing is characterized by low latency, location awareness, geographical distribution, a vast number of nodes, heterogeneity, and real-time applications [14]. It makes computing resources and intelligent services more flexible, accessible, efficient and cost-effective in the IoT network [79].

Regardless of all the benefits of Fog Computing, there are several issues when compared to Cloud Computing [52]: (1) some of the devices that can be used in Fog Computing are not prepared for computing tasks, and equipping them to do so can be challenging, (2) not all nodes are suitable for specific processing needs, so it can be challenging when deploying them, (3) since Fog Computing uses traditional networking devices, it is more vulnerable to security attacks, and it is harder to ensure privacy.

2.1.1.2 Edge Computing

Shi *et al*. [64] refer Edge Computing as:

> *"(...) enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services. Here we define "edge" as any computing and network resources along the path between data sources and cloud data centers."*

Since Edge Computing wants to put the computing at the proximity of data sources it has less resources and capabilities but also lower latency which is perfect to handle delay-sensitive tasks (*e.g.*, data collection and compression, information extraction and event monitoring) at the local level [79].

However, it brings several challenges to the IoT systems, such as energy and computing constraints of the edge devices and system reliability, due to the difficulty of detecting failures in this type of devices. Moreover, there is a lack of frameworks and tools to build these systems and it is missing standard ways of naming, and addressing the edge devices.

Keeping data where it is collected lets the users fully own their data and adds the ability to denature data (*e.g.*, blur faces in images, aggregate sensor reading, omit specific fields) before releasing it to the cloud or untrusted devices, which is a better solution for privacy protection. However, edge devices are highly resource constrained, which prevents the deployment of the current security protection methods. Moreover, the highly dynamic environment at the edge of the network also makes the network more vulnerable and unprotected [63, 64].

2.1.2 Industrial Internet-of-Things & Industry 4.0

Industrial Internet-of-Things (IIoT) and Industry 4.0 are two terms that are often used mutually, however, they slightly differ.

Industry 4.0 refers to the current fourth generation of industry focusing on the manufacturing industry scenario. The term was conceived in 2011 by a German initiative of the federal government with universities and private companies with the objective of increasing productivity and efficiency of the national industry. Industry 4.0 relies on the adoption of digital technologies to gather data in real time and analyze it, providing useful information that adds value to the whole product life cycle [35, 1].

IIoT was first introduced in 2012 as the industrial Internet demanded the adoption of the IoT in the perspective of industry in general (both manufacturing and non-manufacturing). It consists of connecting all the industrial assets — machines and control systems — with the business processes and information systems. Consequently, a large amount of data is collected and used to feed analytical models that improve industrial operations' efficiency and productivity [69].

Figure 2.1 presents the concept of IIoT and Industry 4.0 within IoT.

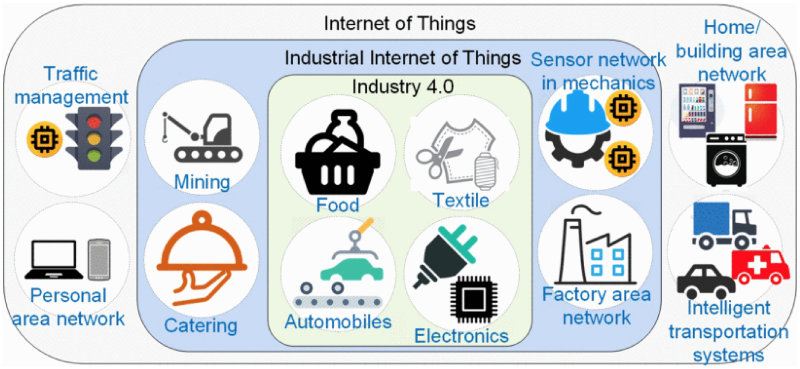

Figure 2.1: IoT, IIoT, and Industry 4.0 [1]. This image represents how the IoT, IIoT and Industry 4.0 terms correlate between them. Industry 4.0 applications are the most specific and are included in the IIoT concept, which in turn is represented by IoT in general.

Industry 4.0 would not exist without IIoT, but IIoT would not be very effective without the bigger-picture framework of Industry 4.0. IIoT is, at its core, about connecting devices, while Industry 4.0 is a cultural philosophy about how to be more competitive by increasing visibility, flexibility, and efficiency across the production. Despite the differences, IIoT and Industry 4.0 are intended to improve manufacturing processes and should be pursued and implemented to achieve positive results and sustain global competitiveness [1].

2.1.3 IoT for Ambient Assisted Living

Ambient Assisted Living (AAL) term is used to define a system that focuses on providing assistance to older adults and people with special needs in their daily routine. The main goal of AAL is

to *"maintain and foster the autonomy of those people and, thus, to increase safety in their lifestyle and in their home environment"* [30].

According to Calvaresi *et al.* [17] there are several objectives behind the AAL systems: (1) extend the time people can live in an environment they are used to by increasing their autonomy, self-confidence, and mobility, (2) maintain health and functional capability of the elderly individuals, (3) promote a better and healthier lifestyle for individuals at risk, (4) enhance security, (5) prevent social isolation, and (6) support caregivers, families and care organizations.

There is a strong relationship between AAL and IoT. The evolution of technology allows the creation of new advanced systems that can be more accurate in the prevention of problems that may arise for patients. By using Internet-of-Things devices and sensors, it is possible to develop solutions that are closer to the elderly and to the doctors and relatives [43, 59]. By using these types of smart objects, thus smart homes, it is possible to collect data in real-time from the elderly lifestyle and make it more accessible to emergency services which sometimes can be critical when there is n urgent need to provide help to the patient.

Figure 2.2 presents a possible general organization of an AAL scenario with the groups of people who are part of the system, as well as some devices are normally used.

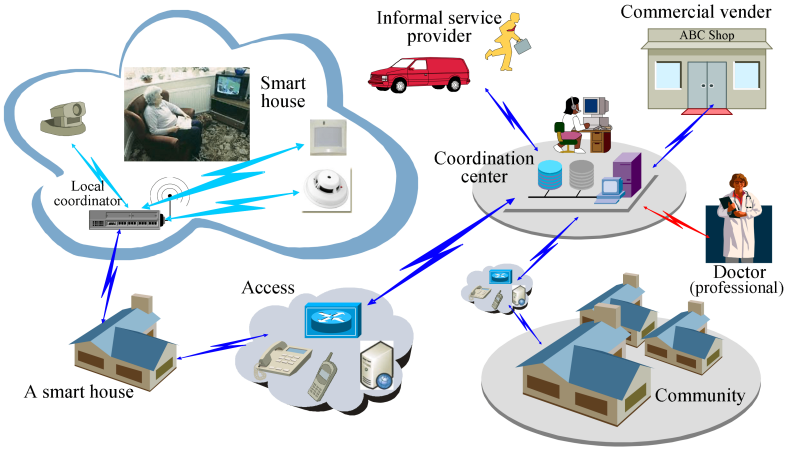

Figure 2.2: Example of an AAL scenario [74].

2.2 Privacy & Security Concerns in IoT

The increase of IoT systems and the increase of personal information generated, processed, and exchanged by smart objects in IoT networks raised several privacy and security concerns. Wherever there are valuable resources, there are also criminal attempts to obtain value from the illegal use of technology or to disable the access for those resources by others [48, 23].

IoT systems present several vulnerabilities that make the sensitive information prone to the access of cybercriminals. These vulnerabilities can be related to the Internet connection, making the system susceptible to remote attacks, either by direct access to networked control interfaces

or downloading malware to devices. Another vulnerability comes from the edge devices used in IoT systems that have lower computational power and storage capacity than a fog or cloud server, which prevents the implementation of complex security algorithms. Moreover, the devices' heterogeneity also makes the system more vulnerable since they use a large variety of protocols and operating systems without a standardized regulation, generally because they come from different manufacturers [4].

To combat the vulnerabilities found in IoT systems, it is necessary to hide the personal information as well as the ability to control what happens with it. To ensure privacy and security, IoT systems need to obey the following requirements [4, 42, 48, 77]:

Data authentication: Verify if data was not modified and is sent by the claimed author;

Access control: Only allow suitably authorized users to access data, communications infrastructure, and computing resources, and ensure that the access is not denied to those authorized users;

Confidentiality: Keep data private so that only authorized users can access it. Cryptography is a crucial technology for achieving confidentiality;

Location privacy: Prevent leakage of nodes' location by ensuring it is impossible to track back to the entity that generated the information;

Resilience to attacks: The system needs to be secured from some common known attacks (*e.g.*, DDoS attacks, side-channel attacks, malware injection attacks, and authentication, authorization attacks). Moreover, it has to avoid single points of failure and should adjust itself to node failures.

Privacy issues in IoT are not limited to consumers but may also impact the industry. Industrial IoT is more complex than traditional IoT systems since, in addition to the risk of violating sensitive employee or customer details, the potential loss of intellectual data increases the possibility of competitors copying the victim's organization's knowledge and capabilities, which may destroy competitive advantage [53].

The initial focus on protecting information was based on ensuring the system's reliability due to expensive hardware costs and computers' rarity. With the increase of technology, the focus shifted from protecting computers to protect the information, which brought more importance to the notion of confidentiality, integrity, and availability, thus appearing the term CIA Triad [62].

The CIA Triad's roots are deeply embedded in the military security ideology, which has always focused on protecting information from external threats. The following three categories that address potential security risks, also represented in Figure 2.3 (p. 11), become the foundation of the CIA Triad [76, 62]:

Confidentiality: relates to protecting data from unauthorized access that may want to take advantage of information stored in the computer;

Integrity: concerns the protection of data validity against undesired modifications or destruction (*e.g.*, sabotage);

Availability: refers to the timely and reliable access of information — and systems — to authorized individuals and processes in the form and format needed.

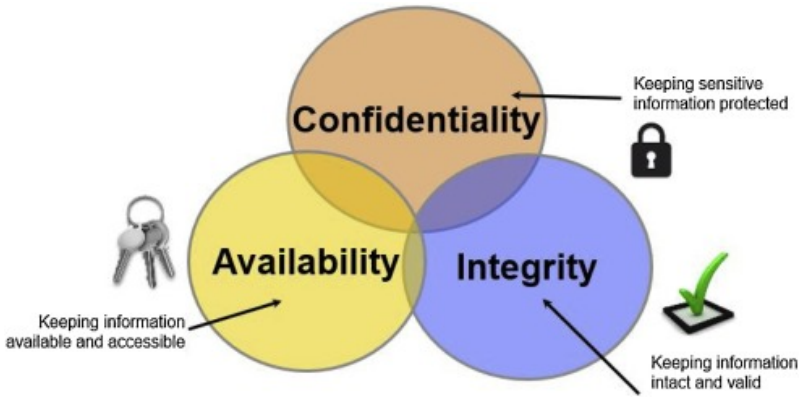

Figure 2.3: The CIA Triad [76].

The CIA Triad represents the basis for privacy rules and the protection of systems that deal with privacy-sensitive data. By analyzing the confidentiality, integrity, and availability of information or information systems, it is possible to perform a classification and qualitative assessment of the security risks and the development of relevant security controls.

The CIA Triad has been criticized for having a particular point of view — data is the center point around how security efforts should be structured — narrowing technical orientation and focus. However, the term is still valuable because it provides a straightforward way to understand and resolve information security issues. The term should never be used in isolation but paired with other security models (*e.g.*, five pillars of information assurance, *Parkerian Hexad* [57]) to achieve defense in depth [62].

2.3 Summary

This chapter introduces concepts regarding IoT, IIoT, Industry 4.0, AAL and privacy and security concerns in IoT systems. Section 2.1 (p. 5) defines Internet-of-Things, as well as each one of its tiers — Cloud, Fog, Edge — explaining in more detail the paradigms of Fog Computing and Edge Computing. Moreover, Section 2.1.2 (p. 8) briefly describes the concepts of both Industrial IoT and Industry 4.0, highlighting the key differences between them. In Section 2.1.3 (p. 8) the purpose of AAL systems are explained as well as how they make use of IoT technology to get real-time information of elderly people. Section 2.2 (p. 9) explains the principal vulnerabilities IoT systems face nowadays, the requirements that should be followed to ensure basic privacy and security protection against threats that may attempt to access or modify data and disable the

system's expected performance, and explains the model CIA Triad used to validate if systems comply with privacy and security rules established by it.

Chapter 3

State of the Art

This chapter describes the state of the art for security and privacy in the Internet-of-Things, as well as systems that orchestrate tasks among devices or servers in distributed architectures. In Section 3.1, the methodology behind the research done is explained. Section 3.2 (p. 14) presents the results for solutions that orchestrate distributed systems, whereas in Section 3.3 (p. 19), we present privacy and security solutions for IoT systems.

3.1 Introduction

To perform a literature review for the current state of the art, we used an iterative methodology (*cf.* Section 3.1.3, p. 14). It consisted of gathering some keywords and define a group of search queries related to each literature research question (*cf.* Section 3.1.1). These queries were then adapted and used to several databases (*cf.* Section 3.1.2, p. 14). The results obtained were then sorted and filtered using some criteria such as the similarity with the domain of this thesis, number of citations and the publish date.

3.1.1 Research Questions

To better understand the work already developed in the scope of this project and to guide the literature review process, we defined the following literature research questions (LRQs):

LRQ1 *What solutions exist that allow the orchestration of devices in distributed IoT system?*

LRQ2 *What techniques have been used to address privacy and security concerns over data that can be applied to orchestration of IoT systems?*

3.1.2 Databases

The solutions analyzed during this research were retrieved from the following databases: (1) IEEE, (2) ACM Digital Library, and (3) Scopus. The material used in the researched literature consists of conference papers, journal and survey articles, as well as popular reference books.

3.1.3 Search Process

To find the analyzed literature, a bottom-up approach, described below, was followed. This approach consisted of executing the following steps in a iterative way:

1. Find keywords related to each research question and generate a query;
2. Use the query to search in the databases, redefining them based on the results when needed;
3. Sort and filter based on number of citations, publication date, and similarity to the domain;
4. Choose relevant results based on title, abstract, and conclusions;
5. Analyze the chosen results with more detail, filtering the ones that are not relevant for this work. From the remaining papers, inspect for important references they may have (*snowballing*);
6. Repeat the process if new terms and concepts were identified by improving the queries used.

3.2 Orchestration of Distributed Systems

Silva *et al*. [68] performed a survey that tries to summarize the state of art in this topic. From the survey results we have analyzed the following tools:

Giang *et al*. [36] propose a Distributed Dataflow (DDF) programming model for the IoT systems that makes use of the computing infrastructures across the Fog and the Cloud. The authors' motivation behind their solution came from the possibility of leveraging the computational capabilities of edge devices to filter, aggregate, and analyze data collected instead of pushing raw data directly to the Cloud which, consequently, lowers the communication cost, storage needs and overall responsiveness of the system. They evaluated their approach by implementing a DDF framework based on Node-RED[1], an open-source flow based run time and visual programming tool for building IoT applications. In their DFF model, the flow is deployed on multiple physical devices rather than just one. Each one of the physical devices may be responsible for executing one or more nodes in the flow, so mechanisms that allow the communication between nodes on different devices are required.

Giang *et al*. implementation consist of D-NR processes running on several devices in local networks and servers. The developers use one of the processes as a development server

[1] https://nodered.org

where the flow is designed by using a dataflow editor. The devices are all subscribed to an MQTT topic, which represents the status of the flow. Whenever an update to the flow occurs, all the participating devices and servers are notified so they can update and parse a new version of the flow and posteriorly decide which nodes should be deployed locally and which ones are to be replaced. When making these decisions, the participators take into account some constraints, namely (1) computation resources, (2) network bandwidth, (3) available storage, and (4) user-defined properties.

Even though their DDF framework provides an alternative approach for designing and developing distributed IoT systems, there are some open issues, such as the need to include a distributed discovery and communications infrastructure between devices and networks.

Blackstock *et al.* [11] proposed a cloud-based platform called named WoT Flow. The system makes use of the open source Node-RED system to provide an execution engine suitable for both multi-user cloud environments and individual devices. By taking advantage of data flow, the authors aim to move the computation between processors and devices so that parts of a data flow can be executed in parallel on different devices. The split and distribution of the data flow makes possible the execution of the operations in the cloud or edge devices, depending on user preferences (*e.g.*, cost, speed of communications, host performance). At the time of the writing, the WoT Flow was in the early stages of development. The future steps were about supporting the automatic partitioning and distribution of data flows based on participating resource capabilities and constraints imposed by the developer around cost, performance and security by using optimization heuristics.

Noor *et al.* [55] introduced DDFlow, a macro programming abstraction that provides an efficient means to build high quality distributed application on an IoT network. The motivation behind the DDFlow system was to provide a framework that would abstract the developer from low-level network, hardware, and coordination details between the devices in the IoT network. To cope with the authors' motivation, DDFlow allows the developers to state high-level objectives by using a system runtime that translates them into a dataflow graph that is dynamically deployed to the devices in the IoT network. The system runtime is based on Node-RED and, by enabling dynamic scaling and adaptation, leads to improved end-to-end latency preserving the system's behavior despite device failures.

The DDFlow system architecture can be seen in Figure 3.1 (p. 16). The intra-device coordination is done through a lightweight web server that runs on every device in the network named *Device Manager*. In contrast, the inter-device coordination is accomplished through a *Coordinator*, a web server that accepts and manages DDFlow apps. The *Coordinator* is responsible for mapping services to available devices in the network while minimizing end-to-end latency and dynamic adapt and recover the system functionality when network changes or device failures occur.

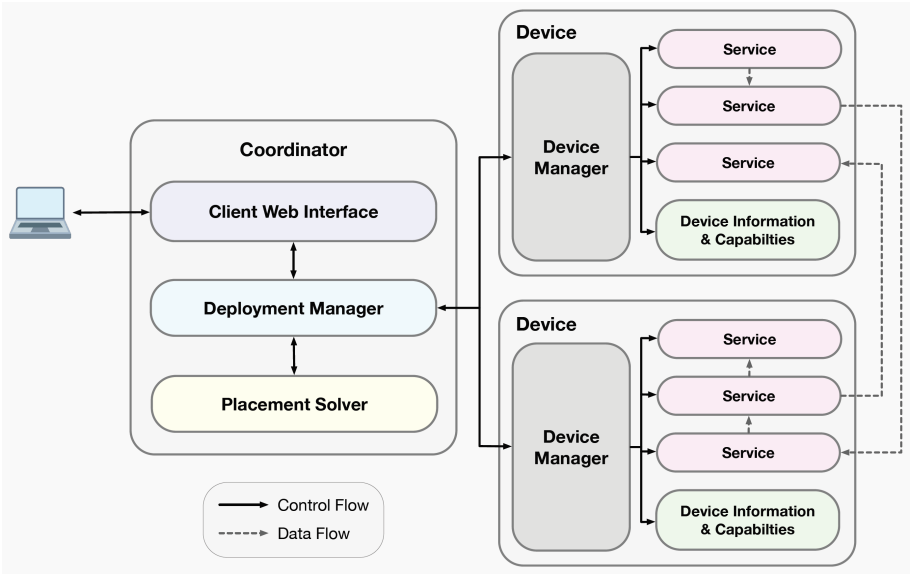

Figure 3.1: DDFlow system architecture [55].

Additionally, we performed a search on Google search engine to find post-survey or non-survey works related to this topic described in the following paragraphs. The keywords used for the search were one or a combination of the following: `Internet-of-Things`, `orchestration`, `distributed`, `decentralized` and `iot`.

Cui *et al.* [18] propose a decentralized and trusted platform for edge computing (DeTEC) for the IoT where the users can submit their computational tasks to the system. These tasks are allocated to the most appropriate edge server by the DeTEC controller using a heuristic algorithm that takes into consideration the propagation latency, node capacity, and reward fairness of the system. The system integrates blockchain technologies to organize the distributed resources and provide incentives for the distributed edge servers' computational contributions.

As seen in Figure 3.2 (p. 17), the DeTEC platform consists of four main components: *IoT user*, *Edge servers*, *Controller*, *DNS server*. The *IoT users* can apply for the DeTEC service to perform computational tasks by calling a remote interface and receiving the results from the edge servers. Each *Edge server* computes the assigned tasks, producing results that are verified and the contributions recorded in the blockchain. The *Controller* is responsible for collecting the network state of the users and edge servers and compute the task allocation scheme. Finally, the *DNS server* resolves the requests from *IoT users* and binds the target *Edge server* according to the allocation scheme computed by the *Controller*.

In order to guarantee the trustworthiness of the system, Cui et al. designed a police patrol model that is responsible for verifying, periodically and randomly, the computational results submitted by edge servers. When the results are incorrect, they are rejected, and the corresponding edge servers are punished. The police patrol model can be centralized or

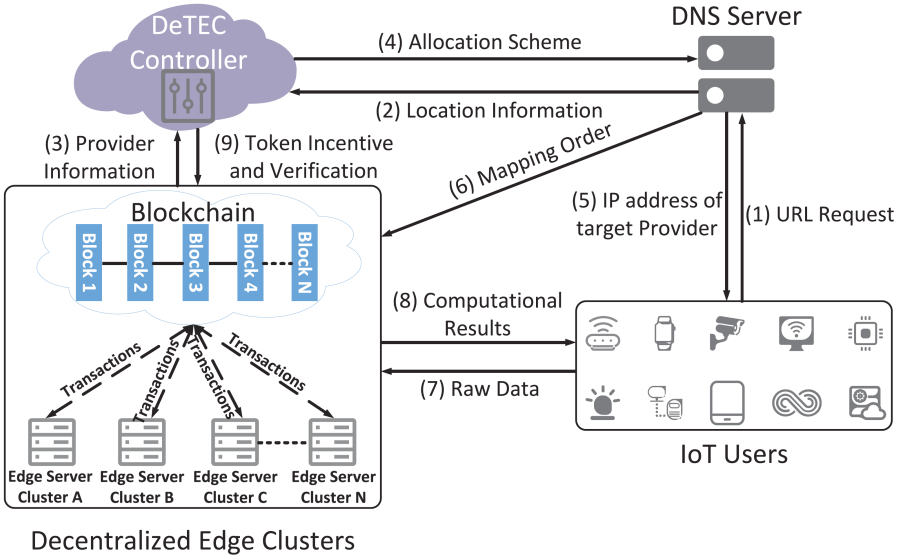

Figure 3.2: DeTEC solution architecture [18].

decentralized. In the case of a centralized approach, the DeTEC controller verifies the computation result of edge servers. On the other hand, in a decentralized approach, the DeTEC system elects a supervisory consortium, and the consortium members are responsible for verifying the results.

The authors validated the system by focusing more on analyzing the latency in task allocation and resource scheduling performance. However, they did not validate if the privacy of the system data was preserved.

NoRDOr Silva *et al.* [19] propose a method capable of automatically orchestrate constrained devices in a distributed IoT network by exploring their computation capabilities. The authors aim to solve the consequences that arise from centralized approaches by developing a prototype that is divided in two parts: (a) extended Node-RED, a visual programming language (VPL), to automatically distribute computational tasks among the available devices, and (b) developed a MicroPython framework that runs custom code assigned by the orchestrator. Visual programming languages allow experienced users to perform rapid application development and non-technical users to extend their application or create their own applications without requiring a lot of programming knowledge [13].

In Figure 3.3 (p. 18), the authors present an overview of the designed system. The devices in the network announce their address and capabilities to a *Registry node* from the Node-RED which, consequently, stores a list of the available devices. The *Orchestrator* has access to the list and assigns, using HTTP, computation tasks to the available devices. Node-RED is centralized by design, so the authors performed some changes to implement a distributed architecture: MQTT was used as the communication protocol to allow the deployment of

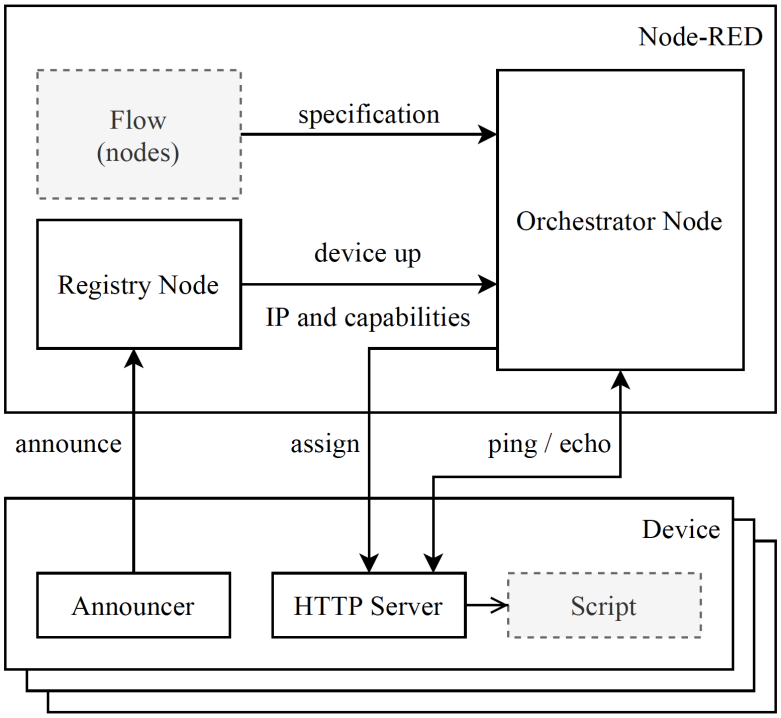

Figure 3.3: NoRDOr solution's overview [19].

nodes externally, and some nodes were modified to convert JavaScript to MicroPython-compatible code, which runs on edge devices. For these edge devices, Silva *et al.*developed a custom firmware that contains packages for a HTTP server and MQTT communication, which are used to communicate the current state of the device (*e.g.*, running, out-of-memory, not running tasks) to the *Orchestrator* and receive custom scripts to be executed. Since the *Orchestrator* knows the state of the orchestrabable nodes, it can re-orchestrate the tasks in the case of a failed deployment to keep the flow of the system running.

The mentioned tools were analyzed based on their approach to the following categories:

- **Scope:** The authors, when developing these tools, had specific use cases in mind. Therefore, it is important to know each solution's scope to understand the gap in what we aim to develop. Example values are *Several*, *Education*, *Industry*, *Home Automation*.
- **Approach:** The solutions present different ways to build the system, for example some can use frameworks or provide a graphical interface. Possible values are: *VPL*, *Framework* or *Methodology*.
- **Computational location:** Each solution distributes computational tasks, however, they differ on the layer where the tasks are executed. Possible values are: *Cloud*, *Fog* and/or *Edge*.
- **Privacy/Security:** Due to the context of this dissertation, it is important to understand if the tools address, in some way, security and privacy concerns over the data or the system.

Table 3.1: Orchestration of Distributed IoT Systems Analysis

Solution	Scope	Approach	Computational location	Privacy/Security
Giang *et al.* [36]	Several	VPL	Fog and Edge	-
Blackstock *et al.* [11]	Several	VPL	Cloud, Fog and Edge	Future work
DDFlow [55]	Security	VPL	Fog and Edge	-
Cui *et al.* [18]	Healthcare	Methodology	Fog	-
NoRDOr [19]	Home Automation	VPL	Edge	Out of scope

VPL: Visual Programming Language

From the analysis and the characteristics of Table 3.1, we can conclude that there are several approaches to orchestrate computation tasks in different layers of a distributed system. Even though the solutions presented do not have the same scope, they can be adapted to the scope we wish. It is also possible to conclude that none of these solutions implemented methods to mitigate potential security and privacy issues they may have. Some even refer that secure data privacy is a future work [11] or a secondary goal for their implementation [19].

The fact that these issues have been ignored raises some possible problems. In the case of DDF [36] the orchestrator is responsible to notify the participating devices and servers whenever a new version of the flow is available. This can bring a privacy problem because an untrustworthy orchestrator can send a malicious update to the network, which is consequently deployed to all the devices and servers. This malicious update may compromise the visibility of the data being treated on those devices and servers, thus compromising the whole orchestrated system and, consequently, data privacy. In the case of Silva *et al.* [19] prototype, we can see a possible privacy problem when an orchestrator sends a custom code to run directly in the edge devices. Similarly, this can leverage unwanted access and compromise sensitive data that may flow through those devices. Like these problems, other data privacy and security problems can be found in the other researched systems so it is important to urge solutions for these concerns.

3.3 Data Privacy & Security in IoT Systems

In the previously analyzed systems there are no implemented mechanisms to mitigate potential security and privacy issues they may have. Thus we performed an extended literature review where we intend to find techniques that can be used to address privacy and security concerns over data that can be applied to orchestration of IoT systems. The works have been selected after a curated search on Google search engine, and are shortly described in the following paragraphs. The keywords used for the search were one or a combination of the following: `Internet-of-Things`, `privacy`, `security` and `iot`.

Skarmeta *et al.* [70] propose a distributed capability-based access control mechanism built on public key cryptography fitting the requirements of IoT regarding scalability and interoperability. To cope with privacy and security challenges in IoT, the authors use technologies

that facilitate a distributed approach in which IoT devices themselves are capable of making fine-grained and context-aware authorization decisions. They developed an optimized version of the Elliptic Curve Digital Signature Algorithm (ECDSA), which is implemented within the IoT device providing end-to-end authentication, integrity, and non-repudiation.

Li *et al.* [47] propose an ECC- and biometric-based user authentication protocol scheme with privacy preserving for IIoT. It aims to guarantee that only valid users can access the sensitive data collected by sensors on an IoT network. The authors' protocol is divided into three essential phases: (a) Registration Phase, (b) Authentication and Key Agreement Phase, and (c) Password Change Phase. For each phase, Li *et al*. detail by steps the procedures that need to be followed in this protocol. They prove the security of the protocol by giving a formal proof of it under a random oracle model and show that the proposed scheme can resist well-known attacks (*e.g*., Anonymity and Untraceability, Resist Replay Attack, ...). Even though the developers simulated the protocol using the NS-3 network simulator with satisfactory results, it is still missing the protocol's validation in a real case scenario.

Dwivedi *et al.* [31] propose a decentralized privacy-preserving blockchain for a healthcare IoT system. The proposed model's motivation comes from the necessity to secure the data collected from patients' wearable devices and consulted by a Health Service. The authors decided to use a decentralized overlay network to ensure the system's scalability and eliminate a single point of failure and delay problems with the data. Moreover, they use a lightweight digital signature scheme to authenticate the data transferred throughout the system, preventing the information from being modified, and a lightweight ring signature to ensure anonymous transactions by authentic users. To prevent hacker attacks, the system uses a double encryption scheme, encrypts the data using a lightweight ARX algorithm, and afterward encrypts the symmetric key used to encrypt the data by using a public key. The developers also use the Diffie–Hellman key exchange technique for securely exchange cryptographic keys over a public channel. By using these lightweight techniques together, the model can guarantee security, privacy, and anonymity of user's data even on small IoT devices.

The system is divided in five parts: *Cloud storage*, *Overlay network*, *Healthcare providers*, *Smart contracts* and *Patient with wearable IoT devices*. Instead of storing the patient's healthcare data over the blockchain, the authors decided to use a *Cloud storage* server. The *Overlay network* is divided into several clusters and it is where the patient digital signature and public key are verified whenever a signed transaction is sent to the network. The *Healthcare providers* are entities that receive alerts generated by *Smart contracts* and are authorized to receive detailed patient data from the cloud. Finally, *Patients with wearable IoT devices* is the part that collects all the data that flows in the system. A summarized logical flow execution of the system can be seen in Figure 3.4 (p. 21).

The authors conclude by evaluating the security margins of the model, proving theoretically that the system can protect itself against the following threats: (a) Denial of Service (DoS)

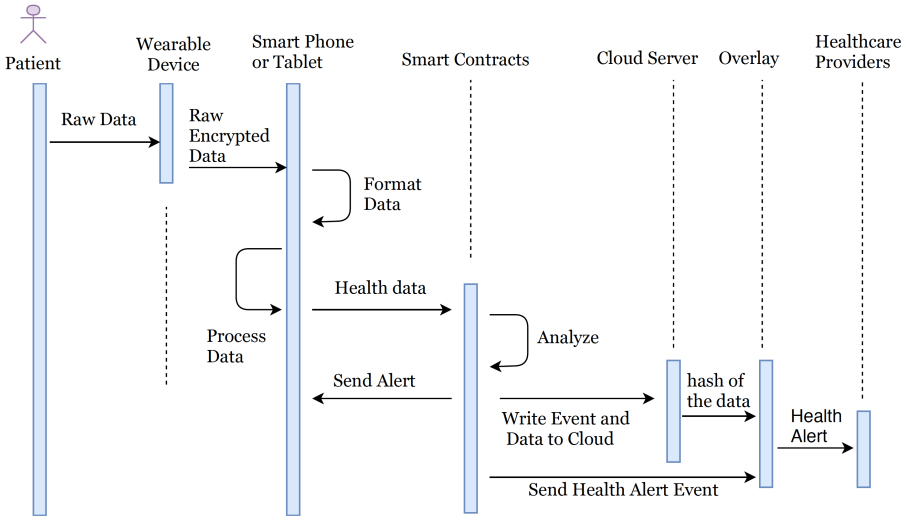

Figure 3.4: Dwivedi *et al.* logical flow execution of the system. Extracted from [31].

Attack, (b) Mining Attack, (c) Storage Attack, and (d) Dropping Attack. However, this work does not guarantee the security in all its components from other potential threats since it was not tested in a real case scenario.

Xing *et al.* [78] propose a mutual privacy preserving k-means clustering scheme for a social participatory sensing application that protects participants' private data and the community data (*e.g.*, patterns, distribution, etc.). By doing this, the authors want to make possible to extract and analyze data (data analyst) about the community without getting access to any user's private information. They also intend to prevent any participant from accessing data from another participant and the community itself.

Two privacy-preserving algorithms compose the scheme: (1) the first one is utilized by each participant to find the nearest cluster, using the Euclidean distance, while the cluster centers are kept secret to them; (2) the second one keeps the cluster centers up to date without leaking any information to the participants by making use of an additive homomorphic encryption scheme. The authors validate the scheme's security against collusion attacks that can happen between the data analyst and participants or among the participants. For both cases, they concluded that even when collusions happen, no private information of the participants is leaked. As future work, the developers want to explore the mutual privacy protection of other clustering algorithms.

Pape and Rannenberg [56] performed an investigation to improve users' privacy by mapping several privacy patterns to IoT, Fog computing, and Cloud computing architectures. According to Ni *et al.* [54], privacy in IoT is mostly relevant in the next four types of information: (1) identity (*e.g.*, name, address, etc.), (2) sensitive data (*e.g.*, health status, etc.), (3) information usage (*e.g.*, usage pattern of a service, etc.), and (4) location data (*e.g.*, current

location, trajectory, etc.). Pape and Rannenberg explain how can privacy be guaranteed in this types of data by using the following privacy patterns:

- **Personal data store**: users have total control over their personal data, and it is stored on their personal devices. Ideally, necessary computations are performed locally in the IoT device, however, in the case of insufficient computational power, the computations can be done in the user's mobile phone. If the data needs to be stored in the cloud then the IoT device should encrypt the data and save the decryption key on the mobile phone.
- **Data isolation at different entities**: data is distributed among several entities (*e.g.*, fog nodes, clusters) where each entity can only see a part of the data.
- **Decoupling content and location information visibility**: fog nodes can alert the user if he is transmitting location information or remove it.
- **Added noise measurement obfuscation**: add noise to measurements whenever a user uses a resource repeatedly over time, depending on the type of information used. The noise can be added by the fog nodes or the user's mobile phone.
- **Aggregation of data**: aggregate the usage information of multiple users or the usage information of a single user over time. Similarly to adding noise, the aggregation can be done by a trustworthy provider or by the users themselves.
- **Aggregation gateway**: this pattern is used when the service provider needs a continuous measurement, and adding noise is not acceptable. An homomorphic encryption is used to encrypt each measurement in the IoT device or user's mobile phone, and then they are transmitted to the cloud computing provider. The cloud can operate on the data received (*e.g.*, aggregate), even though they cannot access the data in clear, and then send it to the fog node. Since the key used for the encryption is shared between the IoT device and fog node, the fog node can have clear access to the result without having individual information of the users or devices. The flow of this pattern can be seen in Figure 3.5 (p. 23).
- **Single Point of Contact**: a cloud computing service manages and coordinates a distributed storage on different fog nodes by issuing security tokens, authenticate local domain users as an Identity Service Provider, certify attributes as an Attribute Provider, and accept external claims as a Relying Party.

For each of these privacy patterns, the authors demonstrated that they could be applied in real-world scenarios. However, they raise a question about whether it is more secure to store data in the IoT nodes or at a central database of the cloud.

The mentioned approaches in IoT systems were analyzed based on the following categories:

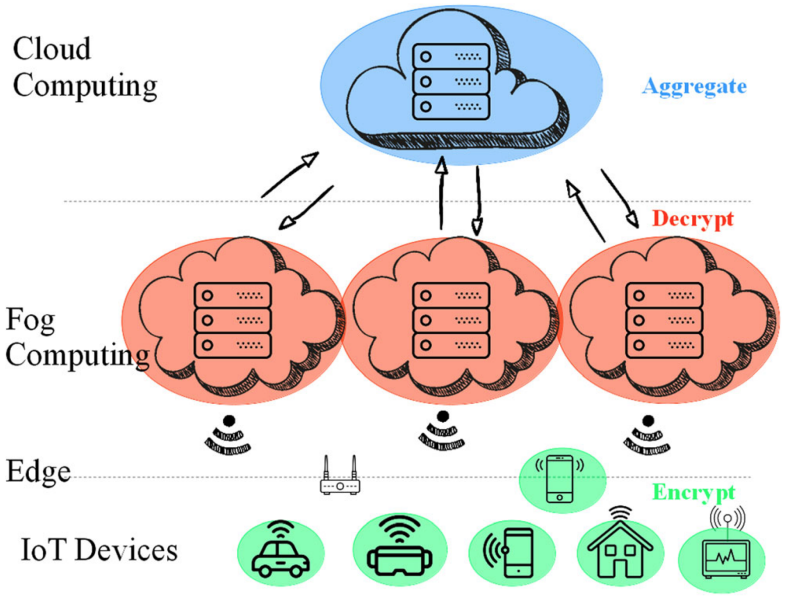

Figure 3.5: Aggregation gateway pattern example [56].

- **Privacy Category:** The approaches differ on which data privacy issues they attempt to solve. By categorizing the methods, it is possible to better understand the focus of each solution. The possible values for data privacy are:
  - *Data perturbation:* distortion of dataset values while keeping its basic statistical distribution properties by introducing additive or multiplicative noise directly into original raw data.
  - *Data encryption:* cryptography-based privacy preserving methods that hide sensitive data.
  - *Publishing restriction:* enables secure data release without revealing sensitive information by converting original data into an anonymous dataset.
- **Architecture:** Some of the analyzed approaches are applied to different IoT systems architectures. Possible values are *centralized*, *decentralized* or *distributed* architectures.
- **Essence of the Solution:** This category exposes in few words the solution each author used to address privacy and security issues in IoT systems.

From the analysis in Table 3.2 (p. 24), we can conclude that solutions that address some widespread concerns about protecting privacy-sensitive data already exist.

Mostly by using encryption algorithms, the authors can hide data from layers of their networks that are more susceptible to attacks [70, 47, 31, 78]. There are also solutions based on biometric authentication [47], data signature [31], and homomorphic encryption [78] that guarantees the anonymity of the user to whom the sensitive data belongs when an entity of the system needs to have access to the data unchanged/unencrypted. It is also noticeable a trend towards the use of

Table 3.2: Privacy & Security in IoT Systems Analysis

Reference	Privacy Category	Architecture	Essence of Solution
Skarmeta *et al*. [70]	Data encryption	Distributed	An optimized version of the Elliptic Curve Digital Signature Algorithm (ECDSA)
Li *et al*. [47]	Data encryption and Publishing restriction	-	An ECC and biometric-based user authentication protocol
Dwivedi *et al*. [31]	Data encryption and Publishing restriction	Decentralized	A lightweight digital signature scheme to authenticate the data and a lightweight ring signature to ensure anonymous transactions
Xing *et al*. [78]	Data encryption and Publishing restriction	-	An homomorphic encryption scheme
Pape and Rannenberg [56]	Data perturbation, Data encryption and Publishing restriction	-	Several solutions [1]

[1] The authors present several solutions by mapping privacy patterns to IoT.

blockchains to guarantee that data being interchanged is authentic and unchanged (*e.g.*, preventing data fabrication and falsification) in IoT [58], concept which was already well-explored in domains such as AAL [29, 24].

In the context of privacy and security, we can also verify how several patterns [56] address these problems in other situations and can also be applied to IoT, making the systems more secure to potential threats.

The implementation of most of these techniques is not trivial and straightforward in some of the IoT systems analyzed before. Most of the *edge* devices have low capabilities and energy restriction, thus they cannot run heavy algorithms to ensure the privacy of data. However, by applying some privacy patterns it is possible to develop a system that orchestrate tasks to devices in IoT networks without compromising the privacy of data flowing in the system.

3.4 Summary

Section 3.1 (p. 13) presents the methodology followed, the key research questions, and the databases used to perform the literature review.

Section 3.2 (p. 14) presents solutions found for the orchestration of computational tasks for distributed IoT systems. For each one of the solutions found they are analyzed and summarized the scope, their approach, in which layer of the system computation tasks are executed and whether or not they address security and privacy problems that may urge in those systems. With the analysis done, we can conclude most of the solutions discard protection methods to the data that flows in the systems and so are prone to attacks that may want to steal the privacy-sensitive data.

Section 3.3 (p. 19) introduces several approaches found that guarantee the anonymity and confidentiality of data and device authentication in the IoT network. Each of the techniques is classified in terms of privacy category, the architecture of the solution, and the solution's essence.

By analyzing the results in Section 3.2 (p. 14) and Section 3.3 (p. 19) it is possible to find a clear gap when it comes to the privacy of data during the orchestrations of distributed IoT systems,

as there are no solutions that perform the orchestration guaranteeing that the information that flows in the system is kept private even though there already exist several ways of doing so in general IoT systems.

Chapter 4

Problem Statement

This chapter describes the problem we aim to tackle in this dissertation, research questions and our experimental methodology. Section 4.1 presents some assumptions done for this project. In Section 4.2 (p. 28) is described the limitations found in the performed literature review. Section 4.3 (p. 29) presents a set of propositions for the system to be implemented. Section 4.4 (p. 29) defines the scope of this work and Section 4.5 (p. 29) describes the research questions to be explored during the development and posteriorly analyzed with the system evaluation. Finally, Section 4.6 (p. 30) outlines the experimental methodology.

4.1 Assumptions

Despite the several concerns over privacy in IoT systems, this project will not try to answer the privacy needs of all the IoT systems. Instead, it will focus on distributed IoT systems with real-time orchestration that use VPLs, specifically the NoRDOr system [19]. The NoRDOr system is comprised of a modified version of Node-RED enhanced with the ability to dynamically allocate computational tasks to edge devices, which can execute custom code due to the custom MicroPython firmware developed. The reasons behind choosing this system as a starting point comes from the fact (1) Node-RED is a tool often used by system integrators [22] in the IoT domain, which provides a flow-based way of connecting together a plethora of devices and online services without requiring programming knowledges, (2) the system is capable of assigning Node-RED *nodes*

to be executed in edge devices, (3) it is open-source and (4) due to the closer connection with those responsible for the development of the system which allows to easily clarify any doubts that may arise. The other analyzed systems were not preferred due to the following reasons: (1) lack on an automatic discovery and communications infrastructure between devices and networks [36], (2) are cloud-based platforms [11], (3) integrate blockchain technologies [18], and (4) are not open-source [55].

Throughout this dissertation, the concept of privacy-sensitive data is used several times. Private data can be data of any type, such as identity, health data, information usage, location data. Thus it is up to the users to define what they think is essential to be protected.

4.2 Open problems

Section 3.2 (p. 14) presents several solutions that orchestrate computational processes in distributed IoT systems. However, these approaches do not use methods that mitigate potential privacy issues over the data that flows in the system. Particularly, in the NoRDOr [19] system, there is no concern to ensure that the data communicated between the edge and fog devices stays private throughout the whole system. Moreover, there is no mechanism to certify if the announced devices are secure to handle possible privacy-sensitive data. With this being said, it is possible to outline the following limitations in this system that should be addressed:

- **Unprotected data:** when defining the *flow* in the Node-RED there is no way to identify *nodes* that can collect privacy-sensitive data. The orchestrator should be aware of the presence of *nodes* that can introduce private data to the flow so it can make an informed and better choice when assigning the *nodes* to the available devices. Like this, the system could be able to protect the data whenever it needs to be sent to risky locations.
- **Data identification:** there are some types of private data (*e.g.*, health data, location, identity) that, when being published to communication channels that other devices may access, is vital to prevent its traceability to any user of the network. Thus, the system should be able to hide the identity of the data flowing.
- **Unauthenticated devices:** devices in the network announce themselves by communicating their address and capabilities. The only concern the orchestrator has with the announced devices is their status — if they are running or if they have failed — and the respective capabilities to execute Node-RED *nodes*. There is no preoccupation in ensuring the fidelity of devices. Therefore, it is necessary that each device is properly authenticated to avoid the system being compromised.

The actual orchestration strategy used in the NoRDOr system does not have enough information to guarantee the privacy of data that flows in the system. The fact that the orchestrator only knows the capabilities of devices and the priorities of nodes makes it unable to find out if devices are safe to handle private data and understand which nodes will potentially generate

or process privacy-sensitive data. Thus, in the current system, the orchestrator cannot make a privacy-oriented choice when assigning nodes to the available devices in the IoT network.

4.3 Desiderata

The purpose of this dissertation is to develop a system that addresses the limitations from the NoRDOr system explained above. This system should fulfill the following desiderata:

- **D1: Automatically protect privacy-sensitive data**, so that no private data reaches possible unsafe locations without first going through some transformation that ensures the data stays private.
- **D2: Generate privacy-oriented *node* assignment**, so that the orchestration can prioritize the execution of nodes that handle privacy-sensitive data in safer locations.
- **D3: Validate devices' identity on the network**, so that no unauthorized devices can announce themselves as safe devices and consequently access or produce data they were not supposed to.

Our work will focus more on providing a system that fulfills *desiderata D1* and *D2*. Any development made towards D3 is secondary.

4.4 Scope

The scope of this work is to come up with mechanisms that can facilitate privacy in a distributed IoT system with real-time orchestration. These mechanisms will be developed based on existing platforms that allow the orchestration of computation tasks among edge devices on an IoT network. Thus, the focus of our work is to ensure the privacy of the data exchanged between these devices. The target audience for our platform are developers that are willing to use a visual programming language to explore the computation capabilities of IoT devices in a network where it is essential to preserve the privacy of data.

Despite existing several approaches that deal with privacy and data integrity using blockchain-based systems [24, 58, 29], it is a solution out-of-scope for this work's focus due to natural complexity of such mechanisms and the complexity of the network needed for a better efficiency, which may not be possible with our scenarios.

4.5 Research Questions

The work developed in this dissertation aims to answer the following research question:

> *How can we improve the NoRDOr system to automatically guarantee a safe flow of privacy-sensitive data?*

The main research question can be split into several other research questions derived from the presented *Desiderata* (*cf.* Section 4.3, p. 29):

RQ1: How can we guarantee that privacy-sensitive data without any type of transformation is only processed by trusted devices?

Private data generated or processed by a device should only be sent without any type of transformation (raw data) to devices that can be trusted by it.

RQ2: How can we ensure the privacy of data but at the same time explore the most of the computational power available in the network?

To allow the system to make use of most of the computational capabilities of edge devices in the network, the system should be able to transform privacy-sensitive data so that it remains private even when sent and processed in possible untrusted devices.

RQ3: Can we provide a way to prioritize the allocation of sensitive nodes to safer locations?

If we could prioritize the assignment of privacy-sensitive nodes to trusted devices without compromising the performance of the devices, the risk of private data being handled in unsafer locations could be minimized.

All these decomposed research questions will be used to analyze the experiments (*cf.* Section 6.3, p. 49) and answer the main research question.

4.6 Methodology

To validate if our proposed solution fulfills the desiderata presented in Section 4.3 (p. 29) and provides reasonable answers to the research questions in Section 4.5 (p. 29), we will use two test scenarios (*cf.* Section 6.1, p. 41) and multiple controlled experiments using virtual devices. In each one of the experiments, we will try to explore different system behaviors by measuring several metrics from the simulated devices that are posteriorly going to be analyzed to validate our solution (*cf.* Section 6.3, p. 49).

4.7 Summary

The assumptions performed for the development of this work are described in Section 4.1 (p. 27). The current issues identified from the literature review on the orchestration of distributed IoT systems are introduced in Section 4.2 (p. 28). A set of requirements that our solution aims to fulfill are detailed in Section 4.3 (p. 29), which summarizes the desired improvements to be implemented to ensure the privacy of sensitive data in a distributed IoT system. The scope and focus of this work are to develop mechanisms that can correctly deal with privacy-sensitive data that flows in distributed IoT systems that can be orchestrated, as indicated in Section 4.4 (p. 29). In Section 4.5 (p. 29) we establish several research questions to guide the development of our solution

and which will be answered along with the analysis of the results from the experiments performed. The methodology for these experiments is detailed in Section 4.6 (p. 30). We will simulate physical devices to execute the experiments in multiple scenarios to gather relevant information to validate our work.

Chapter 5

Privacy-oriented Task Orchestration

This chapter introduces and describes in detail our implementation to tackle down the problems presented in Chapter 4 (p. 27). Section 5.1 provides an overview of the whole solution. In Section 5.2 (p. 34) we further explain the changes made to the devices while Section 5.3 (p. 35) presents, in detail, the improvements implemented in the orchestrator to guarantee the privacy in the system. In Section 5.4 (p. 36), we explain how the *node* assignment algorithm prioritization was modified. Finally, Section 5.5 (p. 39) presents some limitations of our solution.

5.1 Overview

In our implementation, we extend the Node-RED version developed by Silva *et al*. [19, 67], named NoRDOr, providing a more privacy-oriented solution over the data that flows in the system. In Node-RED-based solutions, the term *flow* is used to describe a set of connected nodes. Silva *et al*. platform allows the insertion of *flows*, where for each *node* present are generated tasks that are orchestrated to the edge devices in the network. The platform can be divided into two main components (1) Node-RED instance and (2) a MicroPython based firmware that is flashed to the edge devices in the network. The MicroPython firmware allows the devices to run arbitrary Python code scripts that are generated and sent by the Node-RED *orchestrator* using HTTP. In our solution, we mostly introduced improvements in the Node-RED component, leaving the behavior and functionality of the device almost unchanged. We also kept the communication layer implemented in the NoRDOr system, which allowed the distributed computation communication by using an

MQTT-based instead of an event-based one, present in *vanilla* Node-RED, that does not support communication with external entities.

Regarding the devices, we have extended the announcement functionality to also communicate their zone, in addition to their address and capabilities, to the *Registry node*. Devices' zones can be interpreted as virtual barriers that separate trustworthy devices from devices that may be unsafe to receive and handle potential privacy-sensitive data. Upon receiving the devices' information and the *nodes* from the *flow*, the *Orchestrator node* is responsible for assigning each *node* to one of the available devices. When deciding the *nodes*' assignment, the orchestrator takes into consideration the devices' zone and if the *nodes* generate or deal with sensitive data. Whenever sensitive data is assigned to devices in different zones, the orchestrator introduces a transformation *node* in the middle of the communications between those zones, guaranteeing the privacy of the sensitive data. Moreover, the *node* assignment algorithm was also improved to prioritize the allocation of sensitive nodes to safer locations, reducing the number of zones crossed by private data.

An high-level overview of the system can be seen in Figure 5.1.

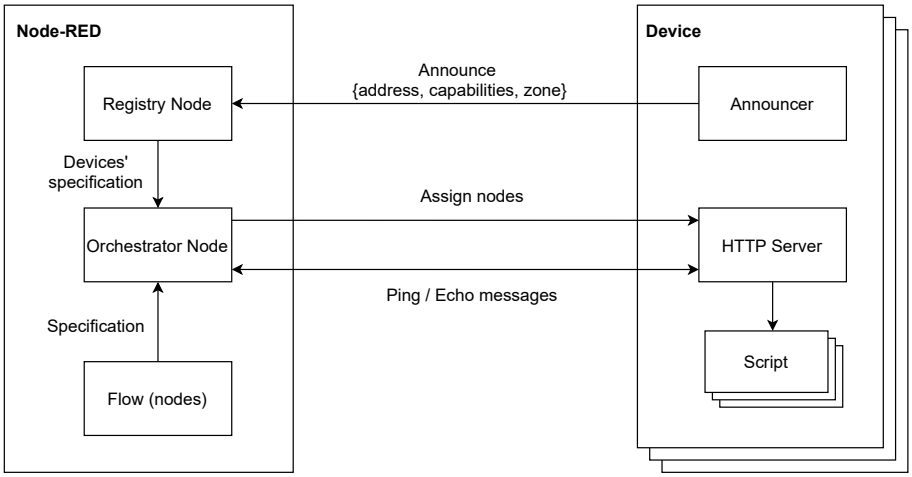

Figure 5.1: High level overview of the system modules. Adapted from [19].

5.2 Devices' Zones

As previously mentioned, in Silva *et al.* solution, the edge devices announce themselves to the *Registry node* by sending their address and capabilities. In our implementation, we extended the devices' announcement to also contain the zone of the device. A device's zone can be interpreted as a virtual barrier that separates devices. Two devices in the same zone are considered to be trusted devices for each other, so data can flow between them without the need to ensure privacy. On the other hand, two devices in different zones are not considered to be safe to each other, and so, whenever data flows from one device to another, we must ensure the data is protected before reaching the other device.

With the definition of zones, we are setting privacy boundaries when orchestrating tasks to the devices. The Node-RED component is aware of each device's zone and capabilities when choosing which devices the *nodes* will be assigned to. Thus, the orchestrator can make a more privacy-oriented *node* assignment choice and introduce transformation *nodes* when needed.

In Silva *et al.* 's work, a custom firmware was used for the devices where one of the main modules was an HTTP Server responsible for the endpoint that allows the monitoring of the device status and the reception of the assigned MicroPython script. In our solution, we had the need to also implement an endpoint for the HTTP server that triggered a delete action inside the device, removing all the custom scripts from the assigned *nodes* it had.

5.3 Node-RED Computation Orchestration

In order to achieve a distributed IoT system with real-time orchestration capable of ensuring that privacy is kept when generating, processing, and exchanging privacy-sensitive data, the orchestrator needs to take action when the data is sent from one device to another in two different zones. Whenever this happens, the system needs to process and transform the data before it leaves the zone where it was generated.

As seen in Section 3.3 (p. 19), several privacy-preserving solutions have been presented, making use of data encryption algorithms [70, 78], signature schemes to authenticate data and protect the anonymity of users [31, 47], and some privacy patterns used in IoT [56]. However, these algorithms and signature schemes presented are not trivial to implement. Therefore, to simplify the development of our solution, we followed some privacy patterns commonly used in IoT.

Firstly, when developing the *flow*, we introduced the possibility to select if the output of a *node* is privacy-sensitive data — for developing purposes, this option was implemented to the *temperature-humidity node* — signaling the data that needs to be protected by the orchestrator. This functionality can be seen in Figure 5.2 (p. 36). Once the orchestrator receives the *nodes*' specification, it goes through the *flow* as a graph and marks the *nodes* where privacy-sensitive data will pass through. To check when privacy-sensitive data is going to be processed in a *node*, the orchestrator verifies if there is at least one MQTT input topic of the *node* that matches the MQTT output topic of an already marked *node*. Once the *nodes* are marked, the orchestrator proceeds to find the best devices for each one of the *nodes*.

After finding the best *node* assignment (*cf.* Section 5.4, p. 36), the orchestrator moves on to find where privacy-sensitive data is flowing between different zones. Whenever it finds two *nodes* that communicate between each other assigned to devices that are not in the same zone, the orchestrator proceeds to transform the data before it leaves a zone. The data transformation could be, for example, encrypting, aggregating, or anonymizing the data. As a proof-of-concept, we introduced to the Node-RED *Palette* an Aggregation *node*, which can aggregate a specified number of messages or the messages received during a specified time. The default configuration used by the orchestrator when inserting aggregation *nodes* is to group messages for 20 seconds or until it receives ten messages and then send the aggregated messages as an array. When the orchestrator

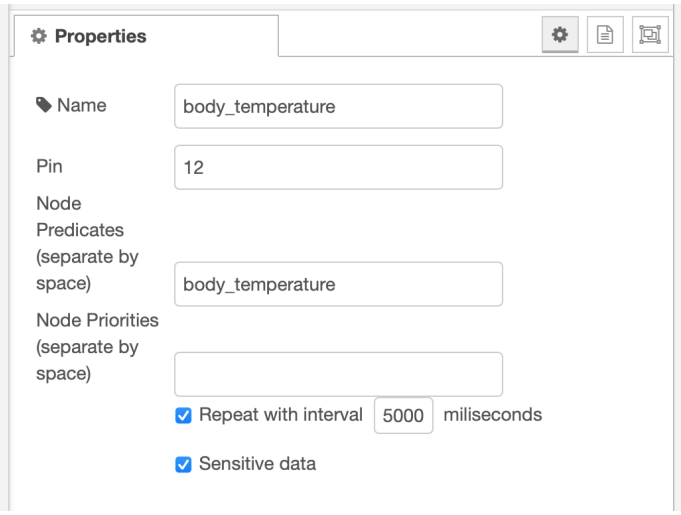

Figure 5.2: Node-RED *node* properties example.

needs to transform data to ensure its privacy, it generates and inserts an aggregation *node* in the middle of the communication of the two *nodes*, changing the respective MQTT input and output topics. The default configuration of the Aggregation *node* is to group messages for 20 seconds or until it receives ten messages and then sends the aggregated messages as an array. Posteriorly, the orchestrator finds the best possible devices for the inserted transformation *nodes* by using a similar algorithm used by Silva *et al.* . Only after ensuring that all the needed transformation *nodes* were introduced to the flow, the orchestrator generates the *nodes* python scripts and sends them to the respective devices.

A visual representation of these events can be seen in Figure 5.3 (p. 37).

5.4 *Node* Assignment Algorithm

Originally, in the NoRDOr system, the orchestrator uses a greedy algorithm to iteratively find the best possible device for each one of the *nodes* by filtering the devices that comply with each *node*'s predicates and choosing the one with a higher value of a heuristic. This heuristic is calculated by taking into account the number of priorities the device can provide and the number of *nodes* that have already been assigned to the device. The NoRDOr assignment algorithm does not meet the requirements of our solution because the information it receives when choosing the *nodes*' assignment is not enough to make a privacy-oriented decision to guarantee a safe flow of private data in the system.

We started by calculating all possible *node* assignments. However, we had some memory and efficiency problems when deploying more complex *flows* since its computation was too heavy for the Node-RED orchestrator. To prevent problems like this from appearing during the orchestration, we decided to use a task assignment algorithm to generate the first *node* assignment. Finding and choosing the best possible algorithm for our system is out-of-scope, and so we chose to use

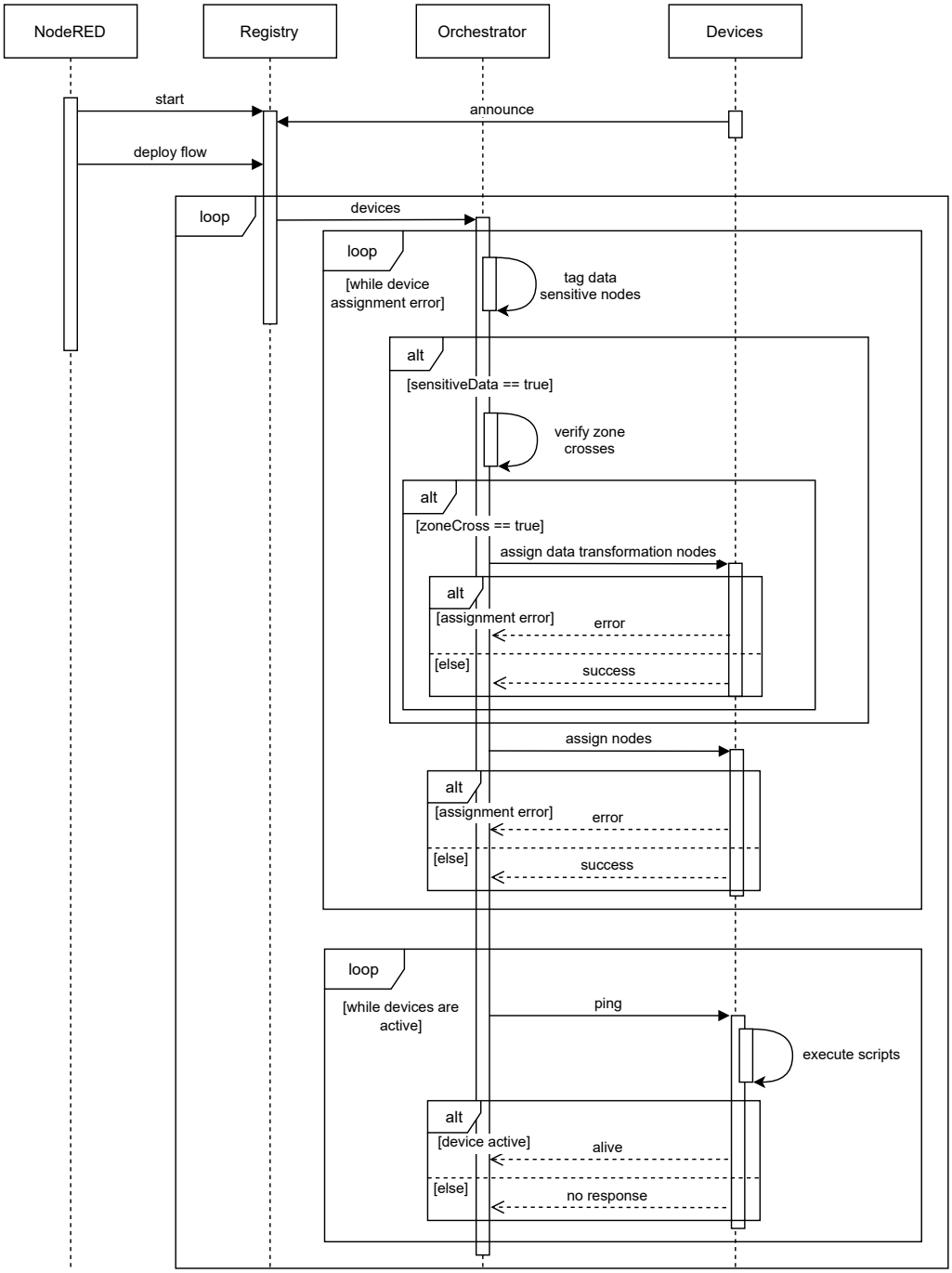

Figure 5.3: Orchestration sequence diagram.

the Simulated Annealing (SA) algorithm in our solution since it is simple to implement and is capable of finding near-optimal allocation efficiently for this system and scenarios [6, 44]. With this algorithm, we aim to obtain a *node* assignment as close as possible to an ideal one given a set o heuristic values defined next. The SA algorithm uses an energy function which it intends to minimize. In our solution the energy function corresponds to the function that calculates the score

of an assignment.

The implemented SA algorithm starts off by using a random possible assignment and sets it as the current assignment. The algorithm proceeds to generate a random neighbor to the current assignment, accepting it if the assignment score is better than the score of the current assignment. Otherwise, the algorithm can still accept a worse assignment with a calculated probability, which tends to decrease during the algorithm iterations. The algorithm only stops once the value, know as *temperature*, reaches zero. This value decreases in each iteration by a constant *cooling rate*. For experimental purposes, we chose to use the value 10000 as the *temperature* and value 0.003 as the *cooling rate*. However, these values may not be the best for our system, and their improvement is out-of-scope for this work. All the random neighbor assignments generated by the algorithm take into account the devices' status — active or not — and the match of capabilities between the devices and *nodes*.

Algorithm 1 Calculate *node* assignment score.

Input : nodesAssignment, deviceList, $\alpha = 0.4, \beta = 0.2, \gamma = 0.3, \theta = 0.1$

Output : assignmentScore

1: $\text{nodesSum} \leftarrow \sum_{d \in \text{deviceList}} \left| (\#\text{devices})^{-1} - \frac{\#d.\text{nodes}}{\#\text{nodes}} \right|$

2: $\text{nrTransformations} \leftarrow \#(n \in \text{nodesAssignment} | n.\text{sensitiveData} \wedge n.\text{zoneCross})$

3: $\text{prioritiesSum} \leftarrow \sum_{n \in \text{nodesAssignment}} \frac{\#(n.\text{priorities} \cap n.\text{device.capabilities})}{\#n.\text{priorities}}$

4: $\text{predicatesSum} \leftarrow \sum_{n \in \text{nodesAssignment}} \frac{\#(n.\text{predicates} \cap n.\text{device.capabilities})}{\#n.\text{device.capabilities}}$

5: $\text{transformationsScore} \leftarrow (1 + \text{nrTransformations})^{-1}$

6: $\text{nodesScore} \leftarrow (1 + \text{nodesSum})^{-1}$

7: $\text{prioritiesScore} \leftarrow \frac{\text{prioritiesSum}}{\#\text{nodes}}$

8: $\text{predicatesScore} \leftarrow \frac{\text{predicatesSum}}{\#\text{nodes}}$

9: **return** $\alpha \times \text{transformationsScore} + \beta \times \text{nodesScore} + \gamma \times \text{prioritiesScore} + \theta \times \text{predicatesScore}$

The score calculation of a possible *node* assignment takes into account the number of transformation *nodes* the orchestrator needs to introduce in the system, which is the most valued heuristic with 0.4 factor. The number of priorities each device can provide to the respective *nodes* is measured with a factor of 0.3, while the number of predicates each device can provide is measured with a factor of 0.1. Lastly, with a factor of 0.2, the average number of assigned *nodes* to each device is measured. The pseudo-code for the score calculation of a *node* assignment can be found in Algorithm 1. The goal of the values used is to find the best assignment possible that reduces the number of transformation *nodes* inserted to the system — which can also be understood as the number of zones crossed by data — and spreads the tasks through all the available devices. Several examples of *node* assignments will be presented along with the discussion of the performed experiments to validate our implementation (*cf.* Section 6.3, p. 49).

5.5 Known Limitations

Despite ensuring privacy is kept for sensitive data in our system, there are some limitations to our solution regarding the insertion of transformation *nodes* in the *flow*.

In some cases, the *flow* is successfully deployed, and the *nodes* are correctly assigned to the respective devices. However, the flow does not act as expected because some *nodes* cannot understand the messages they receive. This *node*'s misunderstanding can be explained by the insertion of transformation *nodes* that can change the structure of the messages exchanged that the *nodes* are expecting to receive.

Similarly, the addition of transformation *nodes* can change the expected behavior of a *flow* even if the privacy is kept. For example, let's assume a *flow* with two *nodes* where *node* A is responsible for measuring the heart rate of a patient while *node* B is responsible for checking if the value received from *node* A is greater than a certain *X* value, triggering an alert message if it is verified. In the possibility of the *nodes* being assigned to devices in different zones, an *aggregation node* will be inserted in the middle of *node* A and B communication. The new *node* will collect several values from *node* A and send an average value to *node* B. Even if *node* A sends a value that is greater than *X*, there is a possibility that the average value will not get greater than *X*, and so no alert will be triggered. Even though the system complies with the implemented privacy constraints, the *flow* fails to fulfill its expected behavior, which can be critical in some scenarios.

In an initial phase of the development of our solution, we chose to make the devices announce their respective zone to the orchestrator without any kind of verification. This allowed us to focus more on improving the orchestrator to correctly handle privacy-sensitive data when crossing zones. However, by doing this, we are introducing a possible security flaw into the system since unsafe devices can announce themselves as if they belong to a zone which they do not just to get access to private data without any type of transformation. As future work, other strategies should be tried, such as devices' zone being configured directly in the orchestrator or the devices' announcement being validated by the orchestrator.

In Section 6.2 (p. 43), we explore in experiments two of the mentioned limitations, while in Section 7.4 (p. 71), we look into possible solutions for all of them.

5.6 Summary

In this chapter, we describe in detail the implementation choices made during the development of our solution.

Section 5.2 (p. 34) describes the implemented changes related to the devices. We have introduced the concept of zones to the devices, which can be understood as virtual barriers that separate devices that can be trusted from devices that cannot be trusted. A device in the same zone as the zone where data is being produced is considered safe to handle privacy-sensitive data.

The introduction of zones and keeping the data flowing between a restricted group of devices was not enough since other devices' capabilities were being wasted. To tackle this problem, our

implementation allows the orchestrator to insert transformation *nodes* in during the assignment to devices. These transformation *nodes* are then used to transform privacy-sensitive data before it leaves a zone. The implementation of transformation *nodes* is detailed in Section 5.3 (p. 35).

To prevent privacy-sensitive data from being processed in possible unsafe locations, we have improved the *node* assignment algorithm as described in Section 5.4 (p. 36). Our assignment algorithm can prioritize the allocation of privacy-sensitive *nodes* by preventing zones from being crossed by private data. This could be achieved by the addition of a heuristic to the algorithm that takes into account the number of transformation *nodes* the system would have to insert for each one of the assignments.

Our solution still faces some limitations, mainly regarding the insertion of transformation *nodes* in the system. The addition of these *nodes* to the system can protect the data but, without a correct configuration, they may change the expected behavior of the system's *flow*. Moreover, by allowing the devices to announce their respective zone, we are also introducing a possible security flaw. These limitations are described in more detail in Section 5.5 (p. 39).

Although, with the introduction of zones, the addition of transformation *nodes*, and the improvement in the *node* assignment algorithm, our solution can successfully answer the research questions present in Section 4.5 (p. 29).

Chapter 6

Experimentation and Evaluation

This chapter presents and analyzes the results of the experiments that will be used to validate and evaluate our implementation. Section 6.1 describes the scenarios where we based the definition of our experiments. Section 6.2 (p. 43) proposes the experiments to be performed, detailing the *flows* and devices' specification to be used. Then, Section 6.3 (p. 49) analyzes and discusses the obtained results while in Section 6.4 (p. 63) we present the replication package for the experiments. In Section 6.5 (p. 64) we analyze if our solution meets the defined requirements. Finally, Section 6.6 (p. 64) presents answers to the research questions that guided the development of our solution.

6.1 Scenarios

To validate and evaluate if our implementation was able to achieve the goals mentioned earlier (*cf.* Section 1.4, p. 3), we evaluate our proposal with a virtual setup using containerized virtual devices. More details regarding the setup used for the experiments in Section 6.2 (p. 43).

We defined the following two experimental scenarios (ES) as a starting point for the experiments to evaluate the implementation:

ES1 A system with two devices, one capable of gathering body temperature data and another responsible to publish data to a specified MQTT topic. For this scenario, we use a simple Node-RED *flow* (*cf.* Figure 6.1) where it is expected that the data generated by the *body_temperature node* is sent to the device running the *output node*. We aim to test how the system reacts when data is sent from one zone to another by forcing the *nodes* to be executed in devices in different zones, as well as verify other system behaviors when changing small details in the system.

Figure 6.1: Node-RED implementation of scenario 1.

ES2 An AAL scenario inspired in the eCAALYX system [59] where several metrics are collected from an elderly person, which are, posteriorly, accessed by the respective doctor, caretaker, and relatives. The eCAALYX system data acquisition is centralized, and its analysis is performed in a central server. Figure 6.2 represents a simplified overview of the eCAALYX system.

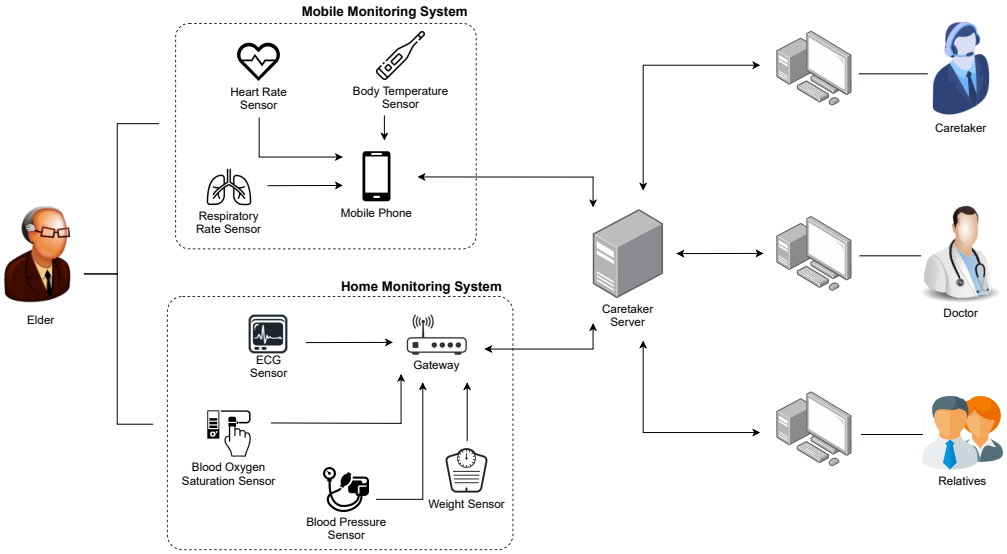

Figure 6.2: The eCAALYX system overview, adapted from [59], including two main parts: the mobile monitoring system and the home monitoring system. Also shown are the end-users: Elder, Caretaker, Doctor and Relatives.

To evaluate our system, the scenario was adapted by introducing zones in the system. In *Zone 0* we include the devices and sensors that collect the elderly data, both the mobile monitoring system and the home monitoring system. The mobile monitoring system is composed of the (1) heart rate sensor, (2) body temperature sensor, (3) respiratory rate sensor, and (4) mobile phone, while the home monitoring system includes the (1) ECG

sensor, (2) blood oxygen saturation sensor, (3) blood pressure sensor, (4) weight sensor, and (5) a gateway. For the caretaker, doctor, and relatives, *Zones 1*, *2*, and *3* were, respectively, assigned. In this scenario, the data gathered in *Zone 0* needs to be sent to all the other zones respecting the zone boundary-crossing constraints. The zones assignment to this scenario can be seen in Figure 6.3.

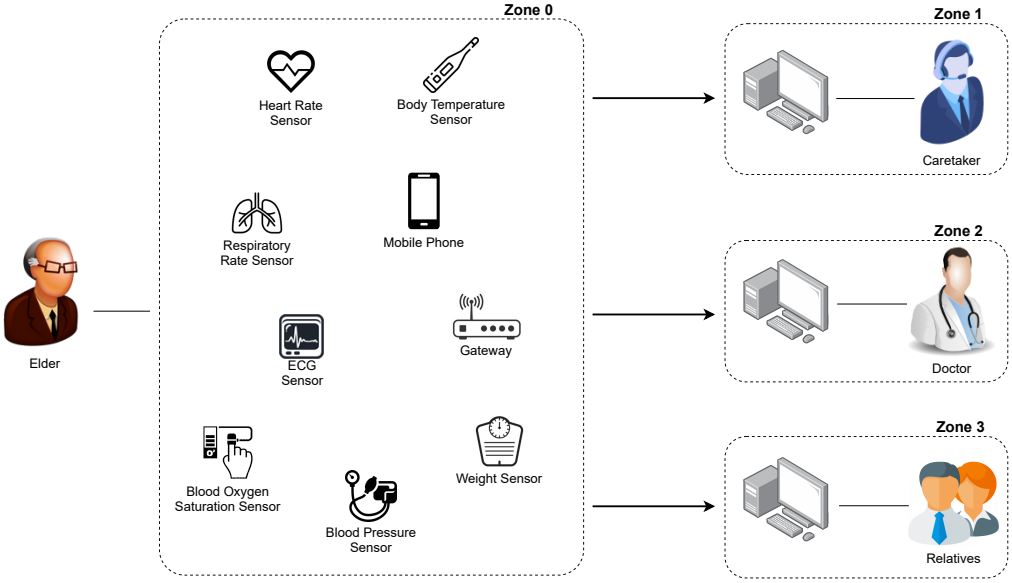

Figure 6.3: Overview of Scenario 2 adapted from eCAALYX system. Includes a visual representation of the assigned zones: devices and sensors from the Elder are in *Zone 0*, the Caretaker is in *Zone 1*, the Doctor in *Zone 2* and the Relatives are assigned to *Zone 3*. Devices in Zone 0 communicate with each one of the other zones.

To summarize, **ES2** represents a real-world scenario inspired in AAL scenarios [59, 34, 33, 72, 73] which has the objective of testing the system with a more complex Node-RED *flow* (*cf.* Figure 6.4, p. 44) and more devices. With this scenario, we expect that all the data collected from the devices of the elder are sent to the caretaker, doctor, and relatives in real-time.

6.2 Experiments

The series of experiences were performed in a MacBook Pro Mid-2015 running macOS Big Sur version 11.3 with a 2.2GHz Intel Core i7-4770HQ processor and 16Gb of RAM. The Eclipse Mosquitto MQTT Broker was running the version 2.0.10, the modified version of Node-RED (NoRDOr) uses the version 1.0.6 of Node-RED and the MicroPython firmware used for the devices was running the version 1.12. The virtualized experiments were performed using Docker Desktop for Mac running version 3.4.0, which in turn was running Docker Engine 20.10.7.

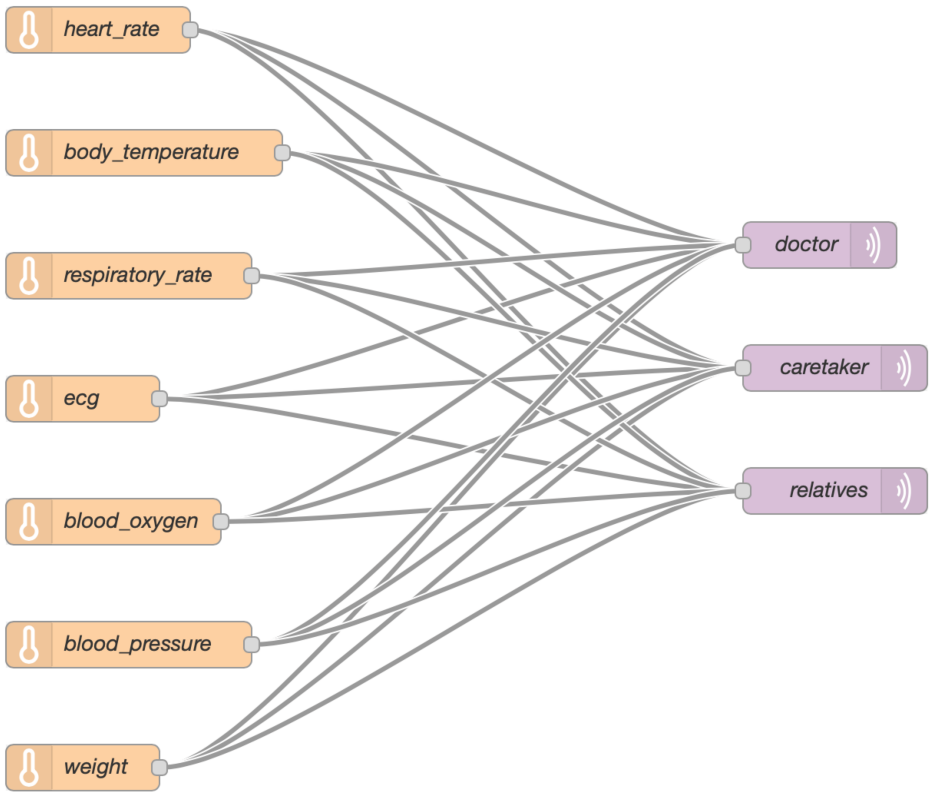

Figure 6.4: Node-RED implementation of scenario 2.

6.2.1 ES1 Experiments

Regarding scenario ES1, we divided our experiments into two main groups: (1) *Sanity Checks* to validate if our solution complies with the constraints defined in the simplest cases; and (2) *Experimental Tasks* where we change certain aspects of the system to verify some specific behaviors.

6.2.1.1 Sanity Checks

ES1-SC1 A simple *flow* (*cf.* Figure 6.1, p. 42) where the *body_temperature node* is configured to collect data with an interval of 5 seconds and is tagged as a *node* that produces sensitive data. The devices were specified with different capabilities and zones, as shown in Table 6.1, with the objective of forcing the *nodes* to be executed in different devices, which in turn forces the data to flow from one zone to another.

Table 6.1: ES1-SC1 devices' specification.

Device	Capabilities	Zone
1	body_temperature, aggregate	z1
2	output	z2

ES1-SC2 With two devices, none of them with capabilities to execute transformation nodes (*cf.* Table 6.2). The orchestrator will try to assign the nodes to the devices in different zones but we expect that it will not be possible since it is required to have a transformation node when private data crosses zones.

Table 6.2: ES1-SC2 devices' specification.

Device	Capabilities	Zone
1	body_temperature	z1
2	output	z2

6.2.1.2 Experimental Tasks

We further defined a set of complementary experiments for ES1, where we want to evaluate how the system behaves when device-related aspects change, namely:

ES1-A With four devices, each one with different processing capabilities as shown in Table 6.3. The devices with capabilities to execute transformation *nodes* will appear and disappear one by one, forcing the system to re-orchestrate. We expect that the system can assign another device to execute the transformation *node* whenever possible, if the previously assigned device fails.

Table 6.3: ES1-A devices' specification.

Device	Capabilities	Zone
1	body_temperature	z1
2	aggregate	z1
3	aggregate	z1
4	output	z2

ES1-B Both devices have the same capabilities but remain in different zones as it can be seen in Table 6.4. We will verify the orchestrator decision of assigning *nodes* in different zones or in the same zone. With this experiment we expect the orchestrator to give more preference in executing all the nodes in one zone instead of using both devices.

Table 6.4: ES1-B devices' specification.

Device	Capabilities	Zone
1	body_temperature, aggregate, output	z1
2	body_temperature, aggregate, output	z2

ES1-C The same devices' specification used in **ES1-B**. However, we will evaluate if the orchestrator decision when assigning *nodes* to devices can change comparatively to **ES1-B** by modifying the heuristic scores of the assignment algorithm (*cf.* Section 5.4, p. 36). For this

experiment we will use a factor of 0.2 to the number of transformation *nodes* the orchestrator needs to introduce in the system heuristic, and a factor of 0.4 to the heuristic of the average number of assigned *nodes* to each device.

ES1-D With five devices distributed between two zones and with the capabilities shown in Table 6.5. A *nothing node*, which only redirects the received data to the following linked *node*, was introduced in the *flow* (*cf.* Figure 6.1, p. 42), between the *body_temperature node* and *output node*. With this experiment we want to verify how the orchestrator finds a middle term in distributing *nodes* equally between the devices and, at the same time, avoiding data to flow from one zone to another, which may lead to the insertion of more *nodes* into the system. We expect that the orchestrator gives more preference to the execution of all the nodes in "z2" since it can distribute equally the number of nodes per device, comparatively to "z1".

Table 6.5: ES1-D devices' specification.

Device	Capabilities	Zone
1	body_temperature, aggregate	z1
2	output, nothing	z1
3	body_temperature, output, nothing	z2
4	body_temperature, output, nothing	z2
5	body_temperature, output, nothing, aggregate	z2

6.2.2 ES2 Experiments

With ES2 experiments, we intend to verify how our system performs in a real-case scenario by collecting several metrics of the system and further analyzing them. We divided the following experiments into two main groups: (1) *Experimental Tasks* where we explore more system behaviors; and (2) *Limitations* to show the current shortcomings of our implementation.

6.2.2.1 Experimental Tasks

In the experimental tasks, we use a *flow* (*cf.* Figure 6.4, p. 44) where the nodes that produce data are set up to collect data with an interval of 5 seconds and, all of them are tagged as producers of sensitive data. The time between each collection of data by the *nodes* is chosen merely for experimental purposes and does not reflect reality for some of the simulated sensors (*e.g.*, weight sensor) since normally they do not produce data so frequently.

ES2-A Using a more complex *flow* (*cf.* Figure 6.4, p. 44) and the devices' capabilities and zones defined as shown in Table 6.6 (p. 47). The expected results for these experiments are that data collected from the elderly sensors can reach the respective caretaker, doctor, and relatives but only after suffering a transformation (*e.g.*, aggregation, anonymization, encryption) since the data collected is highly privacy-sensitive and should not be easily accessed by everyone.

Table 6.6: ES2-A devices' specification.

Device	Capabilities	Zone
1	heart_rate, ecg, blood_oxygen	z1
2	blood_pressure	z1
3	body_temperature	z1
4	respiratory_rate	z1
5	weight	z1
6	gateway	z1
7	mobile	z1
8	caretaker	z2
9	doctor	z3
10	relatives	z4

ES2-B Using the same *flow* and devices' specification as **ES2-A**, however, with this experiment, we aim to simulate when the elderly leaves the house to evaluate if the system is capable of still running properly. The type of data collected in this scenario needs to be delivered in real-time so that medical help can be triggered as fast as possible when needed. To simulate an elderly leaving their house, we introduce a failure to the Gateway forcing the system to re-orchestrate the transformation *nodes*, expectably, to the Mobile Phone.

6.2.2.2 Limitations

To explore our solution's limitations pointed out in Section 5.5 (p. 39), we will carry out two experiments. For such, we use a simpler *flow* based on the ES2, as can be seen in Figure 6.5. For both experiments, we will simulate a sensor that collects the heart rate value of an elderly with an interval of 5 seconds, and whenever the heart rate value reaches 100, an alert should be sent. In the following experiments, we aim to analyze if the system fulfills the privacy constraints as well as if the *flow* works as expected.

Figure 6.5: Node-RED implementation for the limitation experiments in scenario 2.

ES2-L1 With four devices distributed between two zones and with the capabilities shown in Table 6.7 (p. 48). With this experiment, we expect to verify a limitation introduced when a transformation *node* is added to the system, which may change the structure of the messages some *nodes* may be used to.

ES2-L2 With five devices distributed between two zones with the capabilities shown in Table 6.8 (p. 48). For this experiment, we will change the behavior of two nodes: (1) *if node* to only send a message when the result of its condition is "True", and (2) *aggregate node* to calculate the

Table 6.7: ES2-L1 devices' specification.

Device	Capabilities	Zone
1	heart_rate	z1
2	aggregate	z1
3	if	z2
4	alert	z2

mean value of the aggregated messages instead of sending all the messages as an array. With this experiment, we aim to show the system complies with the privacy constraint but does not fulfill the expected behavior of the *flow*.

Table 6.8: ES2-L2 devices' specification.

Device	Capabilities	Zone
1	heart_rate	z1
2	aggregate	z1
3	heart_rate	z2
4	if	z2
5	alert	z2

6.2.3 Metrics collected

For each one of the experiments mentioned previously, several metrics of the system will be measured. Each device firmware was modified to send the respective metrics to MQTT topics every 5 seconds, consequently being captured by a bridge that populated an InfluxDB database (version 1.8). The metrics collected are explained next:

Uptime: represents how long a device is running during an experiment. This graph is useful to verify when devices fail and when they announce themselves back to the system.

Number of allocated nodes: shows how many nodes the orchestrator assigns to each device that announced itself to the system.

Number of allocated transformation nodes: shows how many transformation nodes were introduced in the system and to which devices they are allocated.

Number of allocated nodes per zone: represents the number of nodes assigned per zone. This graph is useful to understand some of the choices of the orchestrator.

Number of exchanged messages per device: displays the number of messages sent and received by each device. This graph can be used to verify if messages are lost and/or if the communication between nodes and devices is working as supposed.

6.3 Results Analysis

During the experiments, we collect several metrics that are detailed in Section 6.2.3 (p. 48) and present them in charts, as well as the *nodes*' assignment to devices, to better understand the system behavior and validate our implementation.

In the following sections, we present and analyze the results from the experiments.

6.3.1 ES1: Sanity Checks

6.3.1.1 ES1-SC1

In this experiment, the objective is to observe how the system handles zone crossing in a controlled way. The system is tested using two devices (*cf.* Table 6.1, p. 44) and a Node-RED *flow* with two *nodes* (*cf.* Figure 6.1, p. 42). A visual representation of the system events for this specific experiment is presented in Annex A (p. 75). In Figure 6.6, it is possible to see the metrics measured from the respective devices.

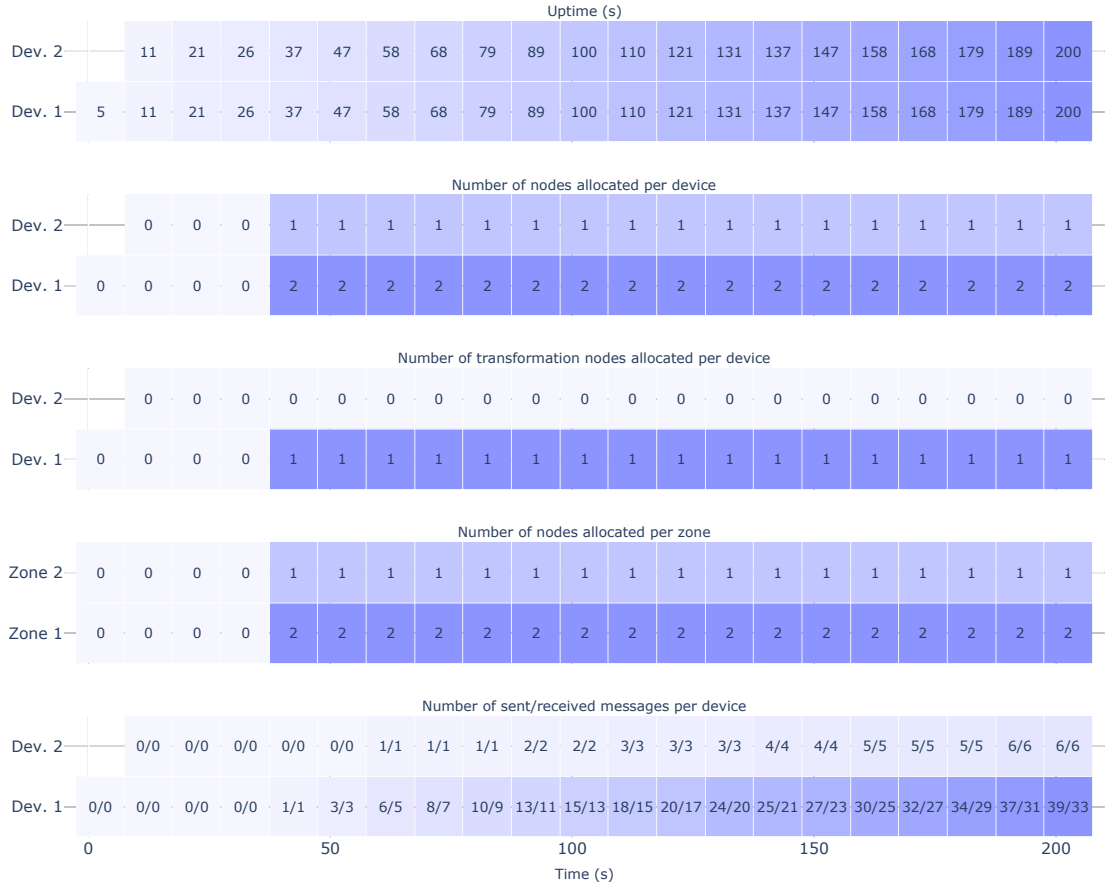

Figure 6.6: ES1-SC1 measurements.

Since the *nodes* are assigned to devices in different zones and the *body_temperature node* was tagged as a producer of privacy-sensitive data, the orchestrator needs to create a transformation *node* to prevent raw data from being sent from one zone to another. An *aggregate node* is assigned to *Device 1*, which is the same device where the *body_temperature node* is executed and, thus, they are executed in the same zone. Even though the *aggregate node* was not defined in the initial *flow* of this experiment, the final *flow* can be interpreted as Figure 6.7 shows. In the same figure, it is also represented the remaining *nodes*' assignment to each device.

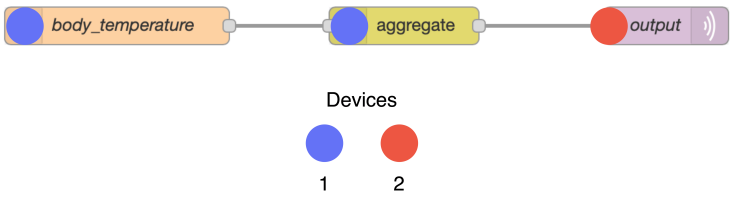

Figure 6.7: ES1-SC1 *node* assignment.

Each *node* starts to communicate with each other as soon as the devices start executing the generated scripts sent by the orchestrator. All the messages sent and received by each device are counted to verify if all *nodes* receive and produce the expected output messages and to check if the data is transformed before leaving *Device 1* and reaching *Device 2*. The number of sent and received messages per device can be seen in Figure 6.6 (p. 49). As expected, *Device 1* sends and receives more messages since it is executing two *nodes* that communicate between each other. The fact that the two sequential nodes being executed in the same device communicate via MQTT topics instead of calling themselves through code was a limitation already pointed out in Silva *et al*. [19] work, and its improvement is out-of-scope for our solution. The difference between sent and received messages in *Device 1*, which constantly increases over time, represents the number of messages sent by the *aggregate node* to *Device 2*, which is the same number as the number of messages received by *Device 2*. Taking into account that the *output node* is a MQTT *node*, the number of messages received and sent by *Device 2* are the same because for each message received, the *output node* forwards it to the specified MQTT *topic*.

The results from this sanity check are satisfactory since the system performs as expected, showing that no raw data is sent from one zone to another without first going throw a transformation process.

6.3.1.2 ES1-SC2

The second sanity check of ES1 has the objective of validating how the system reacts to the deployment of a *flow* where it is impossible to transform data since no device has such capabilities (*cf.* Table 6.2, p. 45).

As expected, the orchestrator can't find a feasible solution that guarantees data privacy because both *nodes* are forced to be executed in different zones, and it is not possible to transform the data

collected before it crosses zones. Instead, the orchestrator produces an error that can be used in future work (*cf.* Section 7.4, p. 71), to inform the user of the lack of privacy-preserving capability of the system.

6.3.2 ES1: Experimental Tasks

6.3.2.1 ES1-A

This experiment is a complement to experiment **ES1-SC1**, where we evaluate how the system behaves when devices capable of executing transformation *nodes* fail and reappear, using the same *flow* but with different devices' specifications (*cf.* Table 6.3, p. 45). The measurements performed for this experiment can be consulted in Figure 6.8.

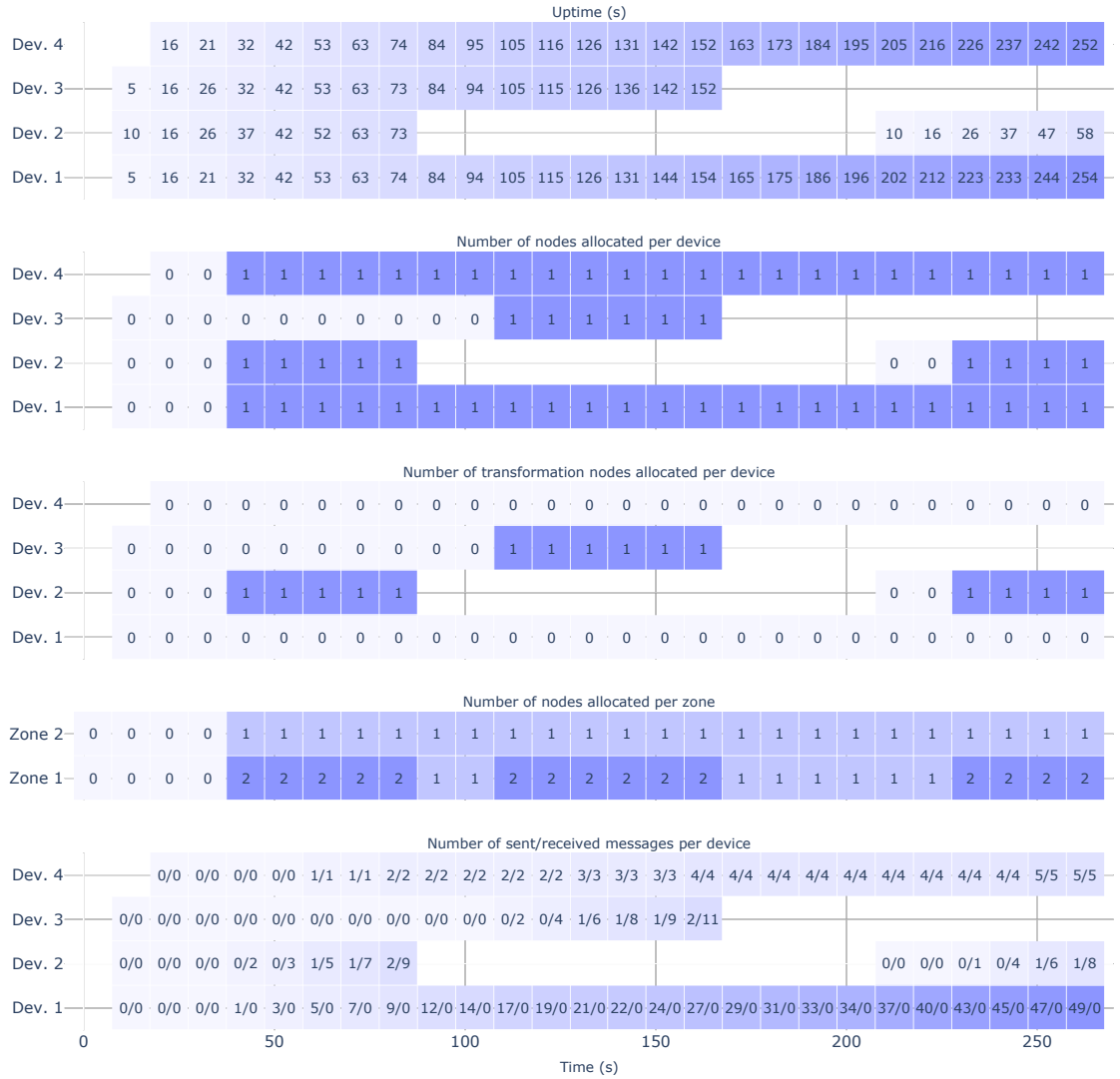

Figure 6.8: ES1-A measurements.

In an initial phase, since *body_temperature node* and *output node* are assigned to devices in different zones, the orchestrator introduces a transformation *node* to the system which is assigned to a different device from where the other nodes are assigned, as it can be observed in Figure 6.9.

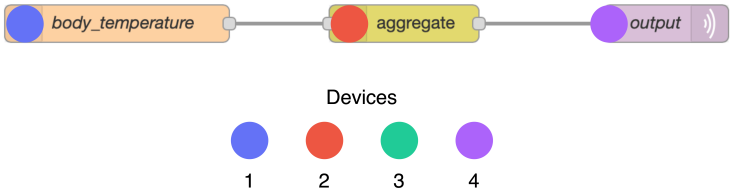

Figure 6.9: ES1-A *node* assignment.

By analyzing the uptime graphic (*cf.* Figure 6.8, p. 51), it is possible to identify when a device fails. Whenever a device fails to respond to the *ping* message sent by the orchestrator, it tries to re-orchestrate the *flow* with the remaining devices. However, sometimes, it may not be possible.

For this experiment, it is possible to verify that *Device 2* fails and the system, after a while, was able to re-orchestrate the *aggregate node* to *Device 3* since it has the needed capabilities. However, after the failure of *Device 3*, the orchestrator could not find any device with transformation capabilities, and so the system has no way of processing data before it changes zones. As soon as *Device 2* announces itself back to the orchestrator, the normal flow of the system is established.

Figure 6.8 (p. 51) also presents the number of messages sent and received by each device over time. When *Device 2* and *Device 3* are not working, it is possible to see that, even though *Device 1* is sending messages from the *body_temperature node*, they are not being received by *Device 4*. During that time, the number of messages received and sent by *Device 4* remains unchanged.

This experiment and results allow us to validate one of the main constraints of our system, the zone cross constraint, which ensures that no data is sent from one zone to another before being transformed.

6.3.2.2 ES1-B

Based on **ES1-SC1** (*cf.* Section 6.3.1.1, p. 49), this experiment makes use of the same *flow* and the same number of devices in the same zones. However, both devices have the same capabilities (*cf.* Table 6.4, p. 45). With this experiment, we want to verify what are the orchestrator priorities when assigning the *nodes* to the devices.

By analyzing Figure 6.10 (p. 53), it is possible to see that the orchestrator prefers to assign all the nodes to *Device 1* because, like that, there is no need to add a transformation *node* since the data does not flow between zones.

A few seconds after the first orchestration is performed, a failure is introduced to *Device 1* to verify how the orchestrator reacts. As expected, after the failure, all the nodes that are being executed in *Device 1* are moved to *Zone 2*.

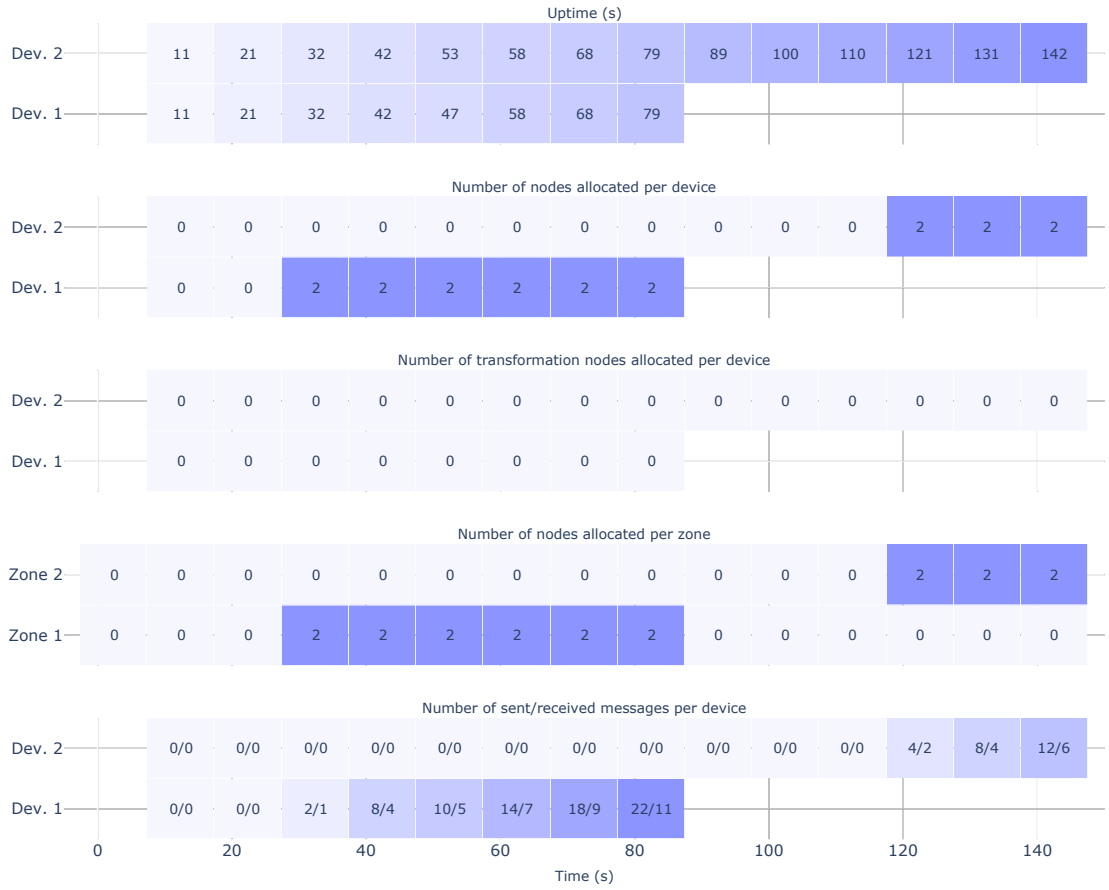

Figure 6.10: ES1-B measurements.

6.3.2.3 ES1-C

Similar to **ES1-B** (*cf.* Section 6.3.2.2, p. 52), the objective of this experiment is to verify if the orchestrator priorities can change. As explained in Section 5.4 (p. 36), the *node* assignment algorithm makes use of several heuristic values that change the score calculation of the best assignment. For this experiment, we change the number of transformation *nodes* the orchestrator needs to introduce in the system heuristic to a factor of 0.2, and we use a factor of 0.4 to the heuristic of the average number of assigned *nodes* to each device. With such factor values, we aim to give more priority to spreading the nodes among the available devices instead of reducing the number of zones crossed by the data that flows in the system. The *flow* and the devices we use are exactly the same used in **ES1-B**.

Since we gave more importance to the equal distribution of *nodes* between the devices when changing the assignment values, the orchestrator prefers to distribute the *nodes* among the devices instead of avoiding the zones being crossed by the data. In Figure 6.11 (p. 54), we can see that due to the *body_temperature* and *output node* being assigned to devices in different zones, the orchestrator adds a transformation *node* to the device generating data.

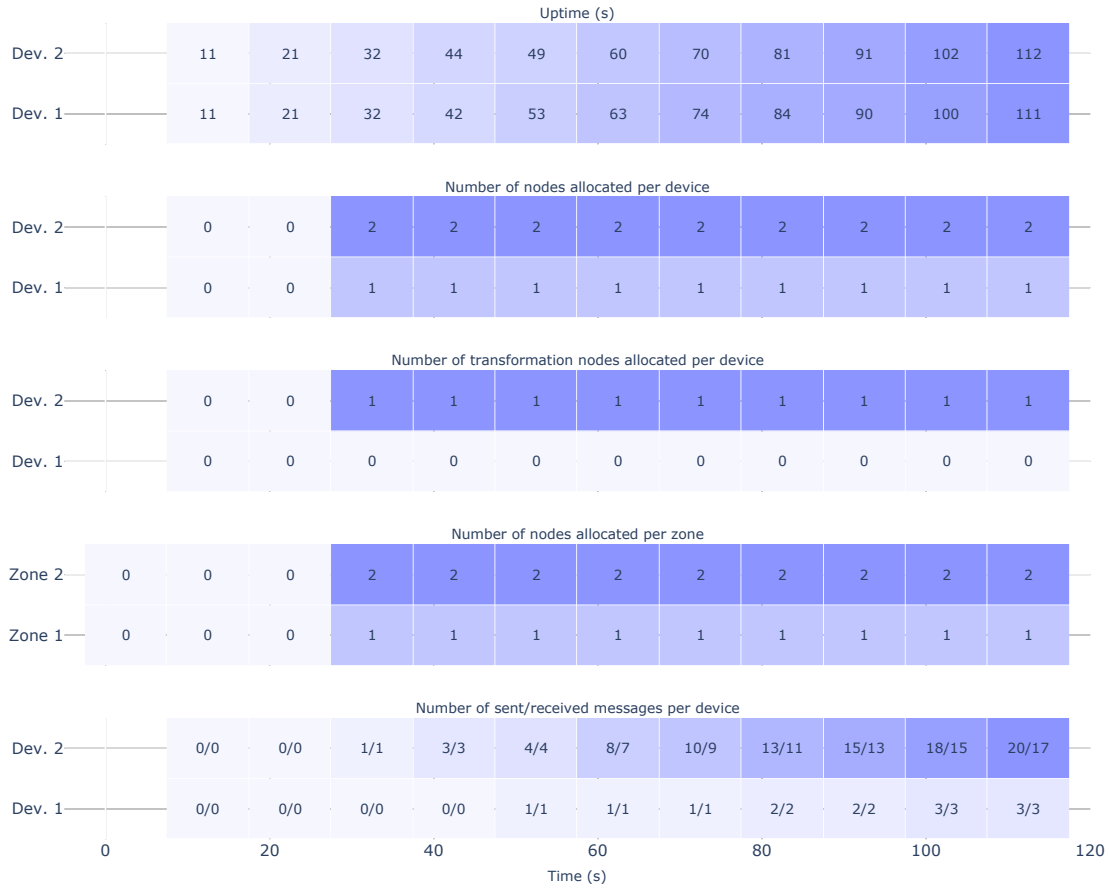

Figure 6.11: ES1-C measurements.

In comparison to the experiment **ES1-B**, the algorithm chooses to assign the nodes to different devices, giving less importance to the fact of data flowing from one zone to another. However, because of that, the orchestrator needs to insert a new node into the system. When comparing the number of nodes being executed in each device, we can conclude that, like experiment **ES1-B**, one of the devices needs to run two *nodes*. With this being said, the default values used for the assignment algorithm in **ES1-B** behave better than the ones we use for this experiment.

With this experiment, we can conclude that by changing the default values used in the assignment algorithm, we can change the behavior of the orchestrator when allocating the *nodes* to the devices where they will be executed.

6.3.2.4 ES1-D

To further assess the orchestrator behavior, we use more complex devices' specification (*cf.* Table 6.5, p. 46) and introduce one *nothing node* which only forwards what it receives to the linked *nodes*. By increasing the complexity of the system, we allow the orchestrator to have more assignment possibilities, either by assigning the *flow* to the same zone or between zones, and even inside

each zone, the *nodes* can be assigned to different devices. With this experiment, we intend to have a better evaluation of the system's performance.

In the first *node* assignment, the orchestrator decides to send all the *nodes* to *Zone 2*, as can be interpreted in Figure 6.12. The reason behind the first assignment choice comes from the fact that the system can achieve a better *node* distribution among the devices in *Zone 2* than doing the same distribution in *Zone 1*.

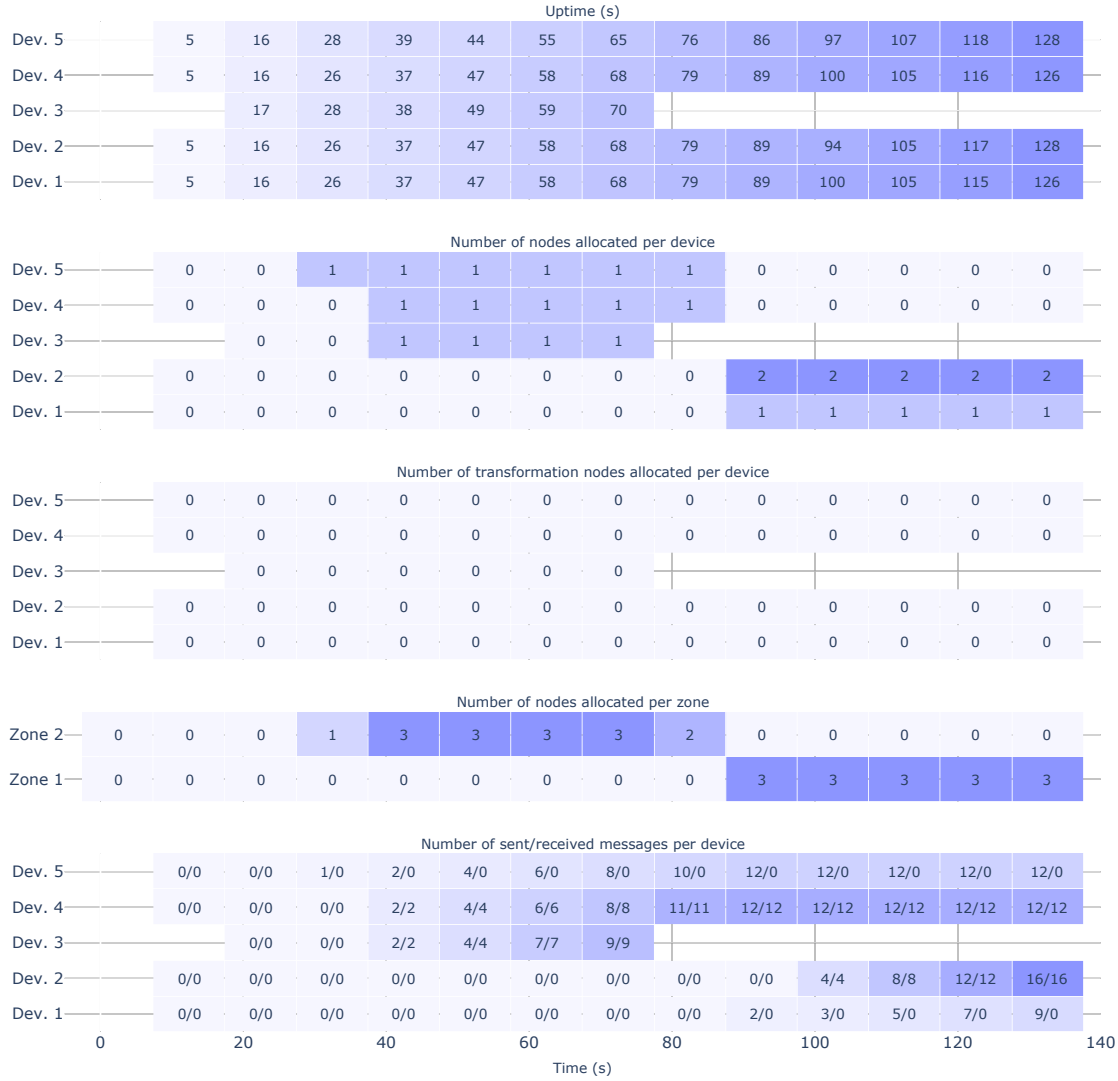

Figure 6.12: ES1-D measurements.

However, when a failure is introduced to *Device 3* at approx. 80 seconds, the system has to re-orchestrate choosing to move the *nodes*' execution to *Zone 1* instead of just assigning the *node* running in *Device 3* to another device in the same zone. This orchestrator behavior is expected due to the factor values used in the *node* assignment algorithm (*cf.* Section 5.4, p. 36). After reducing the number of zones crossed by data and equally distributing the *nodes* among devices, the orchestrator gives more preference to devices with fewer capabilities since there can be other

nodes with the requirement of more device capabilities. Both *node* assignments the orchestrator performs can be compared in Figure 6.13.

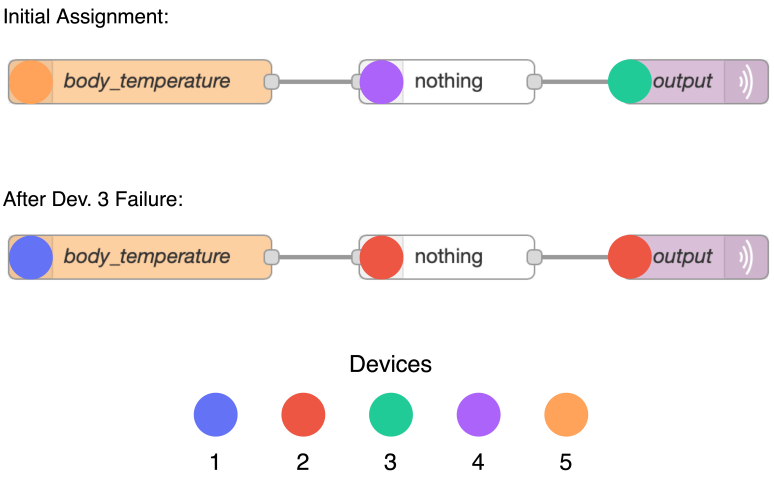

Figure 6.13: ES1-D *node* assignments.

By analyzing the messages exchanged per device and the number of nodes allocated per device (*cf.* Table 6.12, p. 55), it is possible to observe that, even though *Devices 4* and *5* do not stop working when *Device 3* fails, they stop the execution of the nodes that they had previously assigned once the re-orchestration was complete.

6.3.3 ES2: Experimental Tasks

6.3.3.1 ES2-A

As mentioned previously (*cf.* Section 6.1, p. 41), the second scenario consists of a real-case situation where data is collected by sensors in the possession of an elderly person, which are subsequently sent to the respective doctors, caretaker, and family. Since the data generated by the sensors may be privacy-sensitive in some situations, the objective of this experiment is to validate that no person outside the elderly zone can access the data that does not have been through a process of transformation.

The ES2 scenario, based on previous work done in the AAL area, was adapted to our system by introducing four zones (*cf.* Figure 6.3, p. 43). For the experiment, ten possible devices are defined and scattered around the zones where they made sense to be (*cf.* Table 6.6, p. 47).

The measurements performed for this experiment are visible in Figure 6.14 (p. 57). The orchestrator creates one node for each sensor owned by the elderly, so in *Zone 1*, the orchestrator assigns seven *nodes* that collect data and seven *aggregation nodes*. While the remaining three *nodes* are assigned to the other three zones, as expected.

By analyzing the number of messages sent and received by the devices, it is possible to see that *Devices 8*, *9*, and *10* receive the same amount of messages, which means that all the data collected from sensors is delivered equally between the doctor, caretaker, and relatives. Moreover, we can

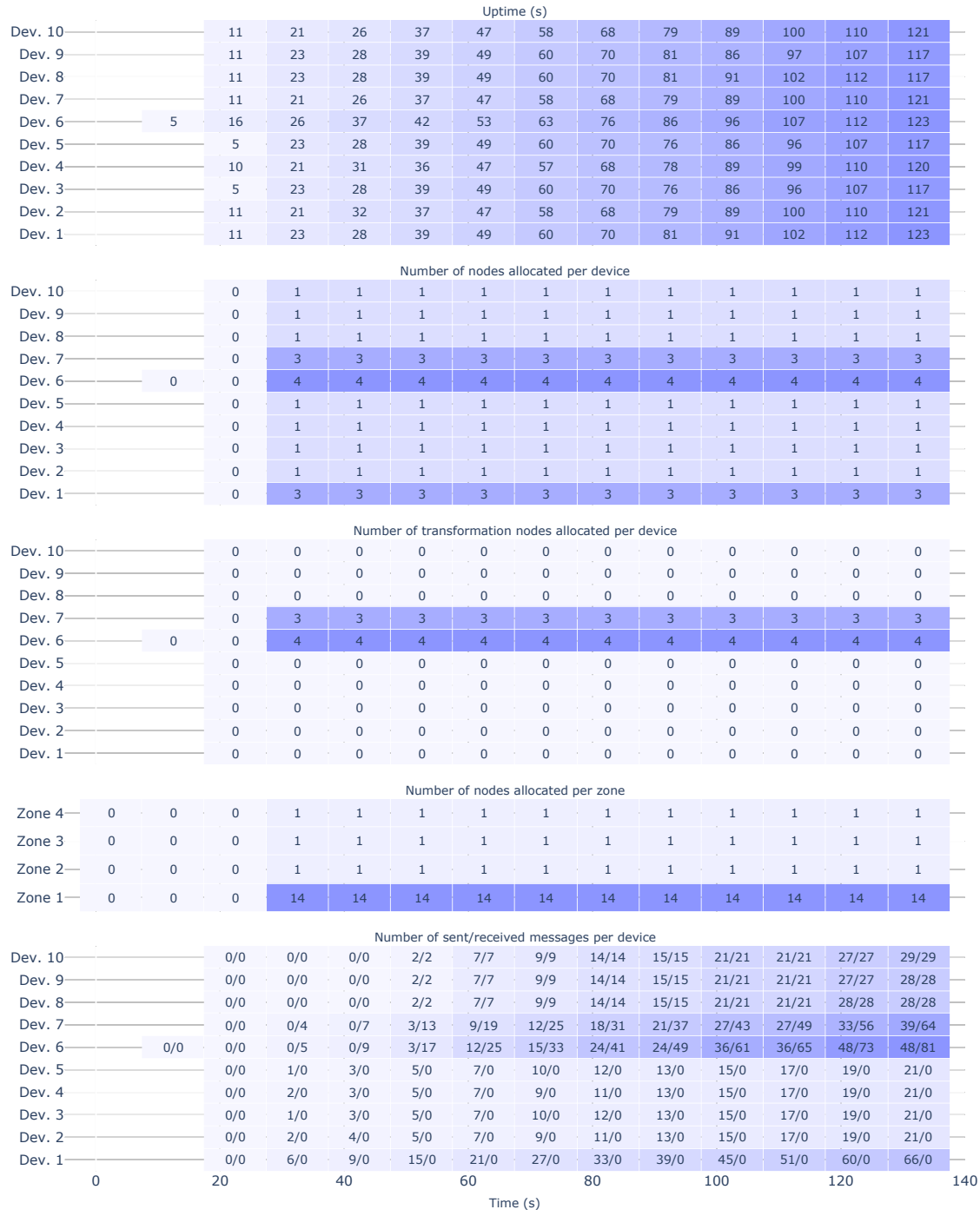

Figure 6.14: ES2-A measurements.

also observe that, over time, all the data produced in *Devices 1-5* is received in *Devices 6 and 7*, which is aggregated, and posteriorly sent to *Devices 8-10*. Even though the *aggregation nodes* are not defined when deploying the *flow*, the orchestrator introduces them to keep the privacy of data when being sent from one zone to another. A detailed node assignment of this scenario is

represented in Figure 6.15.

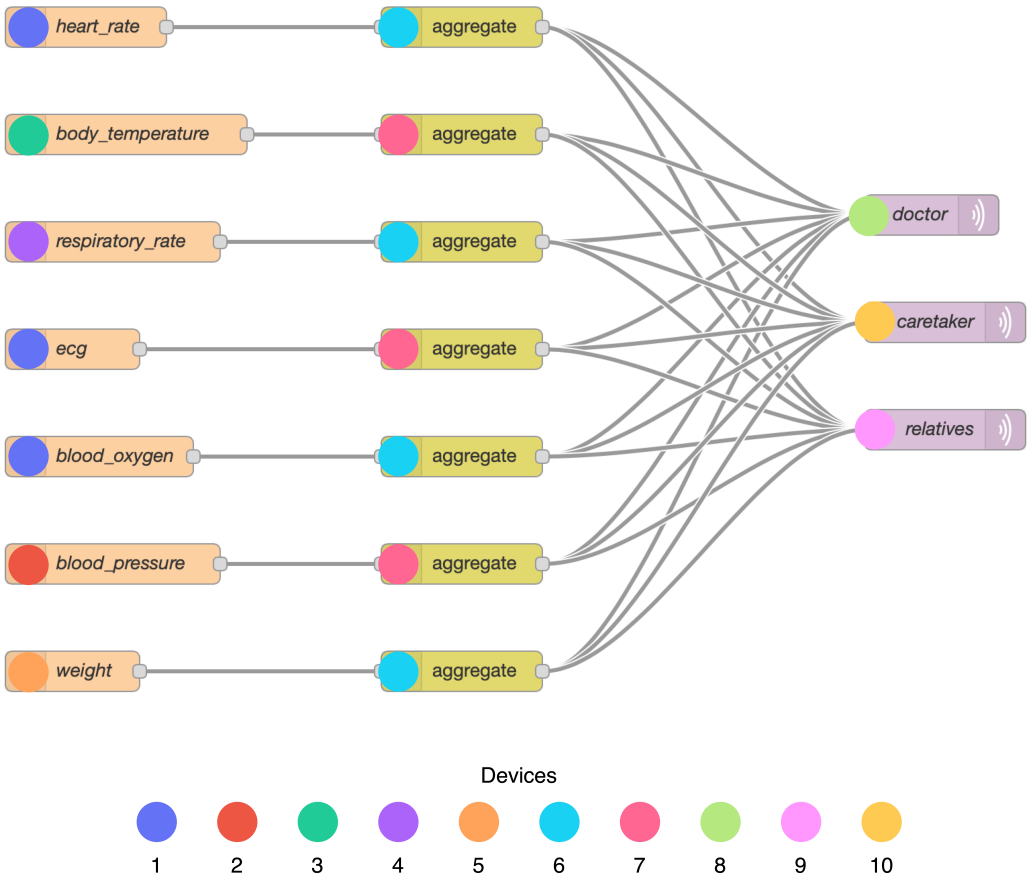

Figure 6.15: ES2-A *node* assignment.

As expected, even with a more complex scenario, the system was able to guarantee that the fundamental constraint of our system was kept.

6.3.3.2 ES2-B

For the second scenario, we perform another experiment where we intended to simulate the elderly leaving their house. To do so, we introduce a failure to the device that simulates the gateway since it is a device that should only be on the elderly house.

The event we aim to test occurs at approx. 90 seconds. As it can be seen in Figure 6.16 (p. 59), *Device 6*, which represents the elderly gateway, fails in the middle experiment and, since there are *nodes* assigned to it, the system re-orchestrates and allocates those nodes to *Device 7*, which is the only device available capable of transforming the data. Even though *Device 7* is given more computational load, it is able to receive, transform, and deliver all the messages to the destinations.

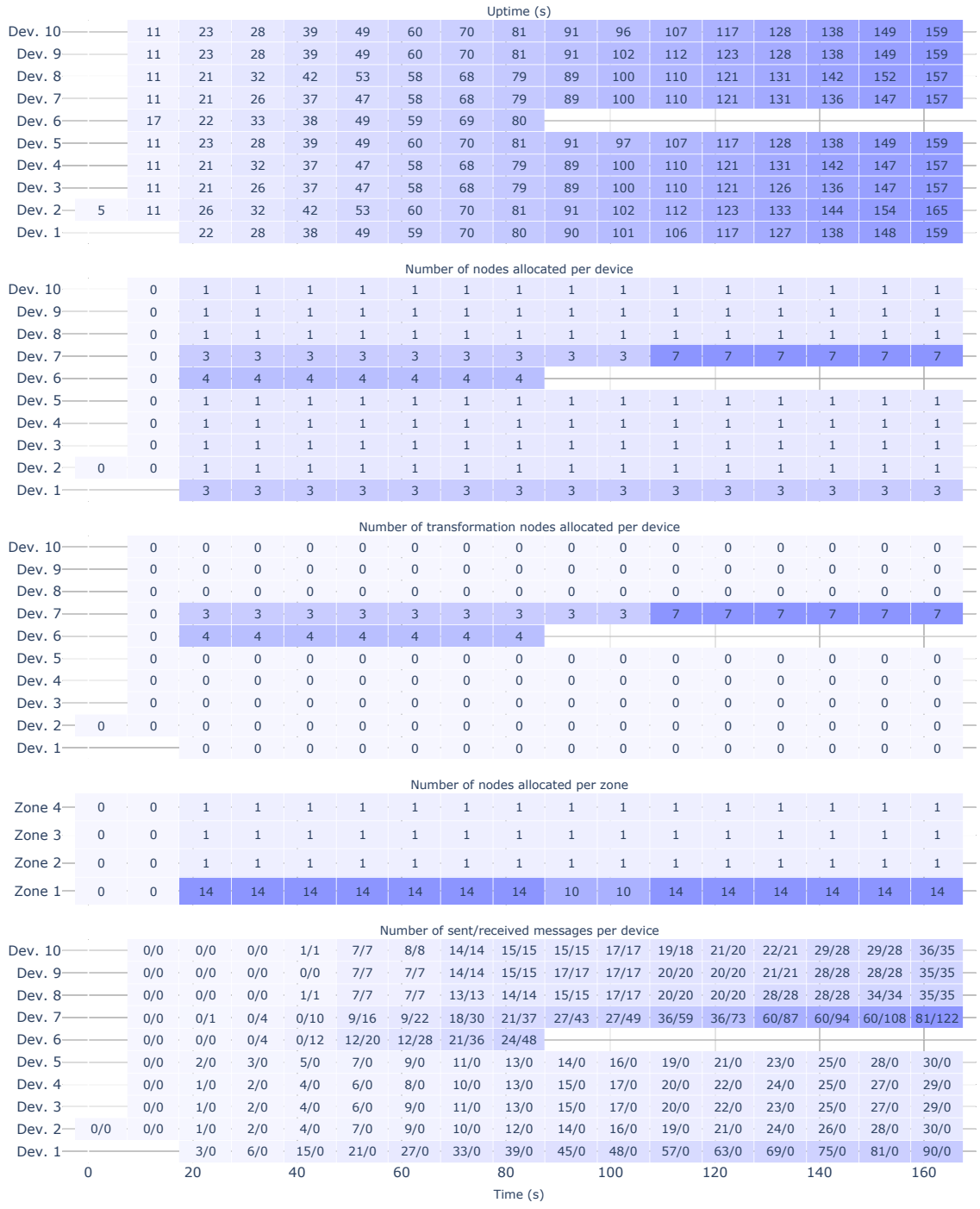

Figure 6.16: ES2-B measurements.

6.3.4 ES2: Limitations

6.3.4.1 ES2-L1

With this experiment, we have the objective of testing a limitation to our solution described in Section 5.5 (p. 39). This limitation occurs whenever a transformation *node* is inserted before a *node* that is expecting to receive data in a certain format. Since the transformation *node* can change the structure of the messages, it can impede the *flow* to run accordingly to what was defined in Node-RED.

In this experiment, we use an AAL-based *flow* (*cf.* Figure 6.5, p. 47) with an *if node* and the devices mentioned previously (*cf.* Table 6.7, p. 48). The *if node* is expecting to receive a message with a numerical value in the payload, which will then be used to verify a defined condition, and posteriorly the result will be sent to the following linked *node*. The measurements collected from the system are presented in Figure 6.17 (p. 61).

By analyzing the graphics, it is possible to see the *nodes* are assigned to two different zones, which leads to the addition of a transformation *node* in the *flow*. The *heart_rate node* is assigned to *Device 1*, which is in *Zone 1*, while the other *nodes* — *if node* and *alert node* — are assigned to *Zone 2*. To comply with the privacy constraints, the orchestrator adds an *aggregation node* to *Device 2* in *Zone 1*, which is placed right before the *if node*. As it can be seen in the number of sent and received messages per device graph, *Devices 1 and 2* are sending and receiving the messages correctly, but *Devices 3 and 4* are not exchanging messages as supposed. *Device 3* is still receiving the transformed data, but it cannot interpret the message, and consequently, nothing is sent to *Device 4*. We can see here that, even though the system complies with the defined privacy restrictions, the *flow* fails to be executed properly.

The limitation verified in this experiment can be explained due to the fact the *aggregation node* is configured to aggregate the messages into an array during 20 seconds and then send the group of messages as an array to the linked *nodes*.

6.3.4.2 ES2-L2

To explore the second limitation referred in Section 5.5 (p. 39), we proceed to use the same *flow* utilized in **ES2-L1**. This limitation does not refer to problems in understanding the messages exchanged, but to the fact the system cannot comply with the expected results of a system given its flow.

In this experiment, we use the same *flow* (*cf.* Figure 6.5, p. 47) used in **ES2-L1** which is based on an AAL scenario where the heart rate from an elderly is measured, and whenever it reaches a specified value, the system triggers an alert message, which in reality would be an alert sent to the Emergency Services. For this scenario, it is critical that when the heart rate reaches some value, an alert message is triggered. The measured values for the experiment are shown in Figure 6.18 (p. 62), and the respective *node* assignment can be seen in Figure 6.19 (p. 63). A graph comparing the expected number of alerts and the actually sent alerts is introduced to

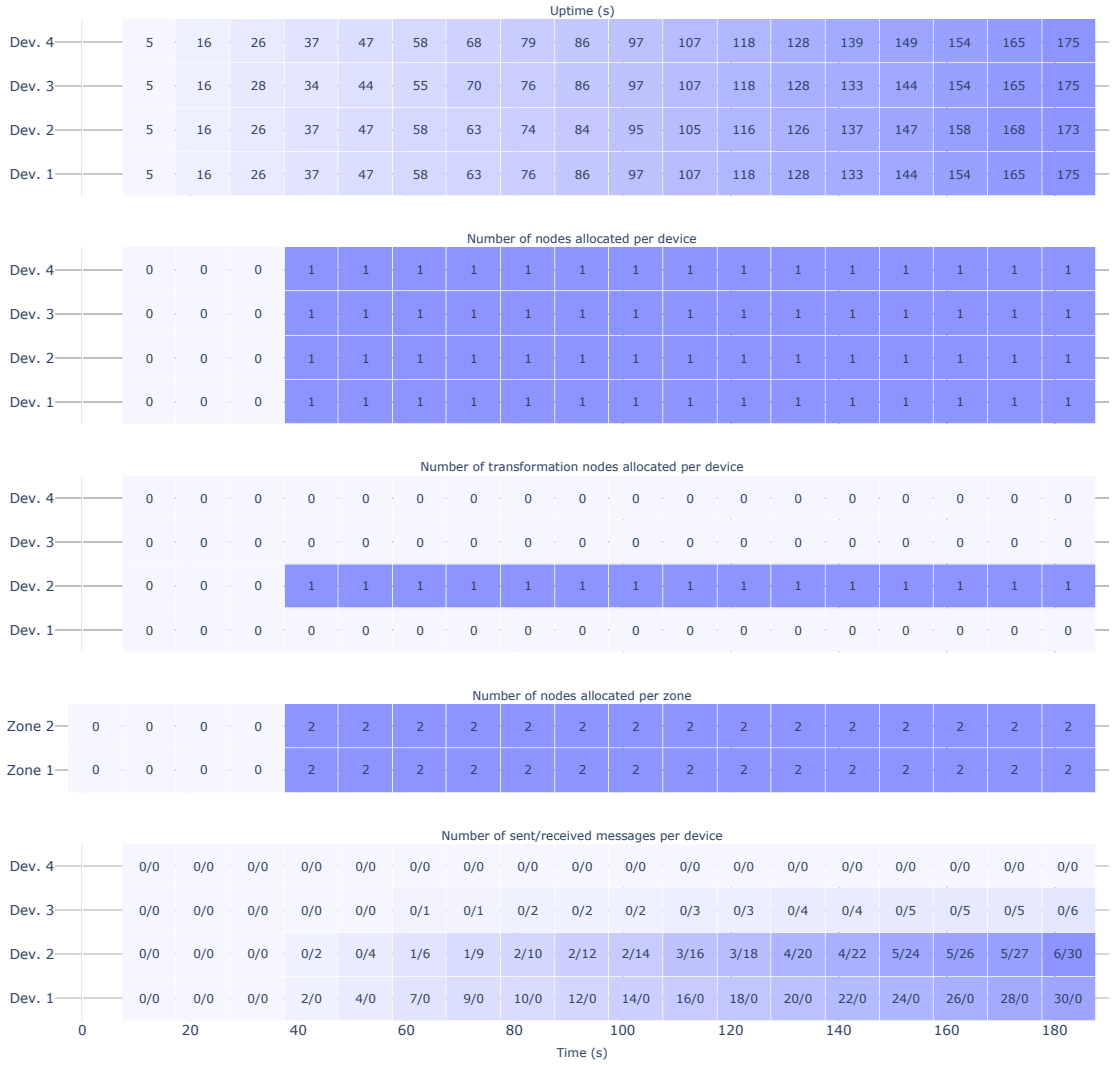

Figure 6.17: ES2-L1 measurements.

better understand the system capacity to fulfill the expected behavior from the *flow*. The system is expected to send an alert message whenever the heart rate goes beyond the value of 100.

As explained before, to perform this experiment, we have modified the behavior of two nodes: (1) the *if node* to only send a message when the result of the condition verified is "True"; and (2) the *aggregation node* to collect the messages and only send the mean value calculated from them.

In the initial phase of the experiment, it is possible to see that the system chooses to execute the *nodes* in *Zone 2* without the need for a transformation *node*. During some time, the system works as expected because for each time the heart rate reaches a value of 100, an alert is sent, which can be seen by the same number of expected and sent alerts. However, to provide evidence for the limitation previously mentioned when a transformation *node* is added to the system, we purposely made *Device 3* fail, which in turn is executing the *heart_rate node*. This event occurs at approx. 160 seconds since the experiment has started.

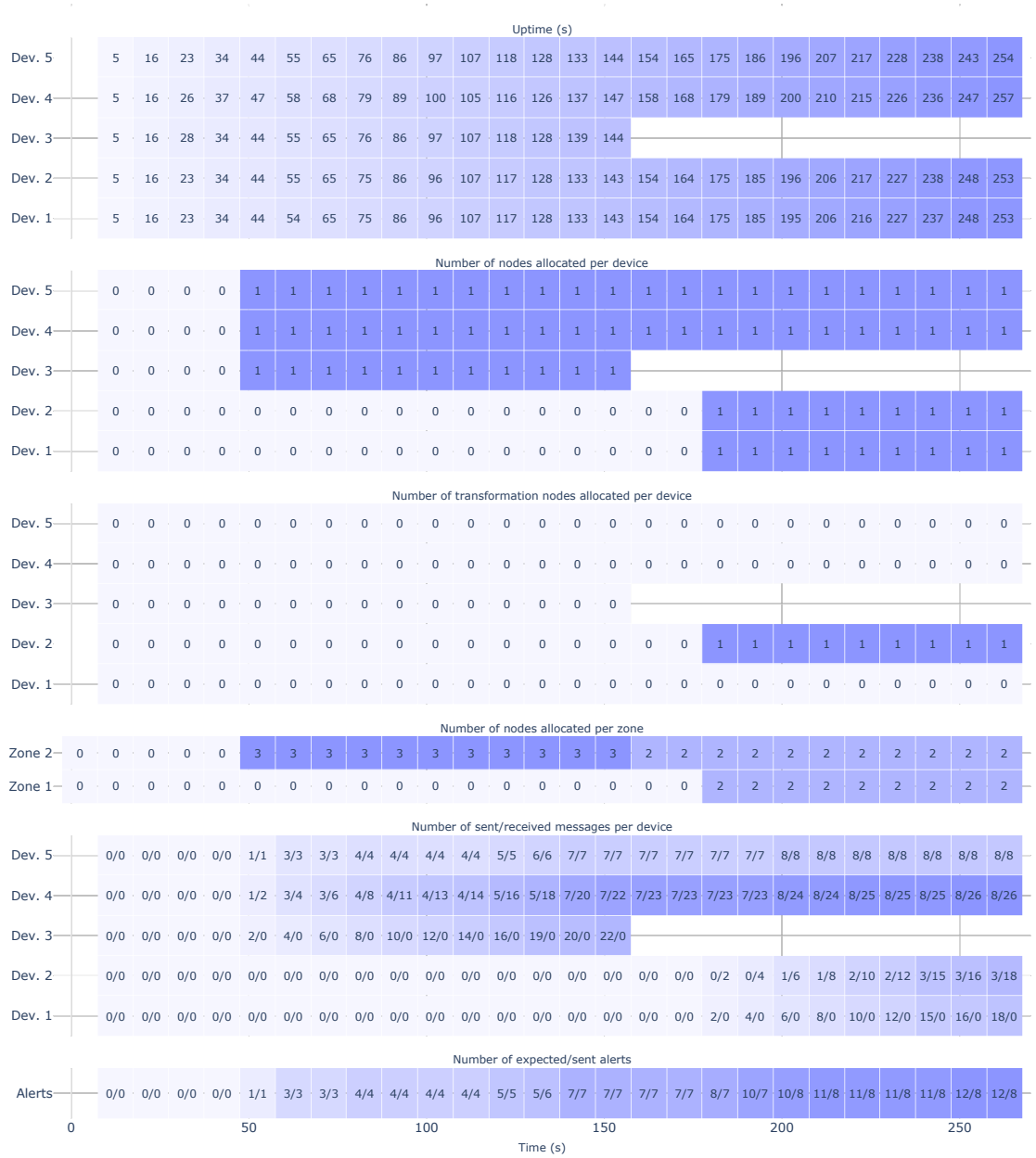

Figure 6.18: ES2-L2 measurements.

When *Device 3* fails, the system needs to re-orchestrate, and the only possible way it finds is to use devices in both zones and consequently transform data before changing zones. By analyzing the number of messages exchanged per device it is possible to see that, when the transformation *node* is inserted, the number of messages received by the *if node* — *Device 4* — does not match the number of messages generated by the *heart_rate node* — *Device 1*. This happens because the transformation *node* aggregates several messages and only calculates and sends a mean value of them.

After the re-orchestration, the number of expected and sent alerts graph shows that the number

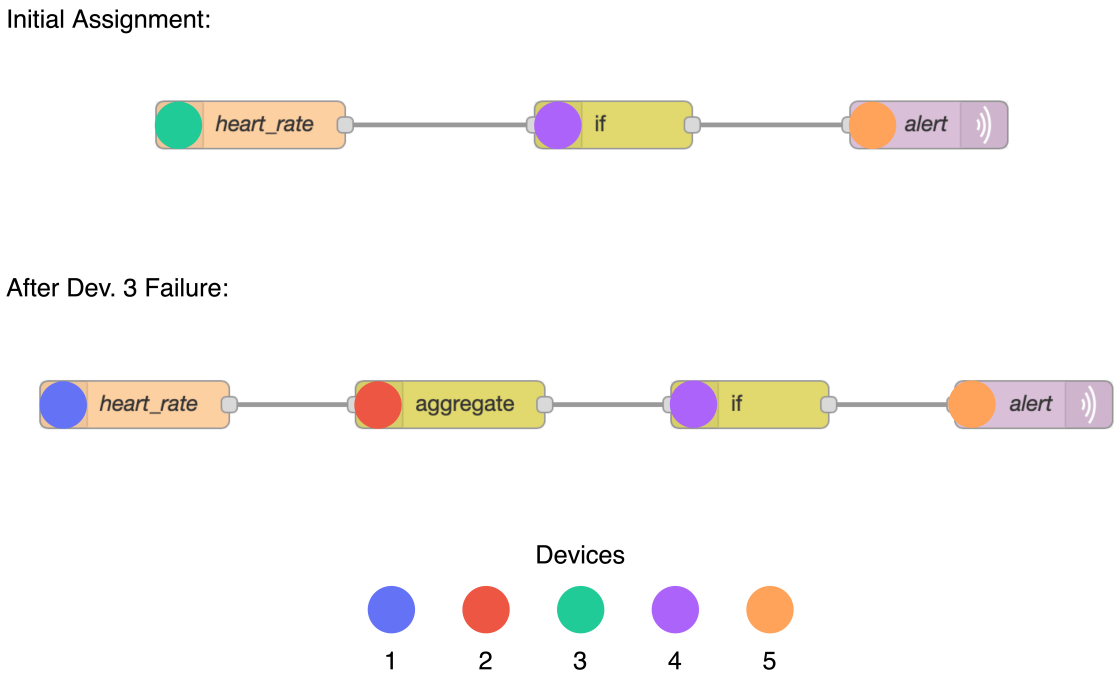

Figure 6.19: ES2-L2 *node* assignment.

of expected alerts does not match the number of actual alerts sent. This means that the values measured by the *heart_rate node* sometimes exceed the value of 100. However, the same does not happen with the mean value calculated by the *aggregation node*. This leads the system to miss some critical alert messages to be triggered.

The limitation verified with this experiment does no invalidate our solution since the system still ensures the privacy of the data when crossing zones. However, it fails to ensure the critical behavior of this scenario.

6.4 Replication Package

To allow the full replication of the experiments presented in this chapter, we have built and published an experiments replication package [50] with the following content:

Experiments folder: With one folder for each one of the experiments performed, where we have included files with the data collected and the Node-RED *flows* used.

Source code folder: With the source code of the modified version of Node-RED and the MicroPython firmware for the edge devices. For each one of the experiments we have included *docker-compose* files.

6.5 Desiderata Revisited

By the analysis of these results, it is possible to conclude the following regarding the *Desiderata* defined in Section 4.3 (p. 29):

D1 Automatically protect privacy-sensitive data: was fulfilled by preventing raw privacy data from being sent to potentially unsafe zones. Moreover, to make use of the computational power of the available devices in the IoT network, a mechanism was developed that automatically introduces transformation *nodes* whenever privacy-sensitive data is sent to untrusted devices to be processed.

D2 Generate privacy-oriented *node* assignment: was achieved by introducing a new value to the heuristic in the calculation of a *node* assignment score and by improving the assignment algorithm itself. Our assignment algorithm can prioritize the allocation of privacy-sensitive *nodes* to devices in a way that it can reduce the total number of zones crossed by private data in the system.

D3 Validate devices' identity on the network: no advancements were made towards this requirement.

6.6 Research Questions Revisited

The research questions defined in Section 4.5 (p. 29) guided the experiments performed to validate our implementation. Thus, we re-visit them and provide answers for each one of them by linking the experiments to it:

RQ1: How can we guarantee that privacy-sensitive data without any type of transformation is only processed by trusted devices?

To prevent private data from being generated or processed in possible unsafe devices, we introduced the concept of zones to the devices. Devices' zones are virtual barriers that separate trustworthy devices from devices that may be unsafe to receive and handle potential privacy-sensitive data. By introducing these boundaries in the system, we can ensure that no private information in raw mode ends up in unsafe zones.

During most of the experiments performed, it is possible to verify this behavior in the system. When the orchestrator assigns *nodes* to the available devices, it ensures that raw data can only be dealt with in devices that ate in the same zone as where the data was generated. Moreover, as it can be seen in the analysis of experiment **ES1-SC2** (*cf.* Section 6.3.1.2, p. 50), when privacy-sensitive data needs to be sent to a *node* that can only be executed in a device outside of the zone where the data flows, and if the system cannot guarantee the privacy of the data then the system won't assign the nodes to the devices and the system will trigger an error.

By the analysis of the experiments performed, we can conclude that the introduction of zones boundaries to the system guarantees that privacy-sensitive data without any type of transformation is only processed in devices in the same zone without compromising the private data.

RQ2: How can we ensure the privacy of data but at the same time explore the most of the computational power available in the network?

With this question, we intended to make use of most of the computational capabilities of edge devices in the network, but at the same time ensure the privacy of data when being sent from one zone to another. To achieve this, the orchestrator analyses the generated *node* assignment and proceeds to verify where privacy-sensitive data crosses the zones boundaries defined for the devices. When this is verified, the orchestrator introduces a transformation *node* that processes and changes the private data so that it can be handled outside the zone where it is originated from without compromising its privacy. This transformation *node* could perform, for example, one of the following actions to data: (1) aggregate, (2) encrypt, or (3) anonymize. For experimental purposes, we only implemented the possibility for data to be aggregated before changing zones.

In the analysis of some of the experiments, we can see the transformation *nodes* being inserted into the system's flow. **ES1-SC1** (*cf.* Section 6.3.1.1, p. 49) is the simplest experiment where this improvement can be analyzed. Since data needed to be sent from one zone to another, the orchestrator correctly inserted a transformation *node* which was executed over the private data before it left the zone where it was produced. This functionality was then further analyzed in experiment **ES1-A** (*cf.* Section 6.3.2.1, p. 51). In this experiment, we analyzed how the system would react if devices that can execute transformation *nodes* started to disappear. As expected and by the analysis of the results, it is possible to conclude that the orchestrator always tries to find a way to execute the transformation *nodes*, and when it cannot succeed, it still ensures that no private data ends up in unsafe locations without first being protected.

To sum up, the answer to this research question, the introduction of transformation *nodes* indeed guarantees the privacy of sensitive data while at the same time enabling the *flow* to be executed in different zones, making use of the computational power of the edge devices.

RQ3: Can we provide a way to prioritize the allocation of sensitive *nodes* to safer locations?

To allow the orchestrator to prioritize the assignment of privacy-sensitive *nodes* to trusted devices without compromising the performance of the devices, we made some improvements to the *node* assignment algorithm. The score calculation of a *node* assignment was changed to take into account the number of times privacy-sensitive data crosses zones. By doing this, we intended the orchestrator to find a better assignment that reduced the number of transformation *nodes* that had to be inserted in the system but at the same time spread the tasks among the available devices.

The changes performed in the *node* assignment algorithm were analyzed in experiments **ES1-B** (*cf.* Section 6.3.2.2, p. 52) and **ES1-C** (*cf.* Section 6.3.2.3, p. 53). In **ES1-B**, we show that our assignment algorithm gives more preference to execute the *nodes* in the same zone to reduce the number of zones crossed by private data, while in **ES1-C**, we show that by changing the factor values for the heuristics, we can easily adapt the orchestration priorities when assigning *nodes* to devices. With experiment **ES1-D** (*cf.* Section 6.3.2.4, p. 54) we proceeded to further analyze the performance of our assignment algorithm with more devices in the system. It was possible to verify that the algorithm tries to reduce the risk of private data being handled in unsafer locations by executing all the *nodes* in the same zone. However, it also takes into consideration the importance of an equal spread of tasks among the devices by choosing the zone with more available devices.

By observing the results of the experiments, we can conclude that our solution indeed provides a way for the orchestrator to prioritize the assignment of privacy-sensitive *nodes* to trusted devices.

After answering each one of the decomposed research questions, we can then provide an answer to the main research question of this work:

How can we improve the NoRDOr system to automatically guarantee a safe flow of privacy-sensitive data?

Considering the result of all experiments and the answers to the decomposed research questions, we can then conclude that our implementation can facilitate privacy during the flow of privacy-sensitive data by: (1) ensuring raw private data only flows in devices on the same zone, (2) protect privacy-sensitive data by transforming it before being sent to untrusted devices, and (3) prioritizing the execution of privacy-sensitive *nodes* in the same zone where data is collected.

6.7 Summary

In this chapter, we present and discuss the experiments performed to evaluate and validate our solution.

To evaluate and validate our implementation, we have defined two experimental scenarios in Section 6.1 (p. 41). For the first scenario, we chose a simple Node-RED *flow* which was used to verify the basic functionality and some behaviors of our solution. The second scenario used represents a real-world scenario that was based on Ambient Assisted Living systems. With such scenario, we wanted to validate our solution with a more complex *flow* that is more close to reality.

In Section 6.2 (p. 43), we have defined several virtual experiments that were performed for both scenarios with the objective of exploring and evaluating our solution. For all of the experiments, we collect metrics from the simulated devices, thus translated into charts to better analyze and discuss the results, which are present in Section 6.3 (p. 49). For some of the experiments, we introduce failures to the devices so that we can analyze how the system reacts to certain changes in

the system. During the experimental process, we also explore and analyze some of the limitations described in Section 5.5 (p. 39). Section 6.4 (p. 63) presents the developed replication package which can be used to rerun the experiments mentioned.

The results obtained from the experiments are then used to revisit the defined *desiderata* in Section 6.5 (p. 64) and to answer the research questions in Section 4.5 (p. 29).

Chapter 7

Conclusions

This chapter presents an overview of this dissertation and identifies future improvements for our work. Section 7.1 outlines the main conclusions regarding the work developed. In Section 7.2 (p. 71), we introduce the contributions that resulted from this dissertation. We then mention the main difficulties faced during the development of our solution in Section 7.3 (p. 71). Lastly, Section 7.4 (p. 71) presents possible future improvements for the limitations found in our work as well as other research directions.

7.1 Conclusions

With the growth of the number of Internet-of-Things devices and the increase of the computational capabilities of even the smaller devices, the Fog and Edge computing paradigms began to emerge. The computation and storage of the collected data were moved to the edge of the network, thus closer to the users. However, this comes with some drawbacks relative to the privacy of data. Since edge devices have fewer capabilities and energy constraints and many of these devices generate, process, and exchange privacy-sensitive data, they are appealing targets for attacks that aim to get access to private information.

During the analysis of state of the art, we found several platforms that allowed the orchestration of distributed IoT systems leveraging the computational capabilities of edge and fog devices. However, none of the platforms found have implemented mechanisms to ensure the privacy of data that flows in the system, often being left for future work. We then proceeded to perform a literature review on current methods used in IoT systems that address privacy and security concerns over the data exchanged in them.

The developed solution addresses the problems identified in the literature review by expanding an open-source Node-RED Distributed Orchestrator (NoRDOr) [19] with mechanisms that can automatically guarantee the privacy of data when generating, processing, and exchanging safety-critical data, as initially set by our main research question:

> *How can we improve the NoRDOr system to automatically guarantee a safe flow of privacy-sensitive data?*

The implementation started off by introducing zones to the system, which can be understood as virtual barriers that separate trustworthy devices from devices that may be unsafe to receive private data. By defining these boundaries, we were able to ensure that no raw private data ended up in a different zone from where it was produced, which could represent a risk for the privacy-sensitive data. However, by only preventing private data from reaching untrusted devices, the system would not be able to use most of the computational capabilities provided by the available devices in the IoT network. To leverage the computation capabilities of edge devices but at the same time facilitate privacy of data, the orchestrator introduces new *nodes* to the *flow*, which transform the private data before it leaves the respective zone.

In addition to protecting privacy-sensitive data exchanged by the devices on the network, the system must also generate privacy-oriented allocation of *nodes* to minimize the risk of private data being processed in possible unsafe locations. Hence, a new assignment heuristic was introduced that aims to reduce the number of times privacy-sensitive data crosses privacy boundaries.

The experiments presented in Chapter 6 (p. 41) were made to validate and evaluate our solution by assessing to which degree the requirements of the *desiderata* (*cf.* Section 4.3, p. 29) were fulfilled and by answering the raised research questions (*cf.* Section 4.5, p. 29). In these experiments, we have collected metrics from the simulated devices, which were then translated into charts to better assess the system behavior. During the experimental process, we have performed sanity checks which allowed us to confirm that our system could comply with privacy restrictions in the most basic scenarios, experimental tasks where we analyzed the system in more complex scenarios, and also experiments to explore some of the known limitations mentioned in Section 5.5 (p. 39).

Considering the results of the performed experiments, by introducing the concept of zones to the devices in our solution, it is possible to guarantee that no raw privacy-sensitive data ends up in untrusted devices — answering **RQ1** — and by adding transformation *nodes* before privacy-sensitive data leaves a zone, we allow the system to make use of most of the computational capabilities of the available devices without compromising the security of private data, thus answering **RQ2**. Moreover, **RQ3** is also answered, since our solution can prioritize the allocation of privacy-sensitive *nodes* to safer zones by the addition of a new heuristic that counts the number of zones crossed by private data, thus improving the assignment algorithm. Furthermore, we can similarly conclude that *desiderata* **D1** and **D2** were fulfilled, and no advancements were made towards the requirement **D3**.

In conclusion, with the results of the experiments and the answers for the decomposed research questions we can conclude that our implementation can automatically facilitate a safe flow of privacy-sensitive data in a distributed IoT system by defining zones for the devices and transforming private data whenever it needs to be sent to a different zone from where it was generated, answering our main research question. However, we recognize that there are some limitations for our solution, thus we believe our work is a good starting point for future work on the development of privacy-oriented solutions to task orchestration system.

7.2 Contributions

The work developed during this dissertation resulted in the following contributions:

The performed Literature Review: The current state of the art of systems that orchestrate computational tasks to devices in an IoT network and privacy and security mechanisms that can be used in these types of systems was analyzed.

The developed solution: We developed a solution based on the NoRDOr platform that ensures the privacy of data throughout the system.

7.3 Difficulties

During the development of our solution, we came across some difficulties that slowed the implementation process.

Understanding and experimenting with the system where we based the development of our solution took more time than expected due to the complexity of the NoRDOr system and some lack of project organization. Moreover, some things did not comply with the system's expected behavior, such as the fact that a device would still run a previously assigned *node* even if, in a re-orchestration, it would not be used to run any *node*. Thus, after the re-orchestration, the device was unexpectedly executing a *node* re-assigned to another device. Only after fixing the system was it possible to further validate our improvements.

Running the experiments in both scenarios also presented some difficulties since we had to perform some changes in certain experiments to validate specific behaviors, which led us to rebuild and restart the system several times.

7.4 Future Work

The mechanisms developed during the course of this dissertation solved some of the issues highlighted in Section 4.2 (p. 28). However, the implementation contains several limitations (*cf.* Section 5.5, p. 39) and introduces problems that require further research. As such, we see some improvements that could be implemented in future work, namely:

Implement new transformation nodes: Some limitations of our solution can be verified when inserting transformation *nodes* into certain *flows*. The orchestrator can successfully deploy the *flow* to the available devices in the network and ensure the privacy constraints of private data. However, when introducing transformation *nodes* to the system, the expected behavior of the *flow* can change since some *nodes* start to receive messages with a different structure which they are not used to. Moreover, as seen in experiment **ES2-L2** (*cf.* Section 6.3.4.2, p. 60), for some flows the aggregation of messages and the calculation of an average value leads the system to miss sending alerts when critical values are higher than a certain threshold. Our implementation lacks on a way to specify how privacy-sensitive data should be treated when being transformed. Nevertheless, a possible solution would be to allow the users to define which transformations can be performed in the data collected when inserting the *flow* into the Node-RED platform. For experimental purposes, our solution only implemented a transformation *node* which allowed the aggregation of messages. However, it is possible to implement other types of transformation *nodes* — such as encryption or anonymization — in future development.

Improve zones specification: Another limitation derived from our implementation comes from the fact that devices announce their respective zone to the orchestrator and the system does not validate the veracity of the devices' announcements. This can introduce security flaws into the system since malicious devices can announce themselves in certain zones just to get access to the information without any type of privacy protection. A welcome change that could prevent this from happening could be by introducing another layer of configuration in the Node-RED where the person responsible for the IoT network could specify the correct zones for each one of the devices in the system. Moreover, it is not clear if whether the users or the devices should be responsible for defining the zones. By taking into account the experiment **ES2-B** (*cf.* Section 6.3.3.2, p. 58), an elderly person who leaves the house and carries devices with him, potentially skips zones and it is not trivial for this to be foreseen in advance. With these being said, further research on this topic is still advised.

Alarmist scenarios: As seen in experiment **ES1-SC2** (*cf.* Section 6.3.1.2, p. 50) when the system cannot ensure the transformation of privacy-sensitive *nodes*, the system triggers an error that is currently being presented in the console. Exploiting these types of error alerts can be considered a good starting point to improve our solution. From this, we envision possible alarmist scenarios that can be implemented: (1) alert the user when there are few devices capable of executing transformation *nodes*, (2) alert the user when there are no available devices capable of ensuring the transformation of privacy-sensitive data, and (3) alert the user when there are no available devices to transform data but still allow the orchestration of the *flow* without guaranteeing the privacy of data. This last scenario suggestion may be used for systems where the execution of the *flow* is more important than ensuring the privacy of data. However, whenever devices capable of transforming data are available, the system

still wants to ensure the privacy of data in the IoT network without compromising the *nodes*' correct execution.

Perform experiments with physical devices: The experiments defined to validate and evaluate our implementation were only performed in virtual devices since the performance of the inserted improvements in the NoRDOr system were not likely to change in physical devices, compared to simulated ones. Although, the performance analysis of this privacy-oriented orchestration system with physical devices is still doable.

Most of the limitations presented here have relevant literature that can be used to address them to some degree. As an example, considering the deviations of normal service which can happen as a result of our approach, these can, at some degree, be addressed using fault-tolerance [12, 9, 10, 61] and self-healing [22, 25, 28, 60] techniques.

Appendix A

ES1 Sequence Diagram

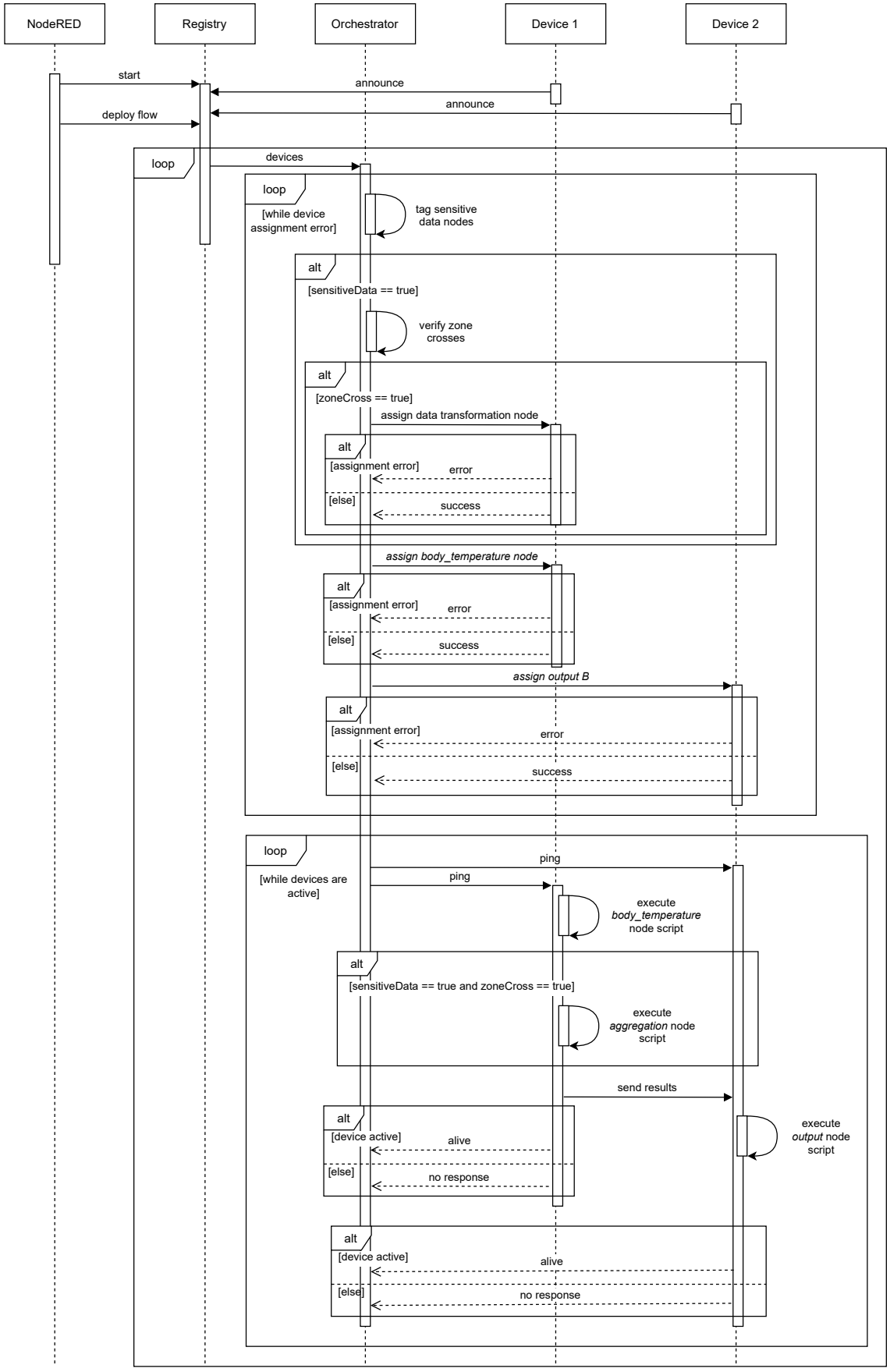

Figure A.1: ES1-SC1 sequence diagram.

References

[1] Mohammad Aazam, Sherali Zeadally, and Khaled A Harras. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(10):4674–4682, 2018.

[2] Alauddin Al-Omary, Ali Othman, Haider M AlSabbagh, and Hussain Al-Rizzo. Survey of hardware-based security support for iot/cps systems. *KnE Engineering*, pages 52–70, 2018.

[3] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry. Iot architecture challenges and issues: Lack of standardization. In *2016 Future Technologies Conference (FTC)*, pages 731–738, 2016.

[4] Mohammad S Ansari, Saeed H Alsamhi, Yuansong Qiao, Yuhang Ye, and Brian Lee. Security of distributed intelligence in edge computing: Threats and countermeasures. In *The Cloud-to-Thing Continuum*, pages 95–122. Palgrave Macmillan, Cham, 2020.

[5] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[6] Gamal Attiya and Yskandar Hamam. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. *Journal of Parallel and Distributed Computing*, 66(10):1259–1266, 2006.

[7] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140, 2017.

[9] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Fundamental Concepts of Dependability. *Technical Report Seriesuniversity of Newcastle Upon Tyne Computing Science*, 1145(010028):7–12, 2001.

[10] Algirdas Avižienis, Jean Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[11] Michael Blackstock and Rodger Lea. Toward a distributed data flow platform for the web of things (distributed node-red). In *Proceedings of the 5th International Workshop on Web of Things*, WoT '14, page 34–39, New York, NY, USA, 2014. Association for Computing Machinery.

[12] Carlo Alberto Boano, Kay Uwe Römer, Roderick Bloem, Klaus Witrisal, Marcel Carsten Baunach, and Martin Horn. Dependability for the internet of things: From dependable networking in harsh environments to a holistic view on dependability. *Elektrotechnik und Informationstechnik*, 133(7):304–309, 11 2016.

[13] Sousa Tiago Boldt. Dataflow programming: Concept languages and applications. In *Proc. 2012 7th Doctoral Symposium in Informatics Engineering*, pages 323–334, 2012.

[14] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh 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*, pages 13–16, 2012.

[15] Rajkumar Buyya and Amir Vahid Dastjerdi. *Internet of Things: Principles and paradigms*. Elsevier, 2016.

[16] Rajkumar Buyya and Amir Vahid Dastjerdi. *Internet of Things: Principles and Paradigms*. Elsevier, 2016.

[17] Davide Calvaresi, Daniel Cesarini, Paolo Sernani, Mauro Marinoni, Aldo Franco Dragoni, and Arnon Sturm. Exploring the ambient assisted living domain: a systematic review. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):239–257, 2017.

[18] L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu. A decentralized and trusted edge computing platform for internet of things. *IEEE Internet of Things Journal*, 7(5):3910–3922, 2020.

[19] Ana Margarida Oliveira Pinheiro da Silva. Orchestration for automatic decentralization in visually-defined iot. Master's thesis, Faculty of Engineering, University of Porto, 2020.

[20] Lucas Santos Dalenogare, Guilherme Brittes Benitez, Néstor Fabián Ayala, and Alejandro Germán Frank. The expected contribution of industry 4.0 technologies for industrial performance. *International Journal of Production Economics*, 204:383 – 394, 2018.

[21] João Pedro Dias, João Pascoal Faria, and Hugo Sereno Ferreira. A reactive and model-based approach for developing internet-of-things systems. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 276–281, September 2018.

[22] João Pedro Dias, Bruno Lima, João Pascoal Faria, André Restivo, and Hugo Sereno Ferreira. Visual self-healing modelling for reliable internet-of-things systems. In *International Conference on Computational Science*, pages 357–370. Springer, 2020.

[23] João Pedro Dias, José Pedro Pinto, and José Magalhães Cruz. A Hands-on Approach on Botnets for Behavior Exploration. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*, pages 463–469. SCITEPRESS - Science and Technology Publications, 2017.

[24] João Pedro Dias, Hugo Sereno Ferreira, and Ângelo Martins. A blockchain-based scheme for access control in e-health scenarios. In Ana Maria Madureira, Ajith Abraham, Niketa Gandhi, Catarina Silva, and Mário Antunes, editors, *Proceedings of the Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018)*, pages 238–247, Cham, 2020. Springer International Publishing.

[25] Joao Pedro Dias, Tiago Boldt Sousa, André Restivo, and Hugo Sereno Ferreira. A pattern-language for self-healing internet-of-things systems. In *Proceedings of the 25th European Conference on Pattern Languages of Programs*, EuroPLop '20, New York, NY, USA, 2020. Association for Computing Machinery.

[26] João Pedro Dias, Hugo Sereno Ferreira, and Tiago Boldt Sousa. Testing and deployment patterns for the internet-of-things. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, EuroPLop '19, New York, NY, USA, 2019. Association for Computing Machinery.

[27] João Pedro Dias, André Lago, and Hugo Sereno Ferreira. Conversational interface for managing non-trivial internet-of-things systems. In *Proceedings of the 20th International Conference on Computational Science (ICCS)*, pages 27–36. Springer, 2020.

[28] João Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Empowering visual internet-of-things mashups with self-healing capabilities. In *2021 IEEE/ACM 2nd International Workshop on Software Engineering Research Practices for the Internet of Things (SERP4IoT)*, 2021.

[29] João Pedro Dias, Ângelo Martins, and Hugo Sereno Ferreira. A blockchain-based approach for access control in ehealth scenarios. *Journal of Information Assurance and Security*, 13:125–136, 2018.

[30] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier. The internet of things for ambient assisted living. In *2010 Seventh International Conference on Information Technology: New Generations*, pages 804–809, 2010.

[31] Ashutosh Dhar Dwivedi, Gautam Srivastava, Shalini Dhar, and Rajani Singh. A decentralized privacy-preserving healthcare blockchain for iot. *Sensors*, 19(2), 2019.

[32] PJ Escamilla-Ambrosio, A Rodríguez-Mota, E Aguirre-Anaya, R Acosta-Bermejo, and M Salinas-Rosales. Distributing computing in the internet of things: cloud, fog and edge computing overview. In *NEO 2016*, pages 87–115. Springer, 2018.

[33] João Pascoal Faria, Bruno Lima, Tiago Boldt Sousa, and Angelo Martins. A testing and certification methodology for an ambient-assisted living ecosystem. In *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, pages 585–589. IEEE, 2013.

[34] Hugo Sereno Ferreira, Tiago Boldt Sousa, and Angelo Martins. Scalable integration of multiple health sensor data for observing medical patterns. In *International Conference on Cooperative Design, Visualization and Engineering*, pages 78–84. Springer, 2012.

[35] Alejandro Germán Frank, Lucas Santos Dalenogare, and Néstor Fabián Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210:15–26, 2019.

[36] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung. Developing iot applications in the fog: A distributed dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 155–162, 2015.

[37] Nam Ky Giang, Rodger Lea, Michael Blackstock, and Victor CM Leung. Fog at the edge: Experiences building an edge computing platform. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 9–16. IEEE, 2018.

[38] Alasdair Gilchrist. *Industry 4.0: the industrial internet of things*. Springer, 2016.

[39] Alex Glikson, Stefan Nastic, and Schahram Dustdar. Deviceless edge computing: extending serverless computing to the edge of the network. In *Proceedings of the 10th ACM International Systems and Storage Conference*, pages 1–1, 2017.

[40] Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Schahram Dustdar, Ognjen Scekic, Thomas Rausch, Stefan Nastic, Sasko Ristov, and Thomas Fahringer. A deviceless edge computing approach for streaming iot applications. *IEEE Internet Computing*, 23(1):37–45, 2019.

[41] Zijiang Hao, Ed Novak, Shanhe Yi, and Qun Li. Challenges and Software Architecture for Fog Computing. *IEEE Internet Computing*, 21(2):44–53, 2017.

[42] Yan Huo, Chun Meng, Ruinian Li, and Tao Jing. An overview of privacy preserving schemes for industrial internet of things. *China Communications*, 17(10):1–18, 2020.

[43] Antonio J Jara, Miguel A Zamora, and Antonio FG Skarmeta. An internet of things–based personal device for diabetes therapy management in ambient assisted living (aal). *Personal and Ubiquitous Computing*, 15(4):431–440, 2011.

[44] Qinma Kang, Hong He, and Jun Wei. An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of Parallel and Distributed Computing*, 73(8):1106–1115, 2013.

[45] AL Koray. Ergonomic conditions assessment of a bus factory. *ICONTECH INTERNATIONAL JOURNAL*, 4(2):35–47, 2020.

[46] André Sousa Lago, João Pedro Dias, and Hugo Sereno Ferreira. Managing non-trivial internet-of-things systems with conversational assistants: A prototype and a feasibility experiment. *Journal of Computational Science*, 51:101324, 2021.

[47] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari. A robust ecc-based provable secure authentication protocol with privacy preserving for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3599–3609, 2018.

[48] Huichen Lin and Neil W Bergmann. Iot privacy and security challenges for smart home environments. *Information*, 7(3):44, 2016.

[49] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[50] Nuno Lopes. Privacy-oriented task orchestration system - experiments validation package. Available at https://doi.org/10.5281/zenodo.5042831, 2021.

[51] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake, et al. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(05):164, 2015.

[52] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

[53] Carsten Maple. Security and privacy in the internet of things. *Journal of Cyber Policy*, 2(2), 2017.

[54] Jianbing Ni, Kuan Zhang, Xiaodong Lin, and Xuemin Shen. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Communications Surveys & Tutorials*, 20(1):601–628, 2017.

[55] Joseph Noor, Hsiao-Yun Tseng, Luis Garcia, and Mani Srivastava. Ddflow: Visualized declarative programming for heterogeneous iot networks. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, page 172–177, New York, NY, USA, 2019. Association for Computing Machinery.

[56] Sebastian Pape and Kai Rannenberg. Applying privacy patterns to the internet of things' (iot) architecture. *Mobile Networks and Applications*, 24(3):925–933, Jun 2019.

[57] Donn B Parker. *Fighting computer crime: A new framework for protecting information*. John Wiley & Sons, Inc., 1998.

[58] Guilherme Vieira Pinto, João Pedro Dias, and Hugo Sereno Ferreira. Blockchain-based pki for crowdsourced iot sensor information. In Ana Maria Madureira, Ajith Abraham, Niketa Gandhi, Catarina Silva, and Mário Antunes, editors, *Proceedings of the Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018)*, pages 248–257, Cham, 2020. Springer International Publishing.

[59] Sandra Prescher, Alan K Bourke, Friedrich Koehler, Angelo Martins, Hugo Sereno Ferreira, Tiago Boldt Sousa, Rui Nuno Castro, António Santos, Marc Torrent, Sergi Gomis, et al. Ubiquitous ambient assisted living solution to promote safer independent living in older adults suffering from co-morbidity. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5118–5121. IEEE, 2012.

[60] Harald Psaier and Schahram Dustdar. A survey on self-healing systems: Approaches and systems. *Computing (Vienna/New York)*, 91(1):43–73, 2011.

[61] Antonio Ramadas, Gil Domingues, Joao Pedro Dias, Ademar Aguiar, and Hugo Sereno Ferreira. Patterns for Things that Fail. In *Proceedings of the 24th Conference on Pattern Languages of Programs*, PLoP '17. ACM - Association for Computing Machinery, 2017.

[62] Spyridon Samonas and David Coss. The cia strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, 10(3), 2014.

[63] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[64] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[65] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of things: The road ahead. *Computer Networks*, 76:146–164, 2015.

[66] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146 – 164, 2015.

[67] Margarida Silva, João Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Visually-defined real-time orchestration of iot systems. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS 2020, New York, NY, USA, 2020. Association for Computing Machinery.

[68] Margarida Silva, João Pedro Dias, André Restivo, and Hugo Sereno Ferreira. A review on visual programming for distributed computation in iot. In *Proceedings of the 21st International Conference on Computational Science (ICCS)*. Springer, 2021.

[69] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, 2018.

[70] A. F. Skarmeta, J. L. Hernández-Ramos, and M. V. Moreno. A decentralized approach for security and privacy challenges in the internet of things. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 67–72, 2014.

[71] Danny Soares, João Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Programming iot-spaces: A user-survey on home automation rules. In *Proceedings of the 21st International Conference on Computational Science (ICCS)*. Springer, 2021.

[72] Tiago Boldt Sousa. Sensors, actuators and services: a distributed approach. In *Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity*, pages 161–166, 2013.

[73] Tiago Boldt Sousa and Angelo Martins. Monitor, control and process–an adaptive platform for ubiquitous computing. In *International Conference on Cooperative Design, Visualization and Engineering*, pages 47–50. Springer, 2013.

[74] Hong Sun, Vincenzo De Florio, Ning Gui, and Chris Blondia. Promises and challenges of ambient assisted living systems. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 1201–1207, 2009.

[75] M. S. Virat, S. M. Bindu, B. Aishwarya, B. N. Dhanush, and M. R. Kounte. Security and privacy challenges in internet of things. In *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 454–460, 2018.

[76] Merrill Warkentin and Craig Orgeron. Using the security triad to assess blockchain technology in public sector applications. *International Journal of Information Management*, 52:102090, 2020.

[77] Rolf H Weber. Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1):23–30, 2010.

[78] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang. Mutual privacy preserving k -means clustering in social participatory sensing. *IEEE Transactions on Industrial Informatics*, 13(4):2066–2076, 2017.

[79] Yang Yang. Multi-tier computing networks for intelligent iot. *Nature Electronics*, 2(1):4–5, 2019.