SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI *MEDITERRANEA* DI REGGIO CALABRIA

DIPARTIMENTO DI INGEGNERIA CIVILE, DELL'ENERGIA
DELL'AMBIENTALE E DEI MATERIALI (DICEAM)

DOTTORATO DI RICERCA IN
INGEGNERIA CIVILE, AMBIENTALE E DELLA SICUREZZA

S.S.D. ING-INF/05
XXXIII CICLO

OSMOTIC COMPUTING: SECURE AND DEPENDABLE MICROSERVICES ORCHESTRATION IN THE CLOUD-TO-THING CONTINUUM

DOTTORANDA:
MIHAELA-ALINA BUZACHIS

TUTOR:
Prof. Massimo Villari

CO-TUTOR:
Prof. Giovanni Pioggia

COORDINATORE:
Prof. Felice Arena

REGGIO CALABRIA, FEBBRAIO 2021

List of Publications

Articles in Refereed Journals

(IJ.1) **Buzachis A.**, Galletta A., Celesti A, Jiafu Wan, Fazio M., Villari M. - "A Map-Reduce Approach for the Dijkstra Algorithm in SDN over Osmotic Computing Systems". *International Journal on Parallel Programming (IJPP).* [accepted]

(IJ.2) **Buzachis A.**, Celesti A., Galletta A., Jiafu Wan, Fazio M. – "Evaluating a Distributed Dijkstra Routing Algorithm in Hybrid Computing Environments for IoT Applications". *IEEE Transactions on Sustainable Computing*, Special Issue on Sustainability of Fog/Edge Computing Systems. [under revision]

(IJ.3) **Buzachis, A.**, Celesti, A., Galletta, A., Fazio, M., Fortino, G., & Villari, M. (2020). A multi-agent autonomous intersection management (MA-AIM) system for smart cities leveraging edge-of-things and Blockchain. *Inf. Sci.*, 522, 148-163.

(IJ.4) Summa, S., Tartarisco, G., Favetta, M., **Buzachis, A.**, Romano, A., Bernava, G., Sancesario, A., Vasco, G., Pioggia, G., Petrarca, M., Castelli, E., Bertini, E., & Schirinzi, T. (2020). Validation of low-cost system for gait assessment in children with ataxia. *Computer methods and programs in biomedicine*, 196, 105705 .

(IJ.5) Summa, S., Tartarisco, G., Favetta, M., **Buzachis, A.**, Romano, A., Bernava, G.M., Vasco, G., Pioggia, G., Petrarca, M., Castelli, E., Bertini, E., & Schirinzi, T. (2020). Spatio-temporal parameters of ataxia gait dataset obtained with the Kinect. *Data in Brief*, 32.

(IJ.6) **Buzachis A.**, Celesti A., Galletta A., Fazio M., Fortino G., and Villari M. - "An Edge Computing Based Post-Disaster Communication Network for Industrial Applications", *Journal of Computers & Electrical Engineering - VSI on Edge Intelligence in Industrial Applications (VSI-eiia).* [under review]

Articles in Refereed Conference Proceedings

(IC.1) **A. Buzachis**, A. Galletta, L. Carnevale, A. Celesti, M. Fazio and M. Villari, "Towards Osmotic Computing: Analyzing Overlay Network Solutions

to Optimize the Deployment of Container-Based Microservices in Fog, Edge and IoT Environments," *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, Washington, DC, 2018, pp. 1-10, doi: 10.1109/CFEC.2018.8358729.

(IC.2) **A. Buzachis**, G. M. Bernava, M. Busà, G. Pioggia and M. Villari, "Towards Osmotic Computing: Future Prospect for the Health Information Technology (HIT) Systems of ISASI-CNR (ME)," *2018 IEEE Symposium on Computers and Communications (ISCC)*, Natal, 2018, pp. 01255-01260, doi: 10.1109/ISCC.2018.8538714.

(IC.3) A. Celesti, **A. Buzachis**, A. Galletta, G. Fiumara, M. Fazio and M. Villari, "Analysis of a NoSQL Graph DBMS for a Hospital Social Network," *2018 IEEE Symposium on Computers and Communications (ISCC)*, Natal, 2018, pp. 01298-01303, doi: 10.1109/ISCC.2018.8538469.

(IC.4) **A. Buzachis**, A. Galletta, A. Celesti, L. Carnevale and M. Villari, "Towards Osmotic Computing: a Blue-Green Strategy for the Fast Re-Deployment of Microservices," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969621.

(IC.5) A. Celesti, **A. Buzachis**, A. Galletta, M. Fazio and M. Villari, "A NoSQL Graph Approach to Manage IoTaaS in Cloud/Edge Environments," *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, Barcelona, 2018, pp. 407-412, doi: 10.1109/FiCloud.2018.00065.

(IC.6) **Buzachis, A.**, Galletta, A., Celesti, A., & Villari, M. (2018). An Innovative MapReduce-Based Approach of Dijkstra's Algorithm for SDN Routing in Hybrid Cloud, Edge and IoT Scenarios. *In Service-Oriented and Cloud Computing* (pp. 185–198). Springer International Publishing.

(IC.7) **A. Buzachis**, G. M. Bernava, M. Busa, G. Pioggia and M. Villari, "Towards the Basic Principles of Osmotic Computing: A Closed-Loop Gamified Cognitive Rehabilitation Flow Model," *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, Philadelphia, PA, 2018, pp. 446-452, doi: 10.1109/CIC.2018.00067.

(IC.8) **A. Buzachis**, A. Celesti, A. Galletta, M. Fazio and M. Villari, "A Secure and Dependable Multi-Agent Autonomous Intersection Management (MA-AIM) System Leveraging Blockchain Facilities," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, 2018 pp. 226-231, doi: 10.1109/UCC-Companion.2018.00060.

(IC.9) **A. Buzachis** and M. Villari, "Basic Principles of Osmotic Computing: Secure and Dependable MicroElements (MELs) Orchestration Leveraging Blockchain Facilities," *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, 2018, pp. 47-52, doi: 10.1109/UCC-Companion.2018.00033.

(IC.10) **A. Buzachis**, A. Galletta, A. Celesti, M. Fazio and M. Villari, "Development of a Smart Metering Microservice Based on Fast Fourier Transform (FFT) for Edge/Internet of Things Environments," *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, Larnaca, Cyprus, 2019, pp. 1-6, doi: 10.1109/CFEC.2019.8733148.

(IC.11) **A. Buzachis**, A. Celesti, M. Fazio and M. Villari, "On the Design of a Blockchain-as-a-Service-Based Health Information Exchange (BaaS-HIE) System for Patient Monitoring," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969718.

(IC.12) **Buzachis, A.**, Filocamo, B., Fazio, M., Ruiz, J.A., Sotelo, M., & Villari, M. (2019). Distributed Priority Based Management of Road Intersections Using Blockchain. *2019 IEEE Symposium on Computers and Communications (ISCC)*, 1159-1164.

(IC.13) **A. Buzachis**, M. Fazio, A. Galletta, A. Celesti and M. Villari, "Infrastructureless IoT-as-a-Service for Public Safety and Disaster Response," *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, Istanbul, Turkey, 2019, pp. 133-140, doi: 10.1109/FiCloud.2019.00026.

(IC.14) **Buzachis, A.**, Fazio, M., Celesti, A., & Villari, M. (2019). Osmotic Flow Deployment Leveraging FaaS Capabilities. *In Internet and Distributed Computing Systems* (pp. 391–401). Springer International Publishing. 2019, Napoli, Italy.

(IC.15) **Buzachis, A.**, Fazio, M., Galletta, A., Celesti, A., & Villari, M. (2019). Intelligent IoT for Non-Intrusive Appliance Load Monitoring Infrastructures in Smart Cities. *AI&IoT@AI*IA*.

(IC.16) M. Fazio, **A. Buzachis**, A. Galletta, A. Celesti and M. Villari, "A proximity-based indoor navigation system tackling the COVID-19 social distancing measures," *2020 IEEE Symposium on Computers and Communications (ISCC)*, Rennes, France, 2020, pp. 1-6, doi: 10.1109/ISCC50000.2020.9219634.

(IC.17) **A. Buzachis**, D. Boruta, M. Villari, J. Spillner, “Modeling and Emulation of an Osmotic Computing Ecosystem using OsmoticToolkit”, emph2021

Australasian Computer Science Week Multiconference (ACSW '21)., Association for Computing Machinery, New York, NY, USA, Article 9, 1–9. doi: https://doi.org/10.1145/3437378.3444366

(IC.18) J. Spillner, P. Gkikopoulos, **A. Buzachis** and M. Villari, "Rule-Based Resource Matchmaking for Composite Application Deployments across IoT-Fog-Cloud Continuums," *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, Leicester, United Kingdom, 2020, pp. 336-341, doi: 10.1109/UCC48980.2020.00053.

Book Chapters

(BC.1) Taheri, Javid; Shuiguang Deng, (ed.): 'Edge Computing: Models, technologies and applications' (Computing, 2020) DOI: IET Digital Library, https://digital-library.theiet.org/content/books/pc/pbpc033e

Patents

(P.1) L. Carnevale, **A. Buzachis**, A. Galletta, C. Cincotta, F. Martella, and M. Villari, "Communication System based on Mesh Networks," November 22, 2018 (P4548IT00).

"*You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future. You have to trust in something – your gut, destiny, life, karma, whatever. Because believing that the dots will connect down the road will give you the confidence to follow your heart even when it leads you off the well worn path; and that will make all the difference.*"

Steve Jobs
(Stanford commencement speech, June 2005)

Abstract

Internet of Things (IoT) is a profound technology evolution incorporating billions of devices (e.g., sensors, RFIDs, smartphones, and wearables) owned by different organizations and people who are deploying and using them for pervasive digital services. Their number, capabilities, scope of use and data volume keep growing and changing rapidly, leading to higher complexity in IoT applications. Thus, new distributed computing paradigms, such as Edge Computing (EC) or IoT-Cloud Computing, have been investigated to extend IoT resources into centralized data centres (e.g., clouds) or at the edge of IoT systems (e.g., edge micro datacenters).

Among the most promising ones is Osmotic Computing (OC), motivated by the lack of a scalable, interoperable, configurable solution for delivering IoT applications in complex, heterogeneous and dynamic computing environments. OC looks at the opportunistic management of microservices (called MicroELements - MELs) to improve the Quality of Service (QoS) and networking management, interoperability, and efficiency of next-generation IoT applications. This thesis investigates how OC can be leveraged to achieve secure and dependable microservices orchestration in the Cloud-to-Thing (C2T) continuum where deployment and orchestration strategies depend on IoT applications' specific requirements and physical/virtual resources availability.

This contribution can be divided into four major parts. The first part overviews the C2T continuum in general and faces the resource management challenges. The second part details basic concepts, methodologies and key technologies behind OC and investigates how IoT applications into the C2T continuum can benefit from it. The third part showcases OsmoticToolkit, a cost-effective and flexible toolkit for OC ecosystems' from-scratch design and real-world applications emulation. Finally, the fourth part gives a more in-depth look into the application deployment strategies and presents a Rule-Based MatchMaker (RBMM) for supporting applications deployment in the C2T continuum.

Sommario

Internet of Things (IoT) è una radicale evoluzione tecnologica che incorpora miliardi di dispositivi (ad esempio, sensori, RFID, smartphone e dispositivi indossabili) che possono essere utilizzati da organizzazioni e/o persone per servizi digitali pervasivi. Il loro numero, le loro capacità, l'ambito di utilizzo e il volume di dati continuano a crescere e cambiare rapidamente. Questo comporta una maggiore complessità nelle applicazioni IoT. Così, nuovi paradigmi di calcolo distribuito, come Edge Computing (EC) o IoT-Cloud Computing, sono stati studiati per estendere le risorse dell'IoT in data center centralizzati (ad esempio, cloud) o ai margini dei sistemi di IoT (ad esempio, micro data center edge).

A tale scopo il paradigma più promettente è l'Osmotic Computing (OC), motivato dalla mancanza di una soluzione scalabile, interoperabile e configurabile per la fornitura di applicazioni IoT in ambienti di calcolo complessi, eterogenei e dinamici. OC si occupa della gestione opportunistica dei microservizi (chiamati MicroELements - MELs) per migliorare la gestione della Quality of Service (QoS) e della rete, l'interoperabilità e l'efficienza delle applicazioni IoT di nuova generazione. Questa tesi è incentrata sullo studio del paradigma OC e approfondisce i meccanismi che permettono di ottenere un'orchestrazione sicura e affidabile dei microservizi nel continuum Cloud-to-Thing (C2T), dove le strategie di implementazione e orchestrazione dipendono dai requisiti specifici delle applicazioni IoT e dalla disponibilità di risorse fisiche/virtuali.

Questo contributo può essere suddiviso in quattro parti principali. La prima parte descrive il continuum C2T in generale e affronta le sfide della gestione delle risorse. La seconda parte descrive in dettaglio i concetti di base, le metodologie e le tecnologie chiave alla base della OC e indaga su come le applicazioni IoT nel continuum C2T possono beneficiarne. La terza parte presenta OsmoticToolkit, un toolkit flessibile ed economico per la progettazione di ecosistemi osmotici da zero e l'emulazione di applicazioni reali. Infine, la quarta parte offre uno sguardo più approfondito sulle strategie di implementazione delle applicazioni e presenta un Rule-Based MatchMaker (RBMM) per supportare la distribuzione efficiente delle applicazioni nel continuum C2T.

Acknowledgments

My Ph.D. has been a fantastic journey that constantly challenged me to push my knowledge and capabilities boundaries. It was a journey that gave me many memories that I will always cherish but would remain incomplete without acknowledging the people who made it possible. I was lucky to work with and make friends with several people who helped me become a researcher, and here I would like to thank them all for their endless love, support and understanding.

First and foremost, I would like to express my deepest appreciation to my advisor, Prof. Massimo Villari, for his support and guidance throughout my doctoral studies. I am deeply impressed by his penchant for innovative thinking and brainstorming ideas with his students, as well as his tireless workaholism and strong self-discipline.

Sincere thanks are due to my co-supervisor, Dr. Giovanni Pioggia, with whom I also had the pleasure of collaborating. I extend my thanks to the National Research Council (CNR-IRIB) staff from Messina (Italy), especially to Dr. Massimo Bernava, who allowed me to collaborate with the Ospedale Bambino Gesù staff from Rome (Italy) and who made me appreciate the biomedical research.

Special thanks go to Prof. Josef Spillner for hosting me as a visiting researcher at the Service Prototyping Lab (SPLAB), Zürich University of Applied Sciences (ZHAW). Thank you for your support and incredible professional vision. My experience there was truly remarkable.

I would also like to thank my committee members, Drs. Javid Taheri and Jörg Domaschka for their constructive feedback and valuable insights that made it possible for me to improve this dissertation's overall quality.

My Ph.D. journey would not have been possible without my family. I would like to thank my mother for her love, support, and encouragement. Special thanks go to my boyfriend, Adriele Magistro, for his continuous encouragement and support that helped me overcome many obstacles.

Alina Buzachis
(Zürich, Switzerland, February 2021)

Contents

1 Introduction

1.1 Context and Motivation

Current advances in the cost, performance, and energy efficiency of Internet of Things (IoT) devices (e.g., sensors and gateways), network technologies (e.g., 5G, Wi-Fi, RFID/NFC, Bluetooth, IEEE 802.15.4), and Cloud Computing (CC) has triggered a ubiquitous connection among people, devices and services anywhere, anytime and anyhow. IoT is predicted to reach 500 billion devices connected to the Internet by 2030 [1], while the global mobile traffic is expected to increase sevenfold by 2021 [2]. This ubiquity has driven unprecedented growth in data management, forcing us to rethink how to efficiently tackle massive volume, velocity, and variety of the generated big data [3, 4].

A plethora of new heterogeneous connected "things", called in general Internet of Everything (IoE) [5], are expected to be deployed in the next years. IoE expands the IoT concept by adding links to data, people (e.g., Internet of People) and processes (Industrial Internet) [6]. IoE is expected to fuel the evolution of traditional services and deploy novel categories of services into several application domains such as healthcare, cities, utility grids, transportation, agriculture, industry 4.0, and disaster management. In this panorama, the need for investigating on-the-fly computation over the IoT data streams is ever more critical.

IoT is generally characterized by real-world small things, widely distributed, with limited storage and processing capacity, which involve concerns regarding reliability, performance, security, and privacy. On the other hand, CC has virtually unlimited storage and processing power capabilities, is a much more mature technology, and partially solves most IoT issues. Thus, a novel IT paradigm in which cloud and IoT are two complementary technologies are integrated is called IoT cloud.

Traditional IoT cloud infrastructures tend to be inefficient due to the following challenges. First, the traditional solutions have mainly relied on centralized communication models, e.g., central cloud, for IoT service operations, making it difficult to scale when IoT networks become more widespread [7]. Further, traditional IoT cloud systems mandate trust to a third party, e.g., a cloud provider, for IoT data processing, which raises data privacy concerns. Finally, the centralized network infrastructure results in more high communication latency and power consumption for IoT devices due to long data transmission, limiting large-scale deployments in practical scenarios [8].

A solution implies decoupling the network delay from the computation time

for processing big data by bringing computations towards the edge of networks (i.e., closer to the end-users who process and consume data). This drove Edge Computing (EC) and Fog Computing (FC) 's emergence, an essential endeavour to sustain such applications' growth.

EC allows performing computations outside the boundaries of data centres. Many approaches have already leveraged on some form of EC to improve the latency perceived by end-users, such as Content Delivery Networks (CDNs) [9], or tapping into resources of client devices [10], among others. This has motivated the emergence of different architecture proposals for taking advantage of EC. In particular, Cisco has proposed the model of FC [11] which strives at increasing the overall performance of IoT applications by placing servers (and network equipment with computing capacity) close to sensors. Fog servers can then pre-process data enabling timely reaction to variations on the sensed environment and filter the relevant information that must be propagated towards cloud infrastructures for further processing. The Open Fog Consortium defines FC as *"A horizontal system-level architecture that distributes computing, storage, control, and networking functions closer to the users along a Cloud-to-Thing continuum"* [12]. Cloud-to-Thing (C2T) continuum has emerged as a revolutionary paradigm for developing infrastructures that extend beyond centralized data centres from the cloud to the edge.

Unlike horizontal system-level architecture, a vertical platform may provide strong support for a single application type (silo). It does not account for platform-to-platform interaction in other vertically focused platforms. A vertical platform promotes siloed applications.

On the other hand, EC/FC proposes to lay computing needs on resource-constrained edge devices. Usually, edge/fog applications are highly time-sensitive (e.g., tactical warning applications for natural disaster management for weather conditions). They are required to act immediately on analysis or response to acquired sensing data.

Mist Computing is an evolution of the FC that has been adopted by industry [13] and intends to push computation towards micro-controllers and sensors in IoT applications, enabling sensors to make data filtering computations and ease the load foisted on fog and cloud servers. Mist Computing can enable local decision-making with the help of micro-controllers and sensors. It can help conserve bandwidth and battery power as only essential data is transferred to the gateway, server or router. Micro-controllers and sensors can only be used for lightweight data processing and a narrow range of tasks. Hence, these devices can be used for limited applications.

Thus, the traditional centralized IoT cloud model is undergoing a paradigm shift towards a decentralized model where data processing occurs in part at the network edge or anywhere else along the IoT-to-cloud path (edge/fog) rather than entirely in the cloud.

Therefore, where computation occurs depends on the specific requirements of each application. For instance, real-time constraints need computation to be performed as closer to the data source as possible (e.g., IoT Gateway). Conversely, batch-wise tasks (e.g., big data analytics) are recommended to run on the cloud where computing resources are sufficient. Edge/fog may be a good compromise if a concomitant demand for both computing power and elaboration timeliness.

A key concern with using computing models to support IoT applications is managing different physical and virtual infrastructures (e.g., data centres, edge devices, and IoT devices) according to specific application and service requirements (e.g., latency, data volume, responsivity, and processing delays). However, these models address specific application issues and often coexist or need to cooperate. The coexistence of these computing paradigms in the same application scenario can be hard to manage, and it requires additional services to support interoperability and service management.

C2T infrastructure is characterized by extreme heterogeneity, geographic distribution, and complexity, where the Key Performance Indicators (KPIs) for the traditional model of CC may no longer apply in the same way. Existing resource management mechanisms may not be suitable for such complex environments and require thorough testing, validation, and evaluation before considering live system implementation. A breakthrough approach to address these issues is decoupling user data and applications from networking and security services.

Osmotic Computing (OC) [14], a new computing paradigm that aims to overcome such constraints moves in this direction by addressing issues related to deployment, networking, and security of microservices, called MicroELements (MELs), that are composed and interconnected over cloud/fog/edge and IoT infrastructures with specified levels of Quality of Service (QoS) and security constraints.

OC, borrowing the term from chemistry, goes beyond simple elastic management of deployed resources because deployment strategies are related to requirements of both infrastructure (such as load balancing, reliability, and availability) and applications (such as sensing/actuation capabilities, context awareness, proximity, and QoS). Specifically, OC is based on an innovative application-agnostic approach, exploiting lightweight container-based virtualization technologies to deploy MELs in heterogeneous fog/edge and cloud. Application delivery follows an osmotic behaviour where MELs in containers are deployed opportunistically in cloud and fog/edge systems. Because of the high heterogeneity of physical resources, the MELs' deployment task needs to adapt to virtual environments and involved hardware equipment. Thus, a bidirectional flow of adapted MELs from cloud to fog/edge (and vice versa) must be managed. Moreover, the migration of MELs in the cloud/fog/edge system implies the need for dynamic and efficient management of virtual network issues to avoid application breakdown or QoS degradation.

As the movement of solvent molecules through a semipermeable membrane into a region of higher solute concentration to equalize the solute concentrations on the two sides of the membrane - that is the osmosis process. In OC, cloud and fog/edge resources' dynamic management evolves toward the balanced deployment of MELs satisfying well-defined low-level constraints and high-level needs. Additionally, intelligent, QoS-aware, and contention-aware resource orchestration algorithms should be developed based on the described models, monitoring systems, and configuration selection techniques.

1.2 Research Questions

This thesis investigates how OC can be leveraged to achieve secure and dependable microservices orchestration in the C2T continuum, where deployment and orchestration strategies depend on both infrastructure and applications' requirements, as previously said.

To ground the OC's fundamentals, a thorough investigation of the C2T continuum challenges is necessary. One of the major challenges in the C2T continuum is resource management. Resource management is an umbrella term that encompasses all the characteristics and usage of cloud resources. It includes two main steps: i) resource provisioning (resource detection and resource selection) and ii) resource scheduling (resource mapping, resource allocation and resource monitoring) [15].

Resource management in the C2T continuum faces similar challenges to traditional CC; however, with significant resource constraints, heterogeneity, multi-tenancy, and dynamism. The complexity of such distributed infrastructure mandates automated orchestration of applications and end-to-end management of networking, infrastructure, security and workload placement.

This is required to efficiently and dynamically deploy workloads that satisfy specific requirements (many of which may have geographic, latency, or other user profile idiosyncrasies), while at the same time (i) minimize the cost and energy consumption of finite physical hardware resources, and (ii) meet Service Level Agreement (SLA) commitments.

Designing resource management mechanisms that identify and select resources (resource provisioning) and then map and execute workloads based on those resources and the required service levels (resource scheduling) across the C2T continuum at hyper-scale increases complexity significantly.

Resource management is also responsible for a high standard of security because in the IoT architecture, the data is divided into many data streams gathered from different sensors and different types of services are provided by the networked devices.

In the quest to assist this, we decouple user data and applications management from networking and security management. Hence, we focus on each

of those aspects and formulate the following research questions (RQ) covering the whole IoT application's lifecycle, i.e., development, deployment, execution, management, and orchestration.

(RQ.1) *How to model and deliver IoT applications in the C2T continuum?*

(RQ.2) *Can existing network technologies support the requirements imposed by the end applications for optimal performance?*

(RQ.3) *How can existing cloud virtualization technologies be exploited to optimize the application service deployment and orchestration in the C2T continuum?*

(RQ.4) *Which emerging technologies can be used to ensure the security and data privacy of services?*

IoT applications, therefore, need to have the ability to adapt to a contracting environment. An osmotic application takes advantage of such infrastructures' ephemeral and heterogeneous nature by continuously practising resource, cost, and quality elasticity. Application components scale vertically and horizontally in the infrastructure to where they incur the least cost, perform the best, or where they can take advantage of a unique resource. In OC, it is necessary to develop holistic decision-making frameworks that automate configuration selection across microservices and resources in cloud and edge data centres to meet QoS constraints.

However, using an osmotic infrastructure poses new challenges for IoT workflow application developers and operations managers as they need awareness of resource/device heterogeneity, virtualization software heterogeneity (e.g., hypervisor vs container), data analytic programming model heterogeneity (e.g., stream processing vs batch processing), geographic distribution and network performance uncertainties. The seamless orchestration and operation of such an infrastructure are hard due to the involved devices' heterogeneity and network connectivity. Accordingly, there is no established solution available yet. In the quest to assist this, we derived the following RQs:

(RQ.5) *How does OC mitigate the heterogeneity of physical resources to support the deployment and elastic management of MELs with different levels of requirements of both applications and infrastructure?*

(RQ.6) *How to implement an osmotic orchestration of the resources (application and infrastructure) within the cloud, fog, edge, IoT, and manages the application lifecycle including deployment, chaining, execution, monitoring, and migration?*

(RQ.7) *How to dynamically determine the optimal placement plan for composite applications given a specific set of objectives and constraints, and deploy*

these components over the C2T continuum as suggested by the placement plan?

1.3 Thesis Contributions

Based on the RQs mentioned above, the main overarching contributions of this thesis are presented below. Our contributions to the scientific community can be classified into four parts.

I We investigate the C2T continuumn resource management deriving four research directions by decoupling user data and applications management from networking and security management.

- Namely, the first direction answer to (RQ.1) by proposing an efficient new service delivery model for IoT applications called IoT-as-a-Service (IoTaaS). We consider three use case scenarios (e.g., IoTaaS for Public Safety and Disaster Response (based on publications (IJ.6) and (IC.13)), a smart home scenario that proposes a smart meter IoTaaS consisting of a Fast Fourier Transform (FFT)-based microservice (based on publication (IC.10)), and an IoTaaS Kinect-based gait assessment for Ataxia (based on pubblications (IJ.4) and (IJ.5))) as an application example and show the potential benefits of the continuums.
- The second direction investigates connectivity challenges in the C2T continuum by proposing cutting-edge approaches, methodologies, technologies and IoT applications that are benefiting from the innovative use of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) edge-cloud integration (answer to (RQ.2) and (RQ.3)). We analyze the performance of four SDN network overlays that are Open Virtual Networking, Calico, Weave, and Flannel, in different setups. According to the use case scenario, the obtained results give valuable support for deciding the most suitable overlay solution. This contribution is based on publication (IC.1)
- Finally, the third contribution provides a comprehensive discussion of integrating the IoT system with blockchain technology. The Blockchain-as-a-Service (BaaS) for the IoT is presented to show how various blockchain technology features can be implemented as a service for various IoT applications. Moreover, two innovative use case scenarios highlight the feasibility and benefits of using blockchain technologies (e.g., Hyperledger Fabric, Ethereum) in the continuums. Further, the first use case scenario proposes a Hyperledger Fabric-based BaaS for intersection management. It faces the problem

of preventing vehicular collisions in intersections by proposing a Multi-Agent Autonomous Intersection Management (MA-AIM) system based on V2I/I2V (based on publications (IC.8)). The second use case proposes an Ethereum-based BaaS solution for Health Information Exchange (HIE) systems suitable for an EHRs/EMRs-IoMT scenario. This contribution addresses (RQ.4) (based on publication (IC.11)).

II The second part investigates how IoT applications into the C2T continuum can benefit from OC. The contributions of this part can be summarized as follows.

- The first contribution seeks basic concepts, methodologies, key technologies behind OC ecosystems. Also, an approach enabling OC environments to quickly redeploy containerized microservices eliminating the application outage and promptly reacting to failures is proposed. To achieve such a goal, we discuss a blue-green deployment technique. Moreover, to test such a mechanism, an osmotic application is discussed and tested. This contribution is based on publications (IC.4) and (IC.9).
- In the second contribution, two use case scenarios highlight the benefits of using OC for IoT application management in continuums. Driven by the needs of complex management mitigation, greater agility, flexibility, and scalability, the first use case aims to propose an innovative OC ecosystem leveraging Functions as a Service (FaaS) (based on (IC.14)). Furthermore, to support the FaaS-based OC ecosystem, an osmotic flow model for video surveillance in smart cities is presented. The second use case proposes a closed-loop OC flow model applied to a gamified cognitive rehabilitation use case. Moreover, the use case introduces a customized virtual reality system based on a serious game that allows the patient to carry out physical and cognitive rehabilitation therapies using a natural user interface based on Microsoft© Kinect (based on publication (IC.7)). This contribution addresses (RQ.5)).

III The third part introduces OsmoticToolkit, a cost-effective and flexible toolkit for the from-scratch design of OC ecosystems and the emulation of real-world applications and workloads in virtualized environments. The toolkit enables network topologies design according to the use-case, deploying private/public cloud, edge/fog, and IoT nodes as Docker containers running a QoS- and contention-aware orchestration of Docker-based applications on those nodes. The toolkit provides valuable support for understanding the impact of processing power, workloads, and QoS require-

ments while preserving the users' Service Level Agreements (SLAs). We conduct an extensive experimental evaluation of the Osmotic Monitoring system to study the proposed solution's scalability (based on publication (IC.17)). This contribution answers to (RQ.6).

IV The fourth part implements a solution to support engineers while transitioning from cloud-native to continuum-native by proposing a Rule-based Matchmaker called RBMM that combines several decision factors typically present in software description formats and applies rules to them. To increase the work's applicability, there is implemented matchmaking along with a composite application deployment scenario, where the OsmoticToolkit is taken into consideration as a practical demonstrator. Further, we present novel approaches to service integration and orchestration in the cloud as well as rule-based and dynamic workflow management without a priori design-time knowledge (based on publication (IC.18)). This contribution addresses (RQ.7)

1.4 Thesis Outline

In this section, the structure for the rest of the thesis is presented in detail. Further, each research chapter is self-contained and introduces an overview of related works looking at its future directions.

Part I overviews the C2T continuum in general and faces the resource management deriving four research directions, each corresponding to a chapter. A new type of service model for IoT applications into the C2T continuum is proposed in Chapter 2. Network management leveraging SDN and NFV is addressed in Chapter 3, while security management leveraging blockchain technologies is faced in Chapter 4.

Part II investigates how IoT applications into the C2T continuum can benefit from OC. Basic concepts, methodologies, key technologies behind OC ecosystems are introduced in Chapter 5. There is also presented an efficient orchestration approach to redeploy containerized microservices eliminating the application outage and promptly reacting to failures. Finally, Chapter 7 and Chapter 6 proposed two use case scenarios illustrating the benefits of applying OC in the continuums.

Part III proposes OsmoticToolkit, a cost-effective and flexible toolkit for the from-scratch design of OC ecosystems and the emulation of real-world applications.

Part IV presents a Rule-based Matchmaker called RBMM for supporting composite applications deployment in the C2T continuum.

Chapter 10 summarizes the proposed ideas of this thesis and concludes.

Part I

Cloud-to-Thing Continuum

2 A Service Model for IoT Applications

Leveraging the benefits of service computing technologies for the Internet of Things (IoT) can help in rapid system development, composition and deployment. Further, private, public, and hybrid cloud providers are pushed to integrate their systems with IoT devices to provide along with the traditional Infrastructure, Platform and Software-as-a-Service (IaaS, PaaS, SaaS), even a new type of service level, that is called IoT-as-a-Service (IoTaaS) or IoT cloud. Besides, a mesh of IoT cloud providers can be federated to provide a universal decentralized sensing and actuating environment where everything is driven by constraints and agreements in a ubiquitous infrastructure. In this panorama, efficient IoTaaS has strengthened the need to shift services from the "central" cloud into an intermediate layer, closer to users, defined edge. Edge Computing (EC) has become an essential endeavour to sustain the growth of such applications. Hence, IoTaaS is a service delivery model to provide IoT resources in infrastructure, platform, and software through suitable APIs. This creates an ecosystem of invisible technology that operates behind the scenes, providing rich, real-time insights that dynamically respond to the Cloud-to-Thing (C2T) continuum's ever-changing conditions. This chapter deals with transferring the computing intelligence from the cloud to the edge, enabling accurate service delivery with low response time avoiding delays and network failures that may interrupt or delay the decision process and service delivery. Three use case scenarios (e.g., IoTaaS for Public Safety and Disaster Response, a smart home scenario that proposes a smart meter IoTaaS consisting of a Fast Fourier Transform (FFT)-based microservice, and a Kinect-based IoTaaS for gait assessment) show the potential benefits of the continuums.

2.1 The Emergence of IoT-as-a-Service (IoTaaS)

As a consequence of the digital revolution, the democratisation of technology has made it possible for businesses to access a wide range of technological services and solutions. The rise in the Infrastructure, Platform and Software-as-a-Service (IaaS, PaaS, SaaS) delivery model ensured that tools such as artificial intelligence (AI), big data analytics, and Cloud Computing (CC), despite being deemed expensive to own, have permeated the mainstream industrial environments with ease. The "as-a-service" culture was initially popular in the software segment, but with the introduction of software into the industrial environments, the business

model is gaining traction in traditionally non-technological segments such as manufacturing and retail.

Today industrial environments are increasingly software-driven, as software serves as a key competitive differentiator. Software-assisted systems can customise functionality over time and scale as desired. To achieve this, the environment's products should be connected and network-native to allow for dynamic updates, support, maintenance, and repair.

IoTaaS refers to the ability to bring together the capabilities of the Internet of Things (IoT) and CC technology stacks to bring forth value to the customer over an end-to-end service delivery mechanism.

2.1.1 Why IoTaaS?

- *Decisions Utilising Data* The major advantage of IoT is the ability to extract critical, business-specific insights that enable organisations to make informed decisions using data from the devices operating within their ecosystem.
- *Automation for Efficiency* The increasing automation capabilities provided by service providers through IoT products help save cost and generate revenue and allow organisations to free up critical resources.
- *Scalability and Large Scale Deployment* Industries looking to expand and those entering the IoT market are benefitted.

The IoT ecosystem's complexity is further exacerbated as services and solutions are delivered through multiple layers of technology sourced from different vendors, requiring varied skillsets for maintenance, support, and repair.

- *Smart Maintenance* In the IoTaaS ecosystem, physical assets can report status and usage metrics. It is instrumental during critical maintenance and repair scenarios. IoTaaS vendors aim to take the device owner out of the equation and provide software updates directly to the connected device. This enables the device manufacturer to analyse trends and identify issues that can potentially increase warranty costs.
- *Service Value Addition* There is a significant shift in the way services are offered in the IoTaaS ecosystem. Providers are increasingly looking to offer services that can augment the product rather than merely improving the functionality. IoTaaS providers offer IoT-based services around the physical product combined with other process information, such as visualisation and analytics, that can deliver actionable insights to improve the business.
- *Product as Service* IoTaaS offers a modern way in which physical assets are consumed in work environments. Along with maintenance and repair,

compliance and regulatory norms are critical elements for the industry's non-disruptive functioning. IoTaaS offers services that alleviate organisations' concerns when it comes to cost surge relating to maintenance and compliance. Through defined Service Level Agreements (SLAs), service providers offer a gamut of services that can address organisations' unique requirements.

- *Data as Product* In today's connected environment, data is the new oil. With the increase in connected offerings comes an abundance of data about how assets are being utilised and services are consumed. IoTaaS providers are increasingly looking at ways to monetise the opportunities brought by the data output. Successful organisations, through insightful information, will be able to leverage their IoT investments and optimise operations and differentiate their products and services.

2.2 IoTaaS for Public Safety and Disaster Response

2.2.1 Introduction

Nowadays, emerging Information and Communication Technology (ICT) for disaster recovery systems are of paramount importance for public safety in general and critical scenarios such as the industrial one in particular. Devastating consequences caused by recent disasters (e.g., hurricanes, earthquakes, or other disasters) manifest the vulnerability of existing Public Safety Communications (PSC) infrastructures unable to fulfil the critical requirements of mobility, ubiquitous access, reliability, scalability, configurability, and flexibility at the same time, and indicate the need for dependable and resilient disaster rescue networks [16]. Thus, this results in the lack of a network that transmits and shares information generated within the emergency. As a consequence, the use of traditional emergency-management systems is no longer viable.

Due to disasters' significant impact on human life, emergency planning and disaster response/recovery approaches vary from one incident to another, depending on each disaster's size and nature. Interoperability, coverage and flexibility of first-aid communication systems are among the most critical problems highlighted by such events. In disasters, communication systems for Public Safety and Disaster Response (PSDR) must be highly reliable and robust and should be able to function in hostile environments.

Many techniques exist to transfer data from the widely distributed sensors that make up the IoT (e.g., using 3G/4G networks or cables), and several emerging wireless technologies have been proposed to provide long-range communication for IoT devices.

However, these solutions have prohibitively high costs, making them impractical for real-life applications. In the last years, Wireless Mesh Network (WMN)

has been considered the most suitable network for disaster recovery applications [17]. The reason is that a WMN is self-organized and self-configured and can be easily implemented without any wired connection between the network nodes. Furthermore, since each node in WMN can also act as a router for forwarding packets, we can build wireless networks covering large areas. WMNs offer many advantages like robustness, stable topology, and reliable coverage.

We present a novel, cost-effective infrastructure-less IoTaaS-WMN able to auto-detect an Emergency Relief State (ERS) due to a disaster and auto-configure itself, federating resources different tenants, in order to remain up and running regardless of the disaster nature. The proposed solution does not require a worker or personnel to configure and manage the PSDR infrastructure. For instance, in the case of ERS detection, the system sets up ad hoc connections self-organizing links according to a tandem-based WMN topology and self-configuring access points (APs) for wireless devices (e.g., mobile phones of disaster victims). Devices connected to ad hoc APs become APs in turn, thus extending the network coverage. Available devices of different stockholders (e.g., municipality, first aid organizations, volunteer groups, etc.) are federated in a whole cooperating system, where each device works simultaneously as a node, gateway and router. The performance analysis carried out on top of our preliminary prototype confirm the feasibility and effectiveness of the proposed IoT-WMN system for multi-hop paths within a tandem and tree-based topology.

The rest of this section is organized as follows. Section 2.2.2 reviews the related work. Research problems and potential solutions are discussed in Section 2.2.3. Section 2.2.4 explains the prototype implementation. Performance are discussed in Section 2.2.5 and, finally, Section 2.2.6 concludes the chapter.

2.2.2 Literature Review

Wireless multi-hop and hoc/access networks dedicated to disaster recovery [18, 19, 20, 21, 22] have been an active research field in the last decade. Catastrophic disasters, such as earthquakes and tsunamis, can destroy large industrial sites and, in the process, leaving many victims isolated from the rest of the world.

Recovering the communication infrastructure is typically slow and expensive, which is not suitable for emergency response. Among many kinds of networks, ad-hoc networks have been widely investigated since suitable for disaster scenarios.

An analysis in terms of performance using several well-known routing protocols metrics to evaluate real case disaster scenarios is provided in [23]. An Unmanned Aerial Vehicles (UAVs) known as Flying ad-hoc Networks (FANETs) for disaster monitoring and surveillance applications is studied in [24]. In particular, Urgency Aware Scheduling (UAS) approach for disaster data classification using urgency levels is proposed to efficiently transmit high and low priority packets with minimum delays in the transmission queue. The optimization of a

single UAV's location and movement to improve the network throughput using high mobility of the UAV to adapt to heavy fluctuation of network traffic under disaster situations is investigated in [25]. Using a discrete-event simulation of a model that includes an ad-hoc network, Monte Carlo simulation, Random WayPoint mobility model, and an IoT network, the planning steps, in terms of routing optimization and redundancy in an emergency scenario, were introduced in [26]. Specifically, the processor utilization and mean queueing time of the network under the emergency scenario where nodes and links may be severely affected are analyzed. In [27] a wireless sensor ad-hoc network system that is having the fastest way to sense and respond to the detection of movement of tectonic plate movements is presented, whereas a disaster management scheme based on a cognitive radio ad hoc network (CRAHN) is presented in [28].

LoRaWAN, a promising technology for IoT applications in the Low Power Wide Area Network (LPWAN) space that can also serve as an alternative communication media in post-disaster scenarios when conventional networks are down, is presented in [29]. A flexible network architecture for Emergencies using WMNs based on a Report Message Forwarding Algorithm (REPMSGFW), a location-based on-demand QoS routing algorithm for reliable delivery data for the first responder in the disaster recovery, is discussed in [30].

State of the art on network disaster recovery includes initiatives based on smartphones. A study on how they can be used for providing communications in disaster recovery is presented in [31]. Specifically, they implement a TeamPhone application consisting of two components: a messaging system and a self-rescue system. The messaging system integrates cellular networking, ad-hoc networking, and opportunistic networking, enabling communications among rescue workers. The self-rescue system energy-efficiently groups the smartphones and sends out emergency messages to assist rescue operations.

Among ad-hoc networks, in this context, WMN is considered as one of the most suitable solutions because it can easily configure a network without any wired infrastructure [21, 32]. Closely related to our work, in [33], the authors build a WMN by using a Movable and Deployable Resource Unit (MDRU) as a base station, which has processing servers, storage servers, and Internet connectivity in an emergency scenario. In [34], the authors propose an ad-hoc networking approach for emergency mobile communications in a satellite and wireless mesh scenario, in which ad hoc and IPv6 mobility mechanisms are combined. Similar to our work, in [35], the authors present an approach for the on-the-fly establishment of multi-hop wireless access networks (OEMAN) for disaster response and conduct an experimental evaluation on top of our preliminary prototype over Windows-based laptops to verify the feasibility and the effectiveness for multi-hop paths of up to seven hops. A proof-of-concept prototype for this approach has been built and demonstrated in practice. However, this approach still lacks a high-level fundamental communication abstraction

that can simplify network establishment and configuration, a more rigorous design, and a thorough analysis of its effectiveness in different real-life settings.

These solutions are expensive for deployment and could be slow because specific equipment is not always available on-site. Therefore, it may take a substantial amount of time before emergency rescue/technical teams reach the disaster areas to deploy these equipment/networks. Our approach complements these solutions with a fast mechanism to extend connectivity to the disconnected victims by leveraging their on-site mobile devices.

This contribution overcomes these drawbacks and presents the following main new contributions: i) different from the works mentioned above, our approach aims at simplifying the establishment of multi-hop communication by implementing an infrastructure-less WMN leveraging cost-effective and efficient PRUS, e.g., IoT and on-site devices (see Section 2.2.4); ii) it does not require the intervention of a PDSR worker or personnel to configure and manage the PDSR infrastructure; iii) it allows to extend the connectivity covering a larger disaster-affected area by connecting more WMNs through 2.4/5 GHz WiFi links.

Our solution is realized by carefully integrating existing technologies in a practical implementation. As a result, in an ERS, the WMN can self-organize and auto-heal as injured people can easily connect to the multi-hop access network right after a disaster occurs using their commodity mobile devices. To the best of our knowledge, this is the first infrastructure-less IoTaaS-WMN system for PSDR scenarios.

2.2.3 Public Safety and Disaster Recovery Problem

Information on people's safety (e.g., the number of injured, their positions, health status, etc.) is essential to mitigate disasters. People must share their safety status with rescuers as soon as possible. Unfortunately, when disasters occur, network infrastructures can be damaged while emergency communications, such as confirmation of security and management by the government, become much more critical. However, the recovery of existing PSC infrastructures can be complex and need a long time if we consider that the recovery effort analysis suggests that the first 24 hours represent the *golden time* [36] for ERS.

This work investigates how, in a disaster environment, available battery supplied communication devices can actively work in a federated system to provide a seamless communication service towards rescue and first aid teams. Over such a distributed self-configuring network, innovative applications and services are easily deployed through the node closest to the fixed infrastructure, which acts as a gateway [37]. These networks must be installed quickly and transparently for users to achieve this goal, as victims cannot be expected to perform configuration operations or certain multi-hop-enabled software installed on their devices. In practice, three research questions must be overcome:

1. *How to configure addressing, naming, and routing in these networks simply and automatically?*
2. *How to establish multi-hop access networks without requiring any action from the victims?*
3. *How to implement cost-effective multi-hop access networks?*

Reference Scenario

Figure 2.11 illustrates, at a high level, an example of a wireless multi-hop access infrastructure in a PSDR scenario in a smart city. Several tenants in normal working conditions, such as ambulances, public illumination, and semaphores, use IoT devices to exchange messages for their purposes.As shown in Figure 2.11, devices belonging to a tenant are isolated from the others. For instance, this means that semaphores can communicate only with other semaphores (red-line network) and similarly for public illumination and ambulances. A soon as emergency conditions arise, as shown in Figure 2.2, all the communication links are automatically re-configured in order to provide a federated communication environment for PSDR.

Now, we deal with the above questions considering the smart city reference scenario.

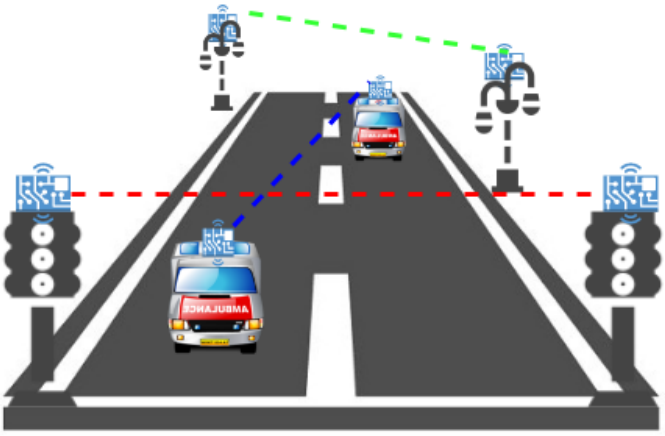


Figure 2.1: Reference scenario: normal conditions.

Research Question #1 Among many types of networks, WMN has been considered the most suitable network for disaster recovery applications. WMN networks can provide wireless coverage of large areas without relying on a wired backbone infrastructure or dedicated access points.

In WMN, a collection of wireless mesh routers provides wireless access to the network, similar to access points in traditional Wireless Access Networks

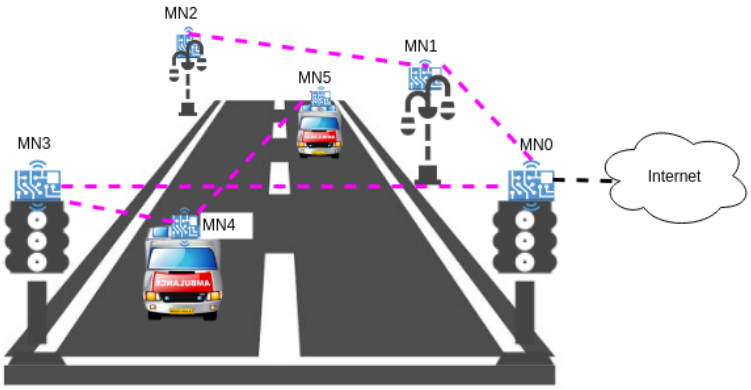


Figure 2.2: Reference scenario: emergency conditions.

(WLANs). However, communication between these mesh routers is achieved through the wireless network, which typically involves multiple wireless hops. One or more mesh routers connected to the Internet can act as a gateway for all other nodes and provide Internet connectivity for the entire mesh network. One of the key features of WMNs is their ability to self-organize and dynamically self-configure. The nodes of a WMN network automatically detect neighbouring nodes and establish and maintain network connectivity in an ad-hoc manner, generally through the use of ad-hoc routing protocols [38]. The self-configuring nature of WMNs allows for easy and rapid implementation of the network. WMNs also can dynamically adopt evolving environments and essentially self-heal in the event of node or link failures.

The proposed scenario involves different types of devices, such as routers, base stations, etc. In detail, a client device is referred to as a "station", while the access router is termed an *Access Point* (AP). In standard infrastructure mode, the AP controls the hand-off procedure and is responsible for:

- Determining that a hand-off is necessary;
- De-associating with the current AP;
- Scanning the 802.11 channels looking for another AP;
- Authenticating to the new AP;
- Re-associating with the new AP.

When a disaster occurs, and at the same time the electrical power grid is also damaged, the network nodes auto-power using the battery. Thus, the $MN0$ node belonging to the semaphore tenant identifies an ERS and starts re-configuring the network. In particular, the $MN0$ node itself becomes an AP and then the adjacent node identifies the newly created ad-hoc AP and connects to it. As

previously mentioned, the device works both as a client, against an AP, and from an AP creating a network.

At that moment, the devices that are close to the configured AP, regardless of which tenants belong, start to federate their resources; that is, $MN1$ (that belongs to the public illumination tenant), try to associate to this AP ($MN0$). The AP initiates the $MN1$ and also transforms $MN1$ into an AP, thus extending the connectivity to farther nodes as shown in Figure 2.2. Therefore, $MN1$ works as a common AP for nearby nodes.

Research Question #2 In this scenario, we can identify two kinds of victims - passive and active; for instance, the system must behave differently for each victim. An active victim has access to his mobile phone and can connect using WiFi to the nearest AP (e.g., helpPoint); once connected, the victim will be automatically forwarded on a web page to share his position safety state. This information is then crucial to saving the victim within the golden time. A passive victim does not have the possibility of using mobile devices to inform rescuers about his position because he/she was injured (we also suppose that the victim disposes of a mobile device). The recovery communication system must also deal with these situations. To do so, the system will scan, through Bluetooth up to 15 m [39], to find the victim's mobile devices and collect this information to help rescuers identify if there could be injured people. In this case, the timeliness of intervention makes it possible to reduce the number of deaths and missing persons. In fact, through this system, there is no longer the need to waste time to make statistics about the number of people present within the disaster area at that moment, understand if people are missing and possibly trying to search them, in this way, there could be necessary sometimes even days or weeks.

Research Question #3 The proposed system leverages cost-effective IoT devices. Section 2.2.4 clarifies the motivations behind IoT devices used.

2.2.4 Prototyping

This section gives an overview of the prototyping of the proposed system.

Hardware Our system's prototype has been developed using IoT devices $ESP32$ micro-controllers and a Raspberry Pi 3 model B+ Single Board Unit (SBU). We chose to involve these devices due to their costs, specifications (e.g.. $ESP32 \leftarrow 3.53\$$, Raspberry Pi Model $3B+ \leftarrow 35\$$) and ease of use. We involved Raspberry Pi Model 3 B+ in using both 2.4 and 5 GHz WiFi interfaces to extend the connectivity between two WMNs quantifying the communication overhead impact on PDSR applications' suitability where timeliness is required. The Raspberry Pis are also auto-powered using Battery Pack for Raspberry Pi 3 B+ 4000 mAh, allowing it to be active up to 3-4 hours.

Figure 2.3: Device prototype.

*ESP*32 module is excellent as it can do both the work of an application microcontroller and WiFi-Bluetooth radio at the same time. Given the low energy consumption of the ESP32 microcontroller, we also chose to auto-power by a rechargeable lithium Odec (3.7 V, 2600 mAh) battery (in battery-powered mode, the device is still active up to 2 hours using WiFi and Bluetooth simultaneously).

The device is illustrated in Figure 2.3. Raspberry Pi also acts as an edge node allowing to minimize the communication delay with the IoT devices. We implemented three different WMNs topologies, as shown in Figure 2.4. For instance, we involved a Raspberry PI 3 Model B+ acting as an AP and 6 ESP32 devices to implement the tandem and tree-based topologies. Next, the last scenario is focused on the communication establishment between two WMNs. Here we used a 2.4/5 GHz router with auto-power 12 V power bank, 2 Raspberry Pis Model 3 B+, where each implements an AP for each WMN and connecting to the router, and 4 ESP32 devices.

Software In general, a WMN is an infrastructure-less system, where communication between distant nodes of the same network takes place through adjacent hops; therefore, every node acts as an AP for other nodes to connect to and as a client to connect to one AP of another node. There is a limit of 10 station nodes per AP for ESP32. Each node that is not already/anymore connected to AP scans for AP's from other nodes periodically. It will connect to the AP with the strongest signal, which is not already present in the list of connections or sub-connections, i.e., unknown yet to the node. By only connecting to unknown APs, the mesh avoids the creation of network loops, such that there is a single route between each pair of nodes in the mesh.

In order to implement the infrastructure-less WMN, we adopted the *painlessMesh* protocol. painlessMesh allows creating a self-organizing and repairing network where all nodes are connected. painlessMesh is designed to be used with Ar-

duino, but it does not use the Arduino WiFi libraries, as we were running into performance issues (primarily latency) with them. Instead, the networking is all done using the native ESP32 and ESP8266 SDK libraries, available through the Arduino IDE.

painlessMesh does not create a TCP/IP network of nodes. Rather each of the nodes is uniquely identified by its 32bit bit *chipId* which is retrieved from the *ESP*32 using the *system_get_chip_id()* call in the SDK. Messages can either be broadcast to all of the nodes on the mesh or sent specifically to an individual node identified by its *nodeId*. Messages sent through painlessMesh are encoded as JSON objects.

Raspberry Pi acts as AP for the gateway node of the WMN and, simultaneously, as a server for gathering stats and managing the WMN. We used the Mosquitto (MQTT) broker installed on the same Raspberry Pi. The broker is responsible for receiving all messages, filtering the messages, decide who is interested in them, and publishing the messages to all subscribed clients. Therefore, an essential element of WMN is the gateway, which will act as an MQTT client subscribing and publishing to specific topics.

As previously mentioned, the device works both as a client against an AP whose SSID and password are indicated and as an AP creating a network whose SSID, password, and port must be specified. Besides, the address and port through which the broker may be reached for the publication and reception of messages must also be indicated (see Listing 2.1).

Listing 2.1: Configuration parameters.

```
#define MESH_PREFIX "xxxxxx"
#define MESH_PASSWORD "xxxxxx"
#define MESH_PORT 5555
#define STATION_SSID "xxxxxx"
#define STATION_PASSWORD "xxxxxx"
IPAddress mqttBroker(x, x, x, x);
painlessMesh  mesh;
WiFiClient  wifiClient ;
PubSubClient mqttClient(mqttBroker, 1883, mqttCallback,  wifiClient );
```

The setup() function (see Listing 2.2) allows initializing the mesh node (calling the init() function) by setting the serial to 115200 baud per second to read from the serial port. Then, it allows the device to create its network, exploiting the previously specified prefix, password, and port. Besides, the STA_AP is set to allow devices to act as clients against APs, APs against the mesh network. Next, the password encryption mode is set to WPA-PSK2 through the WIFI_AUTH_WPA2_PSKd parameter. Finally, the channel on which the gateway works is set; it must be the same for the mesh network devices; otherwise, it will be impossible to find the gateway and then connect to the network. Subsequently, the functions to be called are indicated if: i) a message is received using *onRe-*

ceive(&receivedCallback), ii) establishment of a new connection using *onNewConnection(&newConnectionCallback)*, ii) modify or remove a connection using *onChangedConnections(&changedConnectionCallback)* and finally iv) Synchronization with other nodes through *onNodeTimeAdjusted(&nodeTimeAdjustedCallback)*. Then, the connection to the AP is established for sending data outside the mesh network.

Listing 2.2: Mesh setup() function snippet.

```
void setup () {
    Serial .begin(115200) ;
    mesh.setDebugMsgTypes( ERROR |
    STARTUP | CONNECTION );
    mesh. init ( MESH_PREFIX, MESH_PASSWORD,
    MESH_PORT, STA_AP,
    WIFI_AUTH_WPA2_PSK, 1);
    mesh.onReceive(&receivedCallback) ;
    mesh.onNewConnection(&newConnectionCallback);
    mesh.onChangedConnections(&changedConnectionCallback);
    mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
    pinMode(LED,OUTPUT);
    mesh.stationManual(STATION_SSID, STATION_PASSWORD);
}
```

The *receivedCallback()* function (see Listing 2.3) allows the execution of different actions depending on the messages received. For example, if the message received begins with the word *"data"*, it must be published under the topic "painlessMesh/data"; otherwise, it has to be published on "painlessMesh/from/broadcast". In this way, the messages coming from the mesh network are certainly delivered to the server.

Listing 2.3: Mesh receivedCallback() function snippet.

```
void receivedCallback ( const uint32_t &from, const String &msg ) {
    Serial . printf ("bridge: Received from %u msg=%s\n", from,
        msg.c_str());
    if (msg.startsWith("data")){
        String topic = "painlessMesh/data";
        mqttClient . publish ( topic . c_str () , msg.c_str () ) ;
    }
    else {
        String topic = "painlessMesh/from/broadcast";;
        mqttClient . publish ( topic . c_str () , msg.c_str () ) ;
    }
}
```

For the communications from the server to the mesh network instead, the function *mqttCallback()* (see Listing 2.4) is used. If the topic contains the *"broadcast"*

keyword, it means that the message must be forwarded to all the mesh network nodes using the command *mesh.sendBroadcast(msg)*. Conversely, if the topic contains the *"gateway"* keyword, it means that the message must be forwarded to the gateway node. Finally, if the topic does not contain any of the two words, it means that the message is addressed to a specific node, so the device's connection will be verified through the *mesh.isConnected(target)* command, and in case of an affirmative reply, the message is then sent with the *mesh.sendSingle (target, msg)* command. If the target device is not connected, the message *"client not connected"* will be sent to the broker to warn the user of the impossibility of delivering the message.

Next, according to Figure 2.4, the last arrangement allows extending the connectivity on a larger disaster-affected area by connecting two different WMNs. To do so, we used a router to connect both Raspberry Pis where each one acts as publisher and subscriber to the other network sending messages on the topic "painlessMesh/to/broadcast" and forwarding the message received to own mesh.

Listing 2.4: Mesh mqttCallback() function snippet.

```
void mqttCallback(char* topic, uint8_t* payload, unsigned int length) {
    ...
    String targetStr = String(topic).substring(16);
    if(targetStr == "gateway"){
        if(msg.startsWith("GetEpochTime")){
            String mex=(String)(data+(millis()/1000));
            mqttClient.publish("painlessMesh/from/gateway",mex.c_str());
        }
        ...
    }
    else if(targetStr == "broadcast"){
        ...
        mesh.sendBroadcast(msg);
    }
    else {
        uint32_t target = strtoul(targetStr.c_str(), NULL, 10);
        if(mesh.isConnected(target)) {
            mesh.sendSingle(target, msg);
        }
        else {
            mqttClient.publish("painlessMesh/from/gateway", "Client
                not connected!");
            ...
```

2.2.5 Evaluation

The evaluation's main purpose is to verify whether WMN works well to bring connectivity to isolated people in the case of disasters. Several field experiments have been conducted to quantify the network performance when the number of nodes and the number of hops increases. Specifically, we evaluated the network establishment's effectiveness and its performance in terms of round trip time delay (RTT), packet loss, throughput, and goodput in multi-hop topologies. We evaluated the WMN in 2 different configurations, such as a) tandem-based with 5 hops and b) tree-based mesh network with 5 nodes (MN_i, i=1...5) respectively, with varying packet size of 10, 100 and 1000 bytes. As mentioned before, each WMN topology was established with 15 m hop-distance between MN_i, $i = 0...5$) nodes. Concretely, RTT delay, packet loss, throughput and goodput between MN_i, $i = 1...5$ and MN_0 in both topologies were evaluated.

To extend connectivity on a wide area, we also gathered the communication performance between two different WMNs (WMN_1 and WMN_2) in terms of inter-mesh delay using WiFi links of 2.4 and 5 GHz respectively. Finally, we calculated the entire infrastructure's delay to show how WiFi links impact the entire infrastructure.

Every experiment was conducted, repeating it 30 times (calculating average values and confidence interval of 95%). During the experimentation, it was drawn out that the testing condition (affected by the surrounding environment) in each setting (5-hops tandem and 5-hops tree-based mesh networks) did not change significantly during the experiments.

Experimental Methodology

In order to quantify throughput and goodput in WMNs, we adopted the PPTD (packet pair/packet train dispersion probing) with VPS (variable packet size probing) methodology proposed in [40]. For instance, the PPTD technique allowed to quantify the goodput metrics, which is defined as the total useful application-level bytes transferred as indicated in Equation (8.1),

$$g = \frac{s}{t} \tag{2.1}$$

where s is the packet size (bits), t is the transmission time (seconds), hence and g is the goodput(bits per second). For instance, g could also be used as a measure of throughput. This does not happen in wireless networks; for instance, the application-level throughput is never close to the channel level throughput. In fact, at the Maximum Size Data Unit (MSDU), the goodput for 802.11b networks reaches only 60%. Hence, we must quantify the overhead per packet.

In order to do so, supposing the overhead x to be constant, if x is overhead in bits, the bandwidth (e.g., throughput) at the channel level is given in Equa-

tion (4.2).

$$T = \frac{s + x}{t} \tag{2.2}$$

where s is the packet size, t is the dispersion time of packet pair and T the throughput in mega bits per second.

As mentioned above, in our experiment, we uses packet sizes of 10, 100 and 1000 bytes. We collect many values of g (g_i, $i = 1...n$), we then filter this data with a band pass near the median which allows all values of $[median(g) - range, median(g) + range]$. Then,

$$\frac{s_i + x}{t_i} = T = \frac{s_i - s_i i + 1}{t_i - t_{i+1}} \tag{2.3}$$

By replacing t_i with $\frac{s_i}{g_i}$ and t_{i+1} with $\frac{s_{i+1}}{g_{i+1}}$, we obtain

$$\frac{s_i + x}{\frac{s_i}{g_i}} = T = \frac{s_i - s_{i+1}}{\frac{s_i}{g_i} - \frac{s_{i+1}}{g_{i+1}}} \tag{2.4}$$

which gives the overhead x as,

$$x = \frac{s_i - s_{i+1}}{\frac{s_i}{g_i} - \frac{s_{i+1}}{g_{i+1}}} \times \frac{s_i}{g_i} - s_i \tag{2.5}$$

Using different combinations of packet sizes as (s_i, s_{i+1}), (s_i, s_{i+2}), $(i = 1...n))$ we obtain many values of x (overhead) in bits. The values obtained for x, are again filtered over a range, which was chosen at $[median(x) - \frac{median(x)}{2}, median(x) + \frac{median(x)}{2}]$. The remaining values are averaged to get a final measure of x.

Then, we can calculate throughput T using.

$$T_i = \frac{s_i + x}{s_i} \times g_i \tag{2.6}$$

We obtain many values of T; for instance, the final value is averaged as shown in Equation (2.7).

$$T = \frac{\sum T_i}{n} \tag{2.7}$$

Finally, the goodput is calculated using the formula Equation (2.8).

$$g = \frac{s}{s + x} \times T \tag{2.8}$$

As well as, we also calculated packet loss using Equation (2.9) and RTT delay.

$$DLR = 100 \times \frac{n - M}{n} \tag{2.9}$$

where *M*is the number of bytes received and *n* is the number of bytes sent.

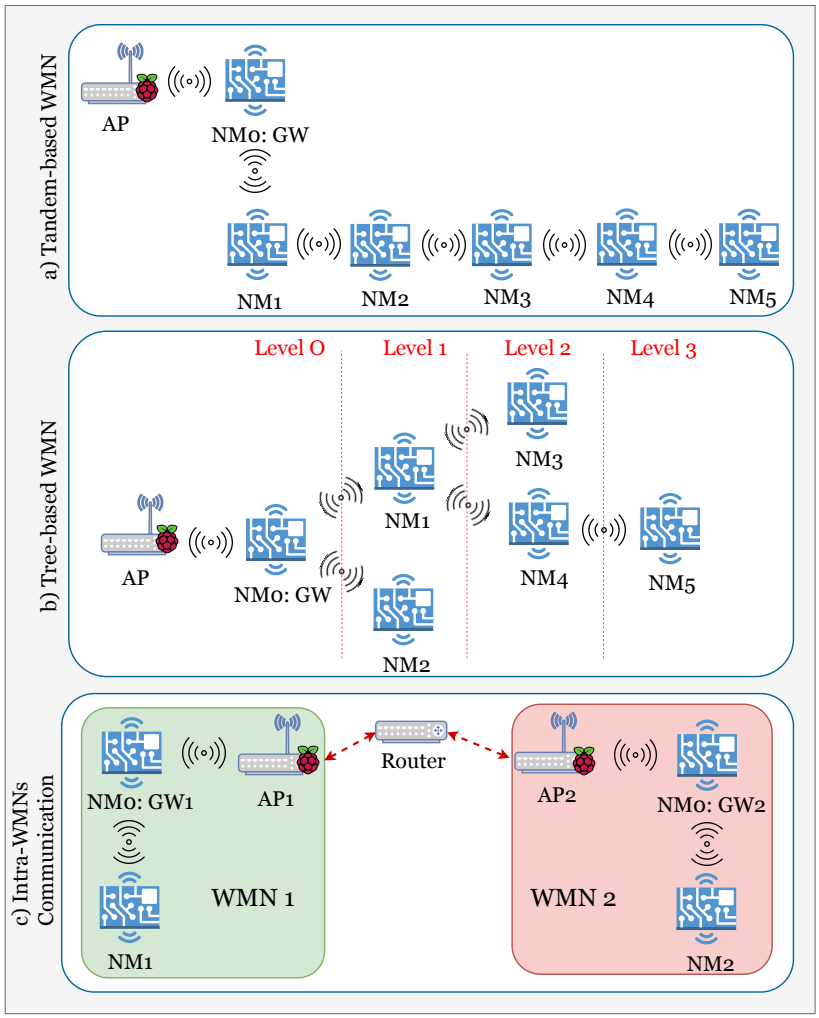


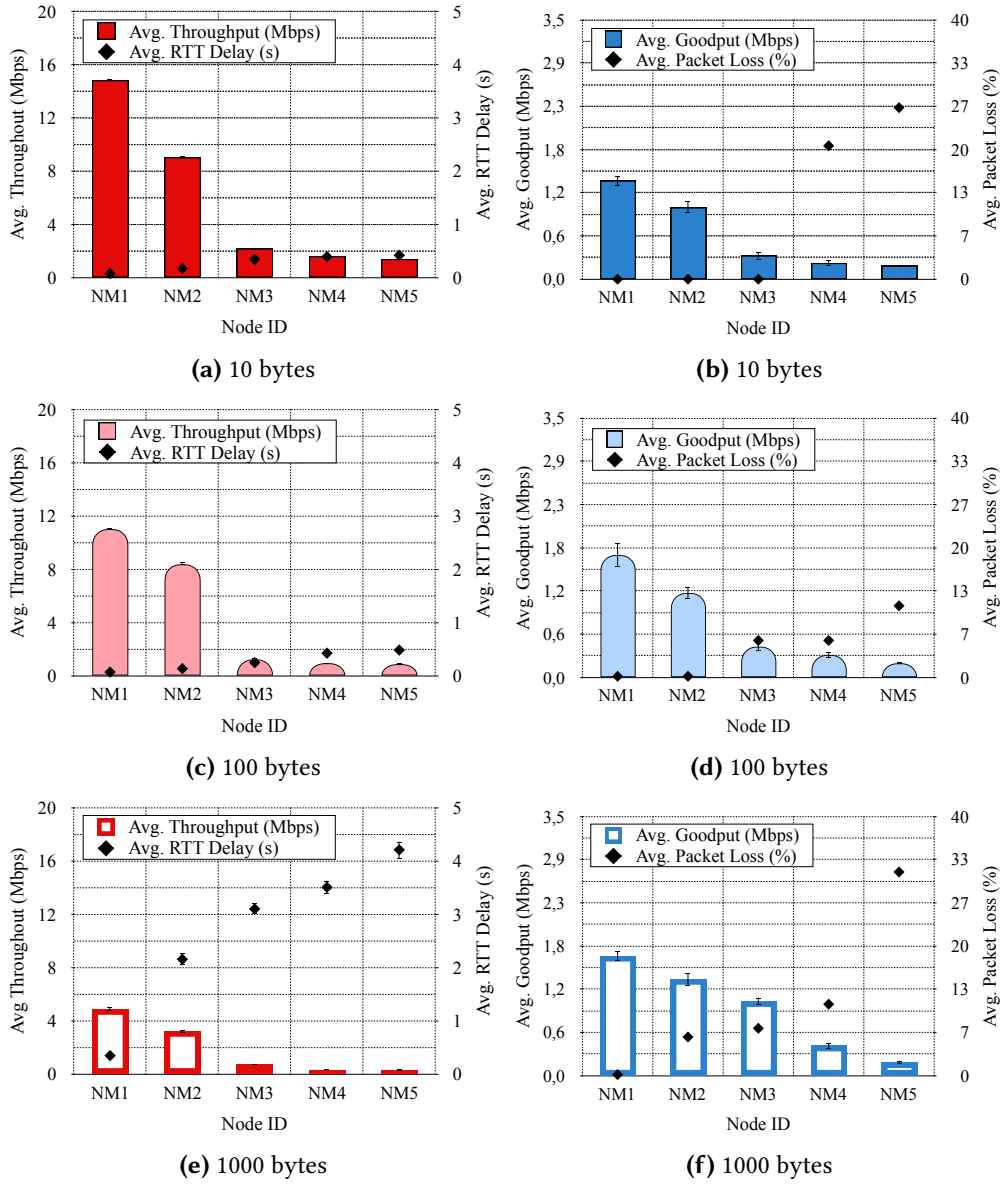
Figure 2.4: Proposed WMN topologies.

Real-World Performance Evaluation

Although a WMN is physically extensible, we carried out further studies to understand how transmissions via multiple wireless hops suffer from rapid drops in performance as the number of hops and payloads increases.

Tandem-based WMN with 5 Hops In the first set of tests, we arrange tandem WMN topology with 5 hops (MN_i, $i = 1...5$) and a MN_0 gateway as shown in Figure 2.5. Figure 2.5 (a), Figure 2.5 (c) and Figure 2.5 (e) show how packet size and the number of hops impact on the variation of the average RTT and throughput.

For example, the avg. RTT delay is small as 0.075 s when the number of hops is equal to 1, and the packet size is 10 bytes; the avg. throughput reaches



(a) 10 bytes **(b)** 10 bytes

(c) 100 bytes **(d)** 100 bytes

(e) 1000 bytes **(f)** 1000 bytes

Figure 2.5: 5-hops tandem WMN: packet size impact and number of hops on RTT, throughput, goodput, packet loss.

14.83 Mbps. With the same packet size of bytes, the last hop (MN_5 node) of the WMN has an avg. throughput equal to 1.41 Mbps and an avg. RTT delay of 0.419 s.

As a result, the avg. RTT delay is still low, namely less than 0.5 s, even for the last nodes of the networks (MN_4 and MN_5) when the packet size is 10 and 100 bytes, respectively. Therefore, we can state that as the number of hops increases, we obtain a degradation in the throughput and RTT delay performances; consequently, a high latency implies a low throughput.

Again, the avg. RTT linearly increases at a rate of around 0.07 s per hop for a packet size of 10 bytes, around 0.15 s per hop for a packet size of 100 bytes, and around 1 s per hop for a packet size of 1000 bytes. Conversely, sending payloads of 1000 bytes saturates the network degrading the performance; the worst case is registered with the last hop (MN_5 node), which has an avg. RTT delay of 4.209 s while the avg. throughput is around 0.3 Mbps, but still reaches acceptable values. For instance, we observe that RTT delay almost linearly increases with the number of hops and the packet size while the throughput decreases.

Next, we want to investigate the impact of packet size on goodput and packet loss (see Figure 2.5 (b), Figure 2.5 (d) and Figure 2.5 (f)). For a packet size of 10 bytes the first three nodes (NM_1, NM_2 and NM_3) have avg. packet loss of 0%, for a packet size of 100 bytes only the first two nodes have avg. packet loss of 0%, while for a packet size of 10 bytes only the first one has avg. packet loss of 0%.

However, we can state that the achievable throughput after two hops decreases to less than half of the achievable throughput after one wireless hop. When data travels 4 or 5 hops, the throughput drops to less than 10% of the achievable throughput after one wireless hop. This is mainly caused by interference between wireless hops; the connections at nodes MN_4 and MN_5 were still intermittent because of high link failure probability. That is why packet loss at those nodes significantly increases. It can also be noted that those nodes have similar average values of goodput, throughput, and RTT delay. As the packet size is increased, the goodput is also increased. However, with the increase in packet size, the cost of error, such as packet loss, is increased as well.

Tree-based WMN with 5 nodes In addition to the tandem topology, the tree-based WMN shown in Figure 2.4, is useful for verifying the network's usability when an intermediate node serves several clients. This experiment evaluated the same performance parameters in terms of RTT, packet loss, throughput, and goodput.

Figure 2.6 (a), Figure 2.6 (c) and Figure 2.6 (e) show how packet size, number of nodes and network type impact on throughput and RTT delay. For instance, by analyzing the network topology illustrated in Figure 2.4, since no fairness mechanism has been employed, performances are slightly different at the nodes belonging to the same level.

For instance, with a packet size of 10 bytes, MN_1 and MN_2 nodes, which are directly connected to the gateway and both at level 1, have almost similar performances in terms of RTT delay and throughput as expected; MN_1 has avg. RTT delay almost 0.024 s and avg. throughput of 3.98 Mbps, MN_2 has an avg. RTT delay equals to 0.025 s and an avg. throughput equals to 3.53 Mbps. Also, MN_3 and MN_4 follow the same trend as they are connected at level 2. Compared with the tandem-based topology, the results reveal that the avg. throughput is around 60% lower - however, the lowest throughput around 0.36 Mbps, which is still adequate for web browsing. The RTT delay follows the same trend for the

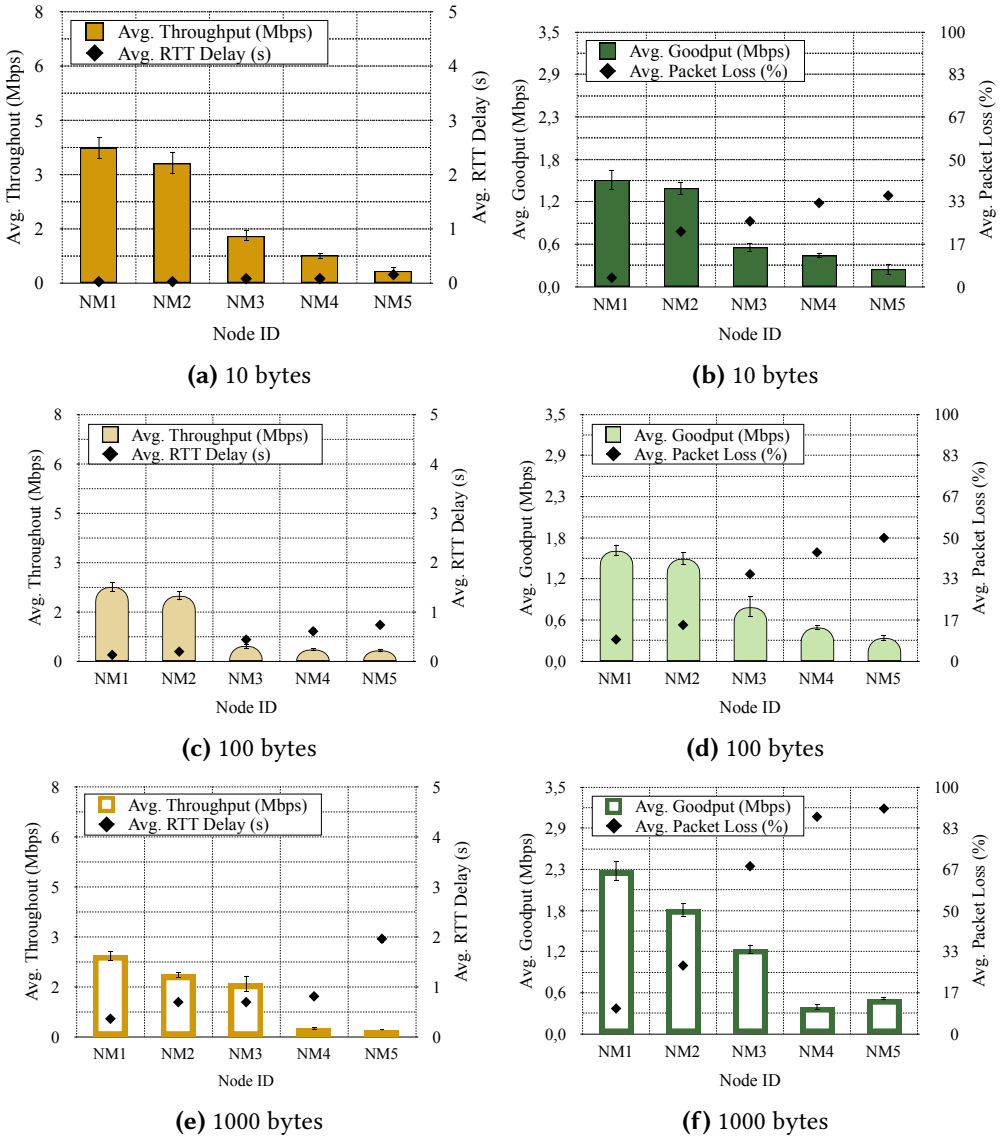


Figure 2.6: Tree-based tandem WMN: packet size impact and number of hops on RTT, throughput, goodput, packet loss.

nodes belonging to the same level. Moreover, compared with the tandem-based topology, the highest avg. RTT delay is almost 2 s, which is around 50% less.

In fact, within this kind of topology, we notice the nodes belonging to the first level have RTT delays comparable with those registered with one hop in the tandem-based topology. The same can be seen for the nodes at the second level, in which RTT delays are comparable with those of the second hop in the tandem-based topology. For instance, this behaviour is mainly caused by interference between nodes within the same level.

Figure 2.6 (b), Figure 2.6 (d) and Figure 2.6 (f)) show the goodput and packet

loss trends as well. As previously mentioned, the nodes belonging to the same level also have similar goodput and packet loss performances. Comparing the goodput and throughput values with those of the tandem-based topology, we notice a significant degradation, mostly in the packet loss. As said above, this is mainly caused by interference between nodes within the same level. In the worst case, the avg. packet loss reaches around 91%. Lower throughput performances imply greater packet loss values.

For both implemented WMN topologies, the overall trend is clear: as the distance and packet size increases, the throughput and goodput values are degraded and become less predictable, RTT and packet loss are increased. Tree-based topology seems to perform better than the tandem configuration in terms of RTT delay and goodput. Conversely, the tandem-based WMN performs better than the tree-based in terms of throughput and packet loss.

To conclude, for both topologies, the RTT values are quick enough for emergency response, throughput, and goodput values. For most nodes in the tandem-based topology, the packet loss is in the range of 0–4%, acceptable in an intermittent wireless environment. The packet loss measured with the tree-based network is not acceptable at all.

Intra-WMNs Communication In this experimentation, we tested the extensibility of the WMN to cover more extensive areas by connecting two WMNs (see Figure 2.4) using WiFi links of 2.4 and 5 GHz, respectively. We calculated the intra-WMNs communication delay and the delay of the entire infrastructure, as illustrated in Figure 2.7. In this experiment, we did not evaluate the perfor-

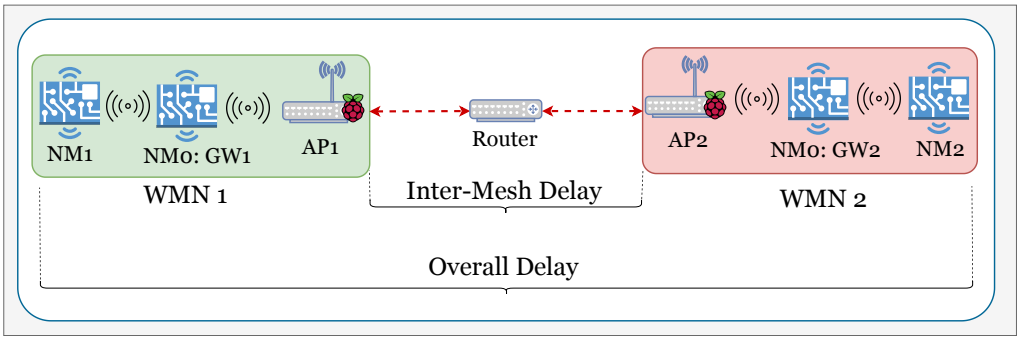


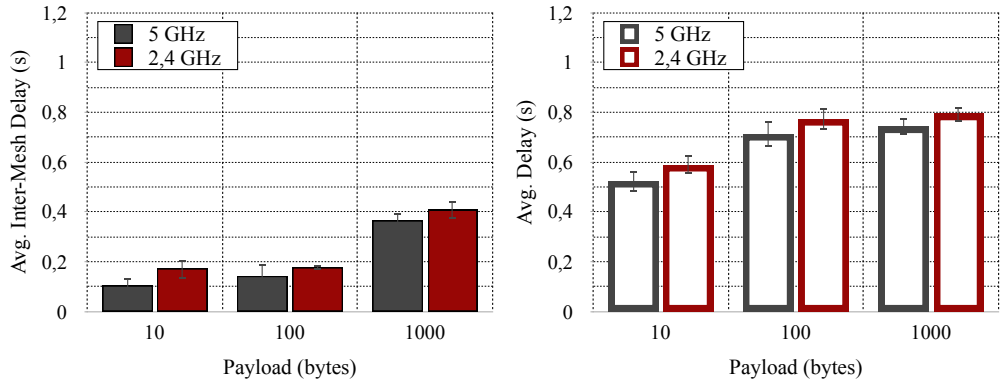
Figure 2.7: Intra-WMNs communication scheme.

mance considering more complex WMNs as above presented because previous experiments already demonstrate each topology's behaviour. This experiment evaluates the impact of WiFi links of 2.4 and 5 GHz on the communication between two WMNs.

Figure 2.8 (a) revels the intra-WMNs delay using WiFi links of 2.4 and 5 GHz, respectively. The intra-mesh delays quantify the time necessary a message arrives from AP_1 to AP_2. We notice the values obtained with the 5 GHz WiFi

outperforms those obtained with 2.4 GHz WiFi. Hence, the 5 GHz WiFi introduces less overhead, translating in higher bandwidths and lowering latencies. For instance, the infra-mesh delays are reduced by around 30% by using 5 GHz WiFi.

Figure 2.8 (b) illustrates how the intra-WMNs delay impacts the entire infrastructure delay. The entire infrastructure delay quantifies the time necessary for the message sent by the MN_1 node of the WMN_1 takes to arrive MN_2 of the WMN_2. As expected, the intra-WMNs delay obtained with the 2.4 GHz WiFi impacts the overall delay; however, the performances are still acceptable.



(a) Impact of 2.4/5 GHz WiFi on the intra-mesh delay.

(b) Impact of 2.4/5 GHz WiFi on the overall delay.

Figure 2.8: Intra-WMNs communication performance.

2.2.6 Summary

This section's focus was on providing a novel, cost-effective, robust, reliable and flexible infrastructure-less IoTaaS-WMN communication system to extend connectivity to disaster victims in a PSDR scenario in the fastest and simplest way using on-site mobile devices. The novelty of the IoT-WMN system is the ability to detect an ERS, to auto-power whether the power grid is damaged and to auto-organize into a tandem-based WMN topology federating devices belonging to different tenants and self-configure ad-hoc APs to disaster victims using their own mobile devices; furthermore, it does not require the intervention of a PSDR worker or personnel to configure and manage the PSDR infrastructure. As well as if they were connected to conventional APs, and automatically contribute to the network extension. Therefore, after the victims connect to the APs, generally, the resulting topology is a tree-based WMN.

Our field experiments carried out on top of our preliminary prototype confirm the feasibility and effectiveness of the proposed IoTaaS-WMN system for multi-hop paths within a tandem and tree-based topology.

2.3 IoTaaS-based Smart Metering

In recent years, great attention has been given to smart grid applications in the C2T continuum. Power Quality (PQ) is one of the major topics associated with the Internet of Things (IoT) from the beginning in both industrial and domestic fields. It determines the fitness of electric power to consumer devices. PQ management is possible through a smart grid, i.e., an electrical grid that includes various operation and energy measures, including smart meters, smart appliances, and energy efficiency resources.

In particular, as far as smart metering in IoT, several initiatives have been proposed up to now. A smart IoT-based energy metering system for microgrids with a load management algorithm is proposed in [41]. A smart metering platform using big data and IoT technologies has been proposed in [42]. A platform prototype was developed and tested for detailed profiling of energy consumption in buildings that allows measuring and monitoring the electric power consumption of individual appliances. An emerging paradigm based on Low Power Wide Area Networks (LPWAN) - LoRa, i.e., a straightforward solution for IoT networks, is discussed in [43]. The deployment analysis of a Narrowband IoT (NB-IoT) system for smart metering is discussed in [44]. A combination of an advanced IoT metering infrastructure with Cloud-based analytics to optimize smart electricity distribution in residential areas is discussed in [45], whereas an IoT based smart home with real-time e-metering using an e-controller is discussed in [46].

A few initiatives have been proposed so far regarding Fast Fourier Transform (FFT) applied for smart metering in IoT. A grid frequency estimation algorithm based on the fractional FFT for IoT nodes time stamps is proposed in [47]. A maximum merging model in the fractional spectrum domain for dealing with complicated power grid channel interference in China grids with data measured by the wireless sensor networks is discussed. An ultra-low-power variable-accuracy bit-serial FFT butterfly processing element for IoT devices is proposed in [48]. Specifically, an ultra-low-power Decimation-in-Time (DIT) radix-2 butterfly arithmetic block for an FFT processor, designed using a bit-serial architecture and operating at a near-threshold voltage (0.6 V), is discussed.

Apart from electric energy monitoring, security is one of the major issues in IoT smart metering. In this regard, a dynamic group authentication and key exchange scheme based on threshold secret sharing for IoT smart metering environments is proposed in [49]. Even regarding security, efficient certificate revocation management schemes for IoT-based advanced metering infrastructures in smart cities is discussed in [50]. In contrast, privacy-preserving protocols for secure and reliable data aggregation in IoT-enabled smart metering systems are discussed in [51].

In this section, differently from the aforementioned scientific initiatives available in the literature, we focus on a smart home scenario proposing a smart

meter IoTaaS consisting of an FFT-based microservice running on a Raspberry Pi 3 device able to coordinate the acquisition and perform harmonic analysis of the frequency signal of the domestic electric grid frequency signal to characterize the non-linear loads originated by the current absorption of electronic devices (e.g., smart TV, computers, etc.) connected on it in order to monitor their status and prevent possible malfunctions and breakdowns.

The rest of this section is organized as follows. Motivations along with an overview of our system prototype are described in Section 2.3.1. Implementation details along with a reference scenario are provided in Section 2.3.2 and Section 2.3.3 respectively. A performance analysis is discussed in Section 2.3.4. Conclusions are summarized in Section 2.3.5.

2.3.1 Problem Analysis and Objectives Definition

Nowadays, electric energy is an essential factor for the development of different application domains. In modern electrical energy systems, voltages and especially currents become less sinusoidal and periodical and even steady-state behaviour may be completely lost due to the increase of non-linear loads such as smart TV, computers, LHE lamps, variable speed drives, fax, laser printers, and other electronic devices representing a critical issue in the power system.

Generally, a linear electrical load draws a sinusoidal current proportional to the sinusoidal voltage. The reason for such a behaviour is that the linear loads do not depend on the voltage to determine their impedance at a given frequency. These loads do not cause any issue to the network they are connected to or other utility's consumers. They always follow Ohm's law. Instead, power electronics loads do not always follow Ohm's law. Unlike linear loads, they do not consume power continuously. When a sinusoidal voltage is applied to a non-linear electrical load, it does not draw a sinusoidal current. The non-sinusoidal current is due to the device impedance changing over a complete voltage cycle. These loads can distort the supply voltage waveform and might cause problems to other loads as well. In fact, harmonic distortions could produce Power PQ problems such as voltage distortion, equipment malfunction, low quality of power, and component failure. Moreover, harmonic distortion also causes financial loss of the customers and electric power companies. Harmonic disturbances encountered are, for instance, harmonics, voltage sags, rapid amplitude variations (flicker), and transients. At the same time, manufacturing processes and various equipment become increasingly sensitive to a distorted voltage waveform.

In this situation, accurate measurement techniques to detect, classify and assess different PQ problems in distorted environments need to be developed and applied. Therefore, developing a proper real-time power quality measurement system based on Digital Signal Processors (DSPs) is essential. DSPs are well suited for mathematical techniques and real-time waveform processing and form a powerful power quality assessment tool.

Advanced signal processing techniques and quantities have to correctly describe changes in all their aspects, whereas it must be possible to assess them accurately and quickly with a limited amount of data. With this identification, power system operators can decide on a strategy to reduce harmonic distortion level with filter placement. FFT allows analyzing the current waveform input in the presence of multiple devices. Moreover, it allows identifying the devices from the current harmonics.

We aim to achieve the objective in this contribution to distinguish different loads and electronic devices based on the spectral variations of the current generated by their connection to the electric grid. To give a clarifying example, we intend to distinguish a smart TV from a computer according to the analysis of their harmonic characteristics. In order to achieve such a goal, it is necessary to develop a smart metering IoTaaS including a microservice able to:

- Acquire the analog electrical signal coming from the domestic grid and to perform A/D conversion;
- Process the sampled data with FFT algorithms and interpret the obtained results to characterize the non-linear loads.

2.3.2 Smart Metering IoTaaS Prototype

Hardware Design

We opted to use Raspberry Pi 3, a small, powerful and lightweight ARM-based computer, to implement the metering system due to its ease of use and cost (only 35$). Raspberry Pi cannot read analog inputs. Analog inputs are useful because many sensors produce analog outputs, so we need to make the Pi analog-friendly. Nevertheless, this can be easily carried out by wiring up an ADC chip on it. In order to find the best trade-off in terms of cost and sapling frequency range for the ADC, we tested two different chips:

1. $ADS1115 : 16bit, 860SPS, I^2C, PGA, cost \simeq 5$ €;
2. $MCP3008 : 10bit, 75KSPSat2.7V, SPI, cost \simeq 3$ €.

Even though the ADS1115 chip guarantees excellent resolution and more advanced programming possibilities, unfortunately, it has a low sampling rate. This, for practical purposes, would limit the sampling band to about 4001 Hz to comply with the Shannon-Nyquist sampling theorem. Such a narrow band may not be enough; there could be harmonics at higher frequencies that our system would not be able to capture. Instead, the MCP3008 chip, even though it has a resolution limited to only 10 bits, allows us to evaluate a wider frequency band. According to the sampling theorem, we could go to sample signals with a

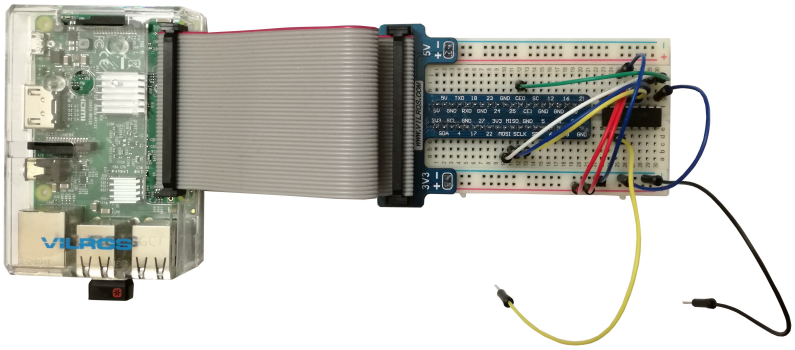


Figure 2.9: Metering system's real architecture.

maximum frequency of about 37 kHz, covering the hypothetical band of interest widely. For this reason, we decide to use the MCP3008 chip.

The MCP3008 chip uses the SPI bus protocol, configured by the Raspberry Pi's GPIO header as shown in Figure 2.9. The MCP3008 chip is electrically powered with 3.3 V since the power supply with 5 V would imply output pulses of the same amplitude that would damage Raspberry Pi 3. By choosing VDD = 3.3 V, we can guarantee the correct output voltage. In order to understand how to interface the MCP3008 chip with the Italian electrical grid providing an alternating voltage of 220 V, we used the *LEM HAIS 50-P* current transducer with low pass filter with cut-off frequency f_T = 3 KHz as depicted in Figure 2.10.

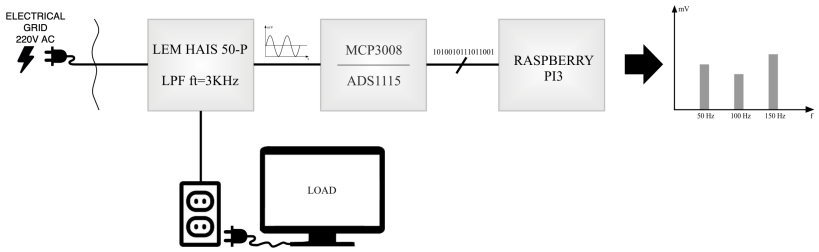


Figure 2.10: Smart metering device.

In order to read analog data, the following pins were used: VDD (power) and DGND (Digital Ground) to power the MCP3008 chip, DOUT (Data Out from MCP3008), CLK (Clock pin), DIN (Data In from Raspberry Pi), and /CS (Chip Select), AGND (Analog Ground, sometimes used in precision circuitry), GND and VREF (Analog Voltage Reference, used for changing the *scale*).

Software Design

This system's software side was developed using a container-based microservice that runs on Raspberry Pi 3 (Raspbian OS) and manages the acquisition, pro-

cessing, and output phases of the smart metering IoTaaS. The container-based approach was adopted in order to simplify service deployment according to an IoT-Cloud scenario. The microservice was developed according to the producer-consumer paradigm. Two Java libraries have been identified and used to process sampled electrical data: *Pi4J* and *JTransforms*. In particular, these Java libraries allow us to manage the Raspberry Pi device and execute the FFT algorithm respectively on the acquired sampled electric grid signal.

Producer In our structure, the producer is represented by the ADC chip, which produces data at each performed conversion. In this paper, we used the *com.pi4j.io.spi* package because the MCP3008 chip uses SPI to communicate with the Raspberry Pi device. The package also includes: *SpiDevice* interface, *SpiFactory*, *SpiChannel* and *SpiMode* classes. Therefore, the Producer class creates a reference of type *SpiDevice* (interface) that will be used to manipulate the object of type *SpiDeviceImpl* necessary for the SPI communication. The *DEFAULT_SPI_SPEED* parameter is used to set the clock frequency to 1 MHz. An infinite loop repeats the *read()* method, passing as a parameter the channel number (pin) on which the input the need to be digitized is connected.

Listing 2.5: Producer classs nippet.

```
public class Producer implements Runnable {
    private final BlockingQueue queue;
    public static SpiDevice spi = null;
    Producer (BlockingQueue q) {
        queue = q;
    }
    public void run() {
        try {
            spi = SpiFactory.getInstance(SpiChannel.CS0,
            SpiDevice.DEFAULT_SPI_SPEED,
            SpiDevice.DEFAULT_SPI_MODE);
            boolean continueLoop = true;
            while (continueLoop) {
                read(1);
            }
        } catch (IOException ex) {
            System.out.println("Error");
        }
    }
}
```

Consumer The consumer is represented by the FFT algorithm, which consumes the data produced by the MCP3008 chip. The *JTransforms* library encompasses several packages, each containing classes that deal with different

transformations, e.g., DCT, DHT, DST, and FFT. Thus, we used *org.jtransforms.fft* package. The packages include two sub-classes that could perform the FFT calculation with either single or double precision and the calculation of mono/-multidimensional transforms. Given the limited resources of Raspberry Pi 3, we chose to use the single-precision *FloatFFT_1D* class that consumes less memory and compute resources. Although the *JTransforms* library allows the processing arrays of any size, it is important to emphasize that the performance is better when the array has a power of 2.

Listing 2.6 reports a snippet of the Consumer class. The n variable represents the size of the array that will be processed by the FFT. The following instructions create two arrays that will contain the amplitude and time values extracted from each element of the FIFO queue. The next instruction creates the array that will contain the data to be transformed. It is $2n$ size because the FFT algorithm is designed for complex numbers, for instance, each sampled value must be followed by an imaginary part equal to zero. Then, the vector which will contain the modules returned by the FFT is created. It has $n/2$ size as it contains the values of the positive part of the spectrum. The negative part will be mirrored with respect to the vertical axis. The last instruction of this snippet creates the object of type Float FFT_1D which will allow the transformation to be performed.

Listing 2.6: Consumer class snippet.

```
public class Consumer implements Runnable {
    private final BlockingQueue queue;
    Consumer (BlockingQueue q) {
        queue = q;
    }
    public void run() {
        try {
            int n = (2048);
            float [] data = new float[n];
            double[] time = new double[n];
            float [] fft = new float[n*2];
            float [] module = new float[n/2];
            FloatFFT_1D fft = new FloatFFT_1D((long) n);
            ...
            ...
```

Buffer for writing/reading data Producer and consumer must be executed concurrently by two threads. Thus, a shared data structure is required and at the same time, concurrent access to the resource must be managed. The ideal data structure is a FIFO queue. The *java.util.concurrent* package provides a series of interfaces and classes for multithreading and concurrent access.

The *BlockingQueue* $< E >$ interface defines blocking methods for inserting

and extracting data from the queue. The *put()* method inserts an element in the queue, while the *take()* one extracts an item from the queue. Among the classes that implement the *BlockingQueue* $< E >$ interface, the *LinkedBlockingQueue* $< E >$ class creates a dynamic queue that is optionally limited in size, in which the elements are arranged according to the FIFO logic. The *put()* and *take()* methods allow us to insert and retrieve data from the structure respectively. We chose to use this class for our microservice.

A solution for calculating f_s through the Sample class The acquisition at a specific frequency from the MCP3008 chip requires real-time I/O management. Our Raspberry Pi device, equipped with a Raspbian Operating System (OS), does not handle real-time applications. In fact, all the running processes and threads need to be scheduled in advance, whereas the microservice has to manage the variable reading frequency from the MCP3008 chip. In order to handle this inconvenience, the Sample class was created (see Listing 2.7). This class conceptually represents a single sample returned by the ADC conversion performed by the MCP3008 chip. It contains two instance variables: a float variable named voltage and a double variable named time. For each conversion, the value of the sampled signal coming from the domestic electric grid will be saved within voltage, while time will save the chronological instant (nanoseconds) in which the data have been stored. Through this artifice, the consumer will know the time interval elapsed for the sampling of the elements taken from the FIFO queue. By saving the time instants relative to the following samples, it will be possible to calculate the frequency f_s with a reasonably reliable approximation.

Listing 2.7: Sample class snippet.

```
public class Sample {
    float voltage;
    double time;
    Sample (float voltage, long time){
        voltage=voltage;
        time=time;
        ...
    }
}
```

2.3.3 Case Study: Smart University Campus

In a university campus, several actors, e.g., professors, students, employees, can connect their own devices to the grid. These devices could be dangerous for others or forbidden by internal faculty regulations. Checking every device connected to the grid is very difficult and expensive. Therefore, the idea is to create a smart metering IoTaaS system that can monitor a specific part of the campus electricity grid and disconnect whether a problem occurs.

Figure 2.11 shows the proposed scenario. As the reader can observe, the scenario labelled with (1) represents the case in which a device (e.g., microwave oven) is damaged and generates on the grid undesired harmonics. The latter could damage other devices potentially connected to the same part of the domestic grid. Therefore, the smart metering IoTaaS can identify that device and eventually disconnect it from the grid.

The scenarios marked as (2) and (4) consider the presence of forbidden devices. In particular, the scenario (2) considers the presence of cryptocurrency mining [52], instead scenario (4) represents the charging of an electric car. In both cases, the smart metering system can identify the forbidden devices and disconnect them from the electrical grid.

Scenario (3), instead, shows the normal usage of a personal computer. In this case, no action has to be done because there is a normal usage of the electrical grid. Apart from a smart university campus, the smart metering IoTaaS can be

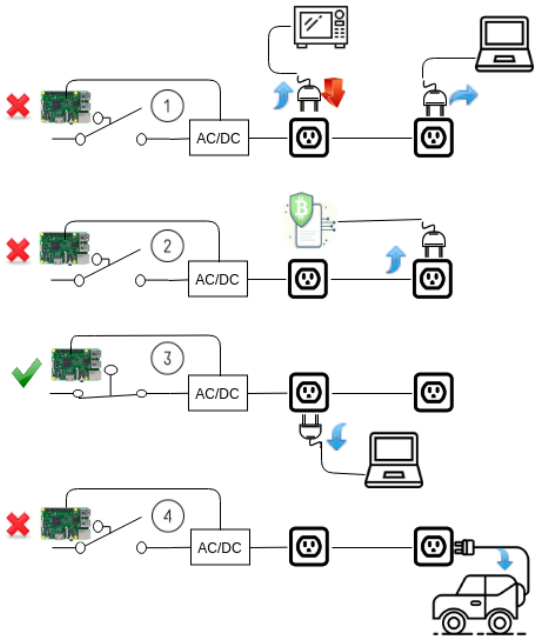


Figure 2.11: Reference scenario.

used in any smart building.

2.3.4 Measurements and Observations of Different Non-Linear Loads

The experiments have been conducted according to two phases, namely calibration and validation. Both phases have been performed in the Laboratory of Electronics, Department of Engineering, University of Messina.

Calibration Phase

Before using the smart meter IoTaaS with real loads, it was necessary to perform calibration tests to ascertain the system's correct functioning, i.e., if it can correctly sample an analog electrical signal and detect its frequency components. To do this, we used a signal generator that allowed us to test the meter's behaviour by varying the input signal at different frequencies and using signals of different shapes (e.g., triangular, sinusoidal, square wave). Furthermore, we have also used an oscilloscope capable of calculating FFT, connected in parallel to the realized system, to verify if the meter's output corresponds to the input signal spectrum.

The FFT algorithm executed on the Raspberry Pi 3, in this configuration, was able to detect with high approximation the frequency components up to about 8 KHz; data processed by the meter deviate slightly from those of the equipment used for the test. Thus, the transform of a pure sine wave with f = 100 Hz, has a spectral component exactly at 100 Hz. The implemented system places the component at a frequency equal to 100 Hz±3%. As previously explained, this error is due to the sample windows and spectral leakage that can occur. This result meets our expectations, as the system allows us to distinguish the fundamental harmonic of the electrical network (50 Hz) from the higher-order replicas (e.g., 100 Hz, 150 Hz, 200 Hz, an so on) that must be detected to characterize the load. This result meets our expectations, as the system allows us to distinguish the fundamental harmonic of the electrical network (50 Hz) from higher order replicas (e.g., 100 Hz, 150 Hz, 200 Hz, and so on) which must be detected to characterize the load.

Validation Phase

The experimentation aim is to characterize non-linear loads generated by specific electrical devices by looking for an identifying *pattern* in the spectrum of each of them. The IoTaaS has been tested with: (i) a computer monitor, (ii) a drill, and (iii) monitor and drill together.

For each characterized load, a bar graph illustrating the representation of the frequency components will be shown. During this phase, windows of different sizes were used. The window of 2048 samples with f_s = 20 KHz allowed us to obtain fairly faithful spectra and acceptable processing times. The consumer's threshold was set to 1, for instance, data saved and displayed in bar graphs do not contain frequency components with an amplitude less than 1 mV. We tried to *isolate* the most significant components for the characterization of loads.

Computer Monitor Figure 2.12 shows the FFT transform of the computer monitor active obtained from the smart metering system. It can be noticed that when the monitor is active there is the presence of many significant harmonics, for instance, at 48 Hz, 148 Hz, 249 Hz and 340 Hz respectively. Such harmonics characterize the computer monitor load.

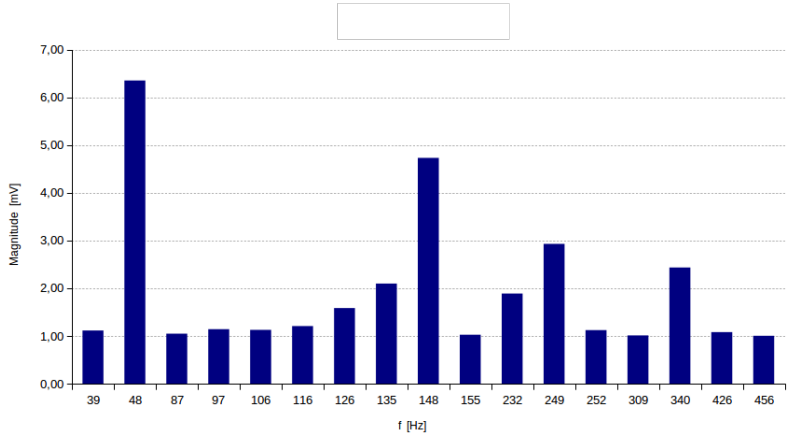


Figure 2.12: Monitor active: FFT spectrum obtained with the metering system.

Drill With respect to the computer monitor, which has a very low current absorption in standby mode, the drill presents a slight current absorption. In particular, we have two peaks corresponding to 48 Hz and 144 Hz as shown in Figure 2.13. As well as in the active mode with 100% of velocity, the FFT transform calculated with the smart metering system is illustrated in Figure 2.14. By turning on the drill, we notice the presence of two significant harmonics

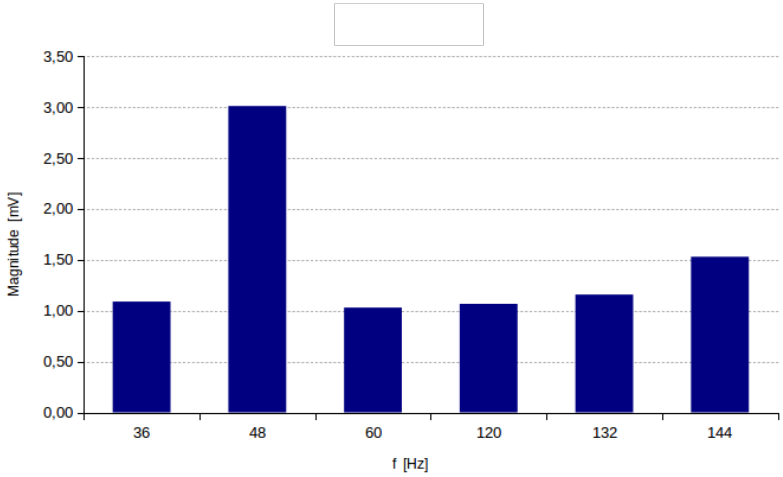


Figure 2.13: Drill stand-by: FFT spectrum obtained with the metering system.

(Figure 2.14) corresponding to 51 Hz and respectively 152 Hz.

Monitor and Drill Together In the last experiment of the validation phase, we considered both computer monitor and drill active. Therefore, we illustrate the FFT transform in Figure 2.15. By analyzing Figure 2.15, we can notice that the obtained spectrum presents two significant harmonics in the neighborhood of

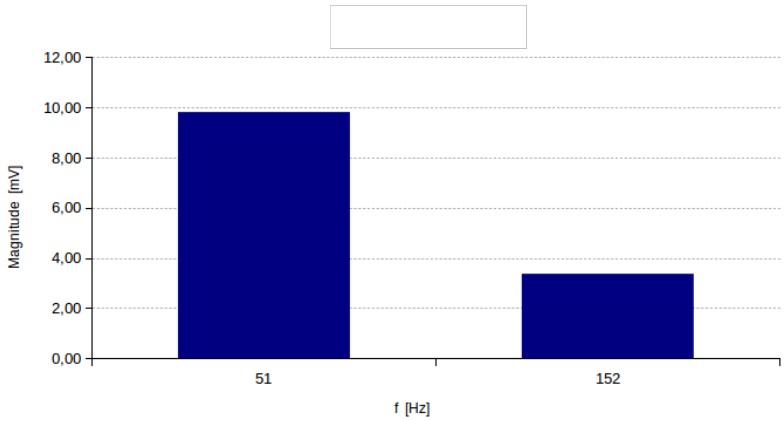


Figure 2.14: Drill active at 100% of velocity: FFT spectrum obtained with the metering system.

50 Hz and respectively 157 Hz. Therefore, these two harmonics are characteristics for both monitor and drill; the amplitude of these two peaks is significantly increased due to both loads active.

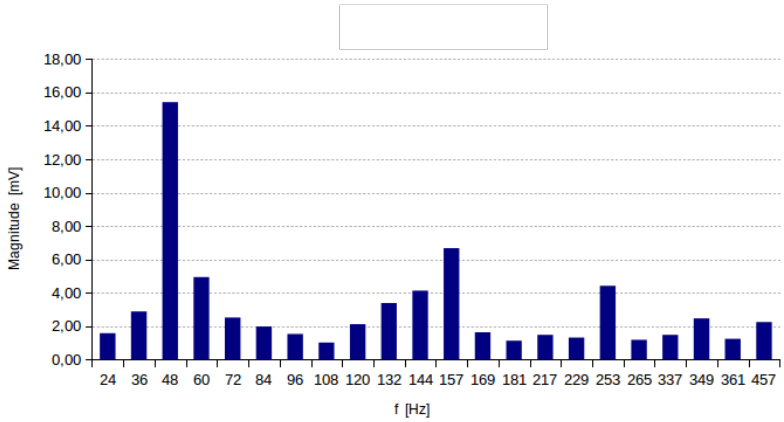


Figure 2.15: Monitor and Drill active: spectrum obtained with the metering system.

Discussion

The proposed metering system can correctly place the harmonics detected in a neighbourhood of the correct frequencies to which they should be located. The phenomenon is linked to the slight variability of the f_s, and the window size, as explained above. However, despite this imprecision margin, it is possible to distinguish between them, the significant harmonics and the correct frequency value to which they refer.

During the validation phase, we also tried to use wider windows. This allowed

Table 2.1: Significant harmonic components of the analyzed loads.

Device	50 HZ	100 Hz	150 Hz	250 Hz	350 Hz	450 Hz
Monitor active	6.35	-	4.73	2.93	2.43	-
Drill stand-by	3.01	-	1.53	-	-	-
Drill active 100%	9.81	-	3.37	-	-	-
Drill + Monitor active	15.41	-	6.67	4.41	2.46	2.24

obtaining a better frequency resolution of the FFT at the expense of processing times. By setting n = 4096, the consumer thread's processing times became higher than those needed to acquire the samples' block. These conditions would not have allowed the system to process the information acquired in real-time. For instance, the plots realized with the measured values produced by the smart metering IoTaaS are quite similar to the spectra shown by the digital oscilloscope. This shows the presence of a disturbance to f ≈ 950 Hz caused by the system itself.

Table 2.1 summarizes the amplitudes associated with the harmonics of various orders for each electrical load produced by the considered electronic devices. Each row of the table (except for the last one) defines a unique *pattern* for identifying the electronic devices. Only the computer monitor presents the *pattern* (fund → 6.35 mV, 3° → 4.73 mV, 5° → 2.93 mV, 7° → 2.43 mV). A well as the drill is characterized by (fund → 108.08 mV, 2° → 10.86 mV, 3° → 4.71 mV). The spectrum generated by several simultaneous loads does not allow the devices' immediate distinction due to the overlap between the harmonics. Still, however, it can be noticed a significant increase in the amplitude of harmonics placed at 50 Hz and respectively 157 Hz which characterize both loads.

2.3.5 Conclusions and Future Work

In this contribution, we developed a smart metering IoTaaS that can recognize non-linear loads generated by electronic devices connected over on a common electrical grid, based on the harmonic spectral content originating from their current absorption calculated using the FFT transform. This has been pursued using the Raspberry Pi 3 device, which, suitably programmed using a container-based microservice, coordinates an ADC chip's acquisition operations and processes the data returned by the FFT algorithm in order to monitor the status of the connected electrical devices.

During the experimental phases, it was possible to identify the characteristic *patterns* for each examined load, which allowed us to recognize the individual power loads connected to the network. The simultaneous use of multiple loads

entails difficulties in distinguishing them due to the overlap between the harmonics. To solve this, in future work, we are planning to use convolutional neural networks (CNNs). Moreover, we would like to extend our efforts to more complicated grid models to get a more complex view of these new evaluation methods' usefulness, mainly from the point of view of distribution network operators.

2.4 IoTaaS-based Gait Assessment for Ataxia

2.4.1 Introduction

Ataxic syndromes are a group of motor disorders due to primary (inherited or idiopathic) or secondary nervous system disorders characterized by poor balance, coordination and gait difficulties. Depending on the clinical progression, ataxic syndrome can be divided in Progressive Ataxias (PAs) and Chronic Ataxias (CAs), encompassing non-progressive or very-slow-progressive entities [53, 54, 55, 56, 57, 58, 59]. PAs are usually neurodegenerative diseases, often with a genetic cause, responsible for highly disabling conditions (e.g., Friedreich's Ataxia, FRDA, and other spinocerebellar ataxias) [56]; CAs, on the other hand, may include secondary forms (e.g., due to vascular causes, trauma, tumour) or inherited neurodevelopmental diseases (e.g., Joubert Syndrome, metabolic or ion-channels diseases), with possible better course and prognosis [54, 55, 57].

Ataxias are still incurable disorders as effective treatments are not available. However, the therapeutic scenario might rapidly change due to the imminence of disease-modifying treatments [60] or other symptomatic interventions (physical therapy or neuromodulation) [61].

The development of effective therapeutic interventions adapted to both PAs and CAs, and the subsequent conduct of clinical studies, however, strictly depends on identifying specific and reliable biomarkers, which allow for early stratification of patients and accurate follow-up both natural than interventional frames. Indeed, the assessment tools to be used as outcome measures for Ataxia are limited and are mainly based on clinical scores, such as the Scale for the Assessment and Rating of Ataxia (SARA) [62], which are inevitably affected by floor and ceiling effects, rater variability [53, 63, 64] and, especially, low accuracy when used in young patients (<10-year-old) [64, 65].

In recent years, new technologies have emerged, including computer interfaces, video games, and wearable sensors, for clinical applications, particularly in the field of neurology and neurorehabilitation, as there have been successful attempts to obtain several reliable, objective, accurate and continuous both in standardized contexts (hospital, laboratory) and at home [66, 67, 68, 69, 70, 71, 72, 73].

Specifically, promising results seem to come from Microsoft Kinect v2 sensor (Microsoft Cop., Redmond, WA, USA), a low-cost RGB-D camera originally manufactured for entertainment, which actually enables a marker-less human motion tracking system [71], particularly useful in movement analysis [66, 67, 68, 69, 70, 71, 72]. Despite some limitations, the use of Kinect v2 in clinical practice is increasing across different neurological disease [74, 75] and innovative protocols for quantitative measurement of Ataxia have been proposed [76].

Significant abnormalities in spatio-temporal gait parameters (reduced speed and stride length, increased base width and greater variability of stride charac-

teristics) are a key feature of ataxic syndromes [77]. Accordingly, gait analysis represents a valuable source for clinical biomarkers in Ataxia, enabling accurate quantitative assessment of patients throughout the disease, from non-manifest to overt phases [78, 79, 80]. However, the gait analysis is currently based on complex and expensive systems available only in a few specialized centres. Conversely, a Kinect-based technology, using machine learning processing methods [81, 82, 83, 84], can lead to a low-cost and easily accessible system [81, 82, 84, 85, 86] for gait analysis [87] in ataxic patients.

This contribution proposes a Kinect-based IoTaaS for gait analysis system to assess ataxic patients, providing the corresponding spatio-temporal gait parameters; besides, we validated its reliability and accuracy compared with the standard motion capture system. Finally, to support this novel tool's potential for clinical applications, we explored clustering algorithms' capability to classify different groups of ataxic patients (PAs vs CAs) and classify different levels of disease severity (Low, Medium or High severity).

2.4.2 Population Definition

The study population was enrolled in the Neurorehabilitation Unit - Department of Neurosciences of the Bambino Gesù Children's Hospital (Rome, Italy) in 2018 and included 51 individuals: 31 patients and 20 healthy subjects (H). H group included sex/age-matched healthy volunteers with no personal/family history of neurological diseases and no clinical examination signs (age 14.12(9.1); 12F/8M). Patients were further divided into a PA group (n=15) and a CA group (n=16) depending on the diagnosis and clinical course.

All patients had genetically confirmed diagnosis (Table 2.2 and Table 2.3) and a routine diagnostic work-up, including general and neurological examination, brain MRI, sensory evoked potentials, nerve conduction study and visual acuity assessment; moreover, they were in follow-up at for at least 2 years, in order to guarantee a correct group classification (Table 2.2 and Table 2.3).

None of the enrolled subjects had relevant cognitive impairment or took psychoactive drugs (other usual medications, such as vitamin or antioxidant). Patients with severe disability, moderate-severe cognitive impairment affecting tests execution, brain and/or cerebellar lesions were excluded. Demographics data were collected for the three groups. Motor disturbances were assessed in patients based on standardized clinical scores:: SARA, 6 Minutes Walking Test (6MWT) and Timed 25 Foot Walk Test (T25FW). Clinical assessment was performed by expert personnel. The research complies with the ethical standards established in the 1964 Helsinki Declaration. All subjects participated voluntarily; subsequently, they or their legally responsible signed the informed consent (the local ethics committee approved the study).

2.4.3 Experiment Approach

Enrolled patients first received a clinical evaluation. Then, all 51 subjects underwent a Kinect-based assessment. Only 31 of them were instead able to undergo gait analysis by standard system. In fact, while standard gait analysis is needed that the subjects autonomously walk, the Kinect recognizes automatically multiple people, thus enabling safety assistance to patients. The order of gait analysis acquisition was randomly assigned to avoid bias due to fatigue or poor collaboration. Likewise, sufficient time of rest was assured after clinical tests. Therefore, to validate the Kinect with the standard motion capture system, we analyzed a subgroup of 31 components obtained from those participants who performed both tests regardless of their group.

2.4.4 Kinect-based IoTaaS Prototype for Gait Assessment

Hardware Design

Motion capture system set-up and protocol Standardized gait analysis was conducted by a twelve-cameras motion capture system (Vicon MX, UK) and two force plates (AMTI, or-6, US). Sampling rates were set at 200 Hz for the motion capture system and at 2 kHz using the two force plates. The two force plates were hidden in the middle portion of a 10 m walkway in order to avoid acceleration and deceleration phases into the gait cycle considered. Assessments were video recorded to assist clinical interpretation of data. The 33 markers were located on the subjects' anatomical landmarks as indicated by the Plug-in-Gait protocol to reconstruct a full-body kinematic and kinetic model.

Kinect-based system set-up and protocol The Kinect v2 was placed on a tripod (tilt angle 0°in front of the participant to obtain the front view and at the height of 1 m. It has been suggested that the gait trace should be between 1.5 m and 3.5 m from the Kinect [85]. Kinect v2 framerate is 30 frames per second. In order to ensure that a minimum of one full gait cycle per leg was captured, we placed the Kinect at a distance of 4.5 m from the starting line and asked the patient to walk barefoot at their self-selected speed up to at the stop line that has been placed at 1.5 m from the Kinect, avoiding from the step cycle to analyze the start and end phases of acceleration and deceleration (Figure 2.16).

Software Design

The system's software side was implemented using a container-based microservice that runs on Raspberry Pi 3 (Raspbian OS) that manages the acquisition and preprocessing and another container-based microservice running on the cloud managing the processing and output phases of the Kinect-based IoTaaS. The container-based approach was adopted to simplify service delivery according to an IoT cloud scenario. Both microservices have been implemented in Python.

Table 2.2: Demographic and clinical parameters of the study population.

					Progressive Ataxia			
Diagnosis	Patient n.	Gender	Age [y]	Age of onset [y]	SARA	Gait score	T25FW	6 MWT
FRDA	PA1	F	8	7	13.5	2	6.63	345.6
FRDA	PA2	F	12	6	7.5	1	5.05	465.0
FRDA	PA3	M	32.3	23	9	3	5.50	400.4
FRDA	PA4	F	14	5	16.5	4	10.04	225
FRDA	PA5	M	15	12	12	3	8.98	369.8
FRDA	PA6	M	8	4	11	2	4.65	464.0
FRDA	PA7	F	16	12	8	2	4.70	516.2
FRDA	PA8	F	9	5	10	2	4.74	450.0
FRDA	PA9	M	19	17	11	3	4.77	448.2
FRDA	PA10	M	32	21	19	6	-	-
ARSACS	PA11	M	36	15	11	4	10.53	-
AT	PA12	F	4	1	9	2	-	-
SCA	PA13	F	22	17	12	3	5.23	469.6
SCA2	PA14	M	14	5	10.5	2	5.80	410.7
ARSACS	PA15	F	10	1.5	11.5	2	4.59	410.7
	mean(st.dev)	**7M/8F**	**17.2 (9.5)**	**10.1 (6.9)**	**11.4 (2.9)**	**2.7 (1.2)**	**6.25 (2.1)**	**414.5 (76.1)**

Age and disease duration are expressed in years; **UCA** = Undiagnosed Cerebellar Atrophy/Hypoplasia; **SECONDARY A.** = Secondary Ataxia due to posterior cranial fossa tumor; **FRDA** = Friedreich's Ataxia; **ARSACS** = Autosomal recessive spastic Ataxia of Charlevoix-Saguenay; **AT** = Ataxia–Telangiectasia syndrome; **SCA** = Spinocerebellar Ataxia; **SCA2** = Spinocerebellar Ataxia type 2; **JOUBERT S.** = Joubert syndrome; **PMM2, ADCK3, ITPR1** = are the name of the genes responsible for the disease; **T25FW** expressed in seconds and 6 MWT expressed in meters; **n** = number; other abbreviations are spelled out in the text.

Table 2.3: Demographic and clinical parameters of the study population.

					Non-progressive Ataxia			
Diagnosis	Patient n.	Gender	Age [y]	Age of onset [y]	SARA	Gait score	T25FW	6 MWT
ADCK3	CA1	F	8	6	10	2	5.27	507.6
ADCK3	CA2	F	13	2	10	2	6.30	468.4
UCA	CA3	F	14	10	6.5	1	5.30	438
JOUBERT S.	CA4	M	11	1	6	1	5.81	585
JOUBERT S.	CA5	F	15	0.5	5	2	5.23	481.5
ADCK3	CA6	F	25	7	10.5	2	7.23	489
ADCK3	CA7	M	7	3	15	2	5.64	446
ADCK3	CA8	M	10	3	12.5	2	7.11	436.8
JOUBERT S.	CA9	F	13	0.5	11	2	5.03	295.6
ITPR1	CA10	F	8	0.5	11	2	5.95	432.6
ADCK3	CA11	M	8	5	3.5	1	5.17	390
UCA	CA12	M	13	1	26.5	6	9.46	-
UCA	CA13	M	8	1	22.5	6	9.40	-
PMM2	CA14	M	17	1	12	2	4.93	548
SECONDARY A.	CA15	F	14	11.5	11	2	6.42	416.7
JOUBERT S.	CA15	M	10	0.5	8	3	6.69	321.3
	mean(st.dev)	**8M/8F**	**12.6 (4.5)**	**3.3 (3.6)**	**11.3 (5.9)**	**2.4 (1.5)**	**6.3 (1.4)**	**446.9 (78.3)**

Age and disease duration are expressed in years; **UCA** = Undiagnosed Cerebellar Atrophy/Hypoplasia; **SECONDARY A.** = Secondary Ataxia due to posterior cranial fossa tumor; **FRDA** = Friedreich's Ataxia; **ARSACS** = Autosomal recessive spastic Ataxia of Charlevoix-Saguenay; **AT** = Ataxia–Telangiectasia syndrome; **SCA** = Spinocerebellar Ataxia; **SCA2** = Spinocerebellar Ataxia type 2; **JOUBERT S.** = Joubert syndrome; **PMM2, ADCK3, ITPR1** = are the name of the genes responsible for the disease; **T25FW** expressed in seconds and 6 MWT expressed in meters; **n** = number; other abbreviations are spelled out in the text.

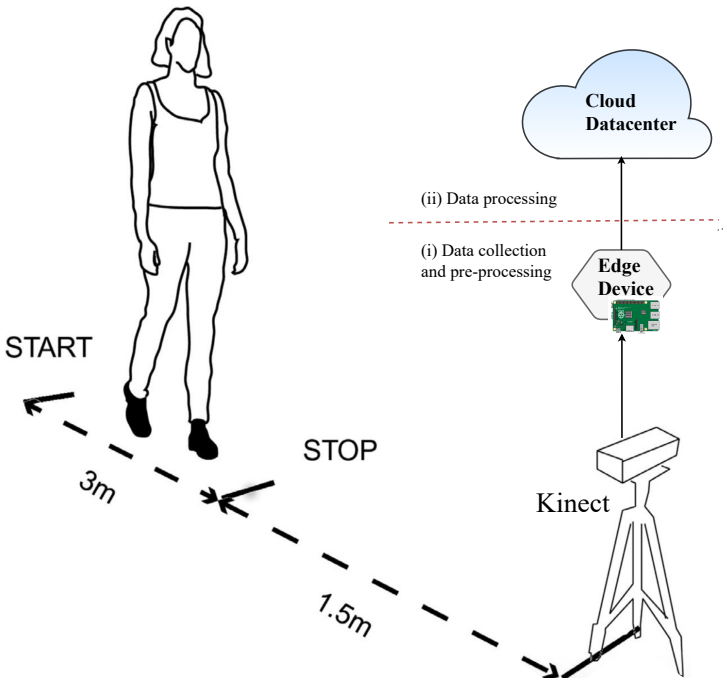


Figure 2.16: Kinect-based IoTaaS. From the starting line to the Kinect there is a total distance of 4.5 m. The stopping line is placed 1.5 m from the Kinect in order to ensure the tracking of the skeleton. The 3 m walkway is enough to guarantee a gait cycle.

Data acquisition and pre-processing The 25 anatomical landmarks, including the spine base, left/right hip, left/right knee, and left/right ankle, and the left/right foot (representing the toes), were collected.

Kinect's markers time series were smoothed with a 4th order Savintzky–Golay filter with a time window of 0.96 s. Although the cut-off frequency of about 1.5 Hz is low compared to motion analysis standards, this value has sufficiently addressed the low accuracy of the Kinect v2 sensor. While the standard motion capture system uses the force platform to identify the gait events that define a cycle, the Kinect does not offer automatic data about the timing of interaction between the floor and the feet. Other methods have been proposed to estimate the timing automatically from the limb displacement, velocity or acceleration. To identify gait events, we examined the evolution of the horizontal displacement differences between the hip and foot of the Kinect markers. In particular, we identified the Initial Contact (IC) and the Final Contact (FC) as the maximum and the Foot Off (FO) as the minimum.

We tested this algorithm's accuracy measuring the mean absolute error of gait events recognition in three standard gaits with respect to the gait events recognized from the system that uses the force platforms.

Motion capture system data have been computed by Nexus 2.7 (Vicon, Oxford UK) following the c3d format convention. The list of variables that have been selected to detect the main strategies adopted by the patients while walking is described in Table 2.4.

To test if there was a significant difference between the spatio-temporal parameters obtained from the two measures, we examined correlation and agreement tests. In other words, we examined the Pearson coefficient (r), hypothesizing a linear relationship between the same parameter but acquired with the two systems. Moreover, we used, for each spatio-temporal parameter, the Bland–Altman test to quantify the agreement of the Kinect measures by comparing it with the standard motion capture system, which is the Vicon system, looking at possible fixed or proportional bias and at the Limits of Agreement (LoA) [88, 89, 90].

We applied the Kolmogorov–Smirnov test to study the normality distribution of the difference between the two measures of each parameter. Fixed bias was evaluated by a 1-sample t-test to see if the mean value of the difference differed significantly from 0. Proportional bias was assessed by linear regression analysis using the difference between measurements as the dependent variable and the gold standard as the independent variable. If this test is statistically significant, it can be shown that there is or that there is not proportional bias [90].

In the case of bias, we applied the appropriate corrections. In particular, the fixed biases were removed and when a proportional bias was shown, the parameter was log-transformed [90]. This comparative analysis was conducted considering a subgroup of 31 participants who had made both acquisitions with the Kinect and the motion capture system. We examined possible correlations using the Spearman test between all the corrected variables and the SARA and using the Pearson test between all the variables and the 6MWT or T25FW.

In particular, we looked at correlations with the parameters obtained by the Kinect corrected. Due to the multiple correlation tests, we applied a correction to the alpha value of 0.05/9, which is 0.006. This analysis was conducted considering all ataxic patients as a unique group (CA+PA groups).

Data processing: machine learning-based classification We transformed the score (from 0 to 6) assigned by the clinician into the gait task of the SARA scale in three main classes functional impairment: Low (from 0 to 1), Medium (from 2 to 3), High (from 4 to 6). The final dataset for machine learning models contained 51 subjects: 24 Low, 21 Medium, 5 High and all kinematic features collected with the Kinect. After the new categorization, the new gait classes were predicted by comparing the overall accuracy and Cohen's kappa value [91] of five most common classifiers, e.g., Multilayer Perceptron (MLP), Naive Bayes (NB) [92], k-Nearest Neighbors (k-NN) [93], random forest (RF) decision tree and Support Vector Machine (SVM). All models and classifiers presented here were obtained using the caret (version 6.0-84), randomForest (version 4.6-14) and e1071 (version 1.7-2) packages for R using default parameters optimized ad

Table 2.4: List of the spatio-temporal parameters and definitions, considered in this study to assess Ataxia gait.

Variable name	Variable definition
cadence [steps/min]	The rate at which a person walks, expressed in steps per minute.
speed [m/s]	Mean velocity of progression.
stride length [m]	Longitudinal distance from one foot strike to the next one of the same limb.
stride time [s]	Total time that begins with initial and final contact of the same limb.
base width [m]	Transversal distance between the right and left foot.
step length [m]	Longitudinal distance from one foot strike to the next one.
stance phase [% cycle]	Percentage of gait cycle that begins with initial contact and ends at toe-off of the same limb.
swing phase [% cycle]	The period during which the foot is in the air for the purpose of limb advancement.
double support phase [% cycle]	Time in which both feet are in contact with the floor.

hoc for each model. For each algorithm, we performed 10-fold cross-validation, using 90% for training and the remaining 10% for testing and measure accuracy. This method has been shown to work best in reducing bias in the resulting classifier [94].

The selected machine learning models were also adopted to classify CA, PA and H subjects. In this case, the dataset contained 51 subjects labelled in this way: 16 CA, 15 PA and 20 H. Finally, we used the Principal Component Analysis (PCA) to reduce the Kinect dataset's multiple dimensionalities given by all the 9 kinematic features and visualize classes. The 1st and 2nd principal components were used for the scatter plot and observed the similarity of gait classes, respectively, low, medium, high and PA, CA, and H. This analysis was conducted considering the three groups CA, PA and H.

2.4.5 Results

Gait events, identified by the algorithm that we used with the Kinect, were compared with those of the standard motion capture system. The mean absolute error of the gait events recognition algorithm was 9 samples.

Relationship Between the Measures

Pearson test revealed that there is a linear relationship between each parameter acquired with the two methods (see Table 2.5, first two columns). Pearson coefficient shows a strong correlation of cadence ($r = 0.81$, $p < 0.0001$), speed ($r = 0.80$, $p < 0.0001$), stride length ($r = 0.81$, $p < 0.0001$), step length ($r = 0.80$, $p < 0.0001$) and stride time ($r = 0.77$, $p < 0.0001$). There is still a correlation but moderate between the two acquisitions of base width ($r = 0.44$, $p = 0.01$), stance and swing phase (respectively $r = 0.45$, $p = 0.01$ and $r = 0.46$, $p = 0.01$).

Agreement Between the Measures

The results obtained from the Bland–Altman test are reported in Table 2.5. The fixed bias is the discrepancy between the two measures. The proportional bias is a slope that resulted from the regression analysis. The reproducibility coefficient is equal to 1.96 × the standard deviation and it is used to assess the LoA. All the spatio-temporal parameters except speed have a significant fixed bias ($p<0.05$). No proportional bias was found with the exception of base width ($p = 0.003$, slope = -0.55).

Correlation with Clinical Tests

We found a significant inverse correlation between speed (rho = -0.54, p-value = 0.001), stride length (rho = -0.58, p-value = 0.0006) and step length (rho = -0.61, p-value = 0.0002) and SARA total score. We found a significant inverse correlation between speed ($r = -0.74$, p-value < 0.0001), stride length ($r = -0.55$, p-value = 0.003), swing phase ($r = -0.55$, p-value = 0.003), cadence ($r = -0.61$, p-value = 0.001) and the T25FW test. Stride time ($r = 0.73$, p-value < 0.0001) and stance phase ($r = 0.56$, p-value = 0.003) and double support ($r = 0.58$, p-value = 0.002) directly correlate with the T25FW test. Moreover, we found a direct correlation between 6MWT and speed ($r = 0.58$, p-value = 0.002) and a negative correlation with stride time ($r = -0.58$, p-value = 0.002).

Classification

The classification respectively of three new gait impairment classes (Low, Medium, High) and PA, CA, H, was carried out performing the 10-fold cross-validation of the whole Kinect dataset and comparing the best classification performance of the selected machine learning models. The mean and interquartile range (IQR) of accuracy and Cohen's Kappa values, sensitivity and specificity for each class have been reported in Table 2.6 and Table 2.7. The PCA scatter plot of 1st and 2nd components reduces the multidimensionality of Kinect measures and provides a visualization of data by each subject respectively for gait score classification (Figure 2.17 (a)) and recognition of PA/CA/H (Figure 2.17 (b)).

Table 2.5: Relationship between measures of each parameter acquired with the Kinect system and those acquired with the motion capture system, reported in the first two columns. Bland–Altman test and correlation analysis between the parameters measured with the two systems.

Measure	r	p-value	Fixed bias	Proportional bias	LoA	RPC	Kinect Mean [CI]	Vicon Mean [CI]
Cadence [steps/min]	0.82	<0.0001	20	0.31	[-12 51]	32	114.81 [82.81 146.81]	134.61 [83.02 86.21]
Speed [m/s]	0.80	<0.0001	0.04	0.31	[-0.3 0.37]	0.34	1.01 [0.68 1.34]	1.05 [0.51 1.59]
Stride length [m]	0.81	<0.0001	-0.13	-0.05	[-0.33 0.08]	0.20	1.06 [0.76 1.36]	0.93 [0.59 1.28]
Base width [m]	0.44	0.01	-0.04	-0.55	[-0.13 0.06]	0.09	0.19 [0.11 0.28]	0.16 [0.07 0.25]
Step length [m]	0.80	<0.0001	-0.05	-0.03	[-0.16 0.06]	.11	0.53 [0.38 0.68]	0.48 [0.30 0.67]
Stride time [s]	0.77	<0.0001	-0.14	0.11	[-0.4 0.13]	0.27	1.07 [0.78 1.35]	0.93 [0.51 1.35]
Stance phase [%]	0.45	0.01	-2.3	0.09	[-11 6.7]	9.1	60.7 [56.51 64.91]	58.4 [48.21 68.53]
Swing phase [%]	0.46	0.01	-0.04	0.10	[-6.7 11]	9.1	39.29 [35.07 43.52]	41.63 [31.47 51.79]
Double support [%]	0.58	<0.0001	-0.03	0.56	[-11 5.3]	7.9	10.63 [7.10 14.16]	7.99 [-1.42 17.41]
	mean (st.dev)	**8M/8F**	**12.6 (4.5)**	**3.3 (3.6)**	**11.3 (5.9)**	**2.4 (1.5)**	**6.3 (1.4)**	**446.9 (78.3)**

LoA = Limits of Agreement; **RPC** = Reproducibility Coefficient (1.96 × standard deviation); **CI** = Confidence Interval; **r** = Pearson coefficient a p-value<0.05 given by the respective statistical test.

Table 2.6: Accuracy & Cohen's Kappa value [mean (IQR)%], for each tested classifier respectively for gait score (Low/Medium/High) and Ataxia Groups (PA/CA/H). Moreover, Sens.x|Spec.x are respectively the sensitivity and specificity of each selected class (x) respect to all the other classes.

Classifier	Accuracy & Kappa (Low/Medium/High)	$Sens_{Low}$ | $Spec_{Low}$	$Sens_{Med}$ | $Spec_{Med}$	$Sens_{High}$ | $Spec_{High}$
MLP	67.5 (19) & 57 (12.1)	0.58 | 0.48	0.62 | 0.60	0.63 | 0.60
NB	78.2 (20) & 61.5 (37)	0.65 | 0.58	0.63 | 0.64	0.68 | 0.69
k-NN	68.5 (2.7) & 43.7 (5.8)	0.61 | 0.60	0.61 | 0.62	0.64 | 0.58
RF	83.2 (0.9) & 72.8 (1.8)	0.75 | 0.72	0.70 | 0.66	0.78 | 0.80
SVM	90.4 (19) & 82.8 (34)	0.88 | 0.80	0.86 | 0.80	0.97 | 0.94

MLP = Multilayer Perceptron; **NB** = Naive Bayes; **k-NN** = k-Nearest Neighbors; **RF** = Random Forest; **SVM** = Support Vector Machine.

Table 2.7: Accuracy & Cohen's Kappa value [mean (IQR)%], for each tested classifier respectively for gait score (Low/Medium/High) and Ataxia Groups (PA/CA/H). Moreover, Sens.x|Spec.x are respectively the sensitivity and specificity of each selected class (x) respect to all the other classes.

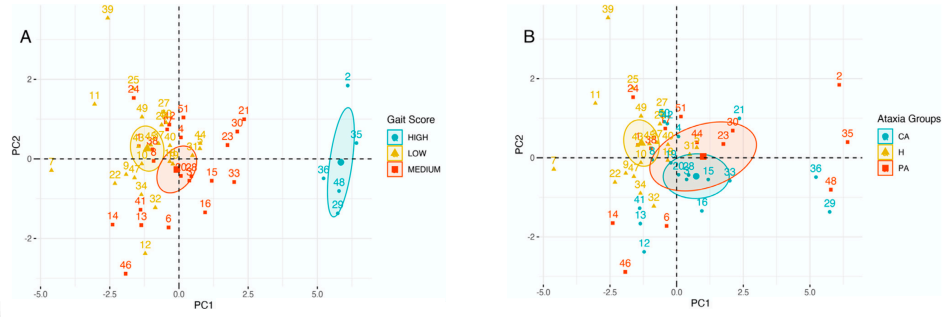
Classifier	Accuracy & Kappa (Low/Medium/High)	$Sens_{Low}$ | $Spec_{Low}$	$Sens_{Med}$ | $Spec_{Med}$	$Sens_{High}$ | $Spec_{High}$
MLP	55.1 (5.5) & 30.4 (10.2)	0.50 | 0.51	0.53 | 0.49	0.57 | 0.44
NB	51.1 (23.6) & 24.3 (34.6)	0.49 | 0.47	0.51 | 0.47	0.53 | 0.50
k-NN	45.1 (10) & 14.2 (10.4)	0.46 | 0.48	0.52 | 0.54	0.51 | 0.45
RF	58.9 (12.12) & 36.5 (15.8)	0.60 | 0.62	0.57 | 0.62	0.58 | 0.59
SVM	68.6 (3.4) & 49.7 (8.95)	0.62 | 0.63	0.64 | 0.59	0.64 | 0.57

MLP = Multilayer Perceptron; **NB** = Naive Bayes; **k-NN** = k-Nearest Neighbors; **RF** = Random Forest; **SVM** = Support Vector Machine.

2.4.6 Discussion

This study aimed to evaluate a cost-effective Kinect-based IoTaaS for gait analysis in young patients with Ataxia to provide a reliable definition of spatio-temporal gait parameters [95, 96]. Moreover, we applied a series of machine learning algorithms to classify subjects depending on the disease severity or the ataxia subtype (CA or PA) to estimate a potential clinical use of the Kinect-based gait analysis.

We demonstrated that the Kinect was able to identify specific gait events, whose computation is necessary to analyze spatio-temporal parameters. Gait events are the interaction of the foot with the ground. The spatio-temporal parameters' accuracy can vary remarkably depending on the algorithms used to detect the gait events or the low frequency of acquisition of the Kinect. We found a mean absolute error of gait events recognition of 9 samples. The



(a) PCA with cluster of gait score (low-medium-high).

(b) PCA with cluster of Ataxia groups (PA-CA-H).

Figure 2.17: PCA results. Data are projected in the first two PCs. Observations are colored in green, yellow or red representing, respectively, high, low, medium or PA, CA or H.

appropriateness of this error is due to the frequency of acquisition. Despite such low frequency (30 frames per second), the use of Kinect is spreading across different neurological diseases.

[81, 82, 83, 84, 85, 86, 87], supported by the numerous advantages of the methodology. Compared to standard gait analysis systems, Kinect is a low-cost marker-less tracking system that simplifies acquisitions and reduces costs, allowing wider use in clinical settings. Besides, it can acquire motion from more than one subject, thus enabling assistance and evaluation of non-autonomously walking patients.

To demonstrate the validity and quality of Kinect acquisition, we compared its measures with those of a standard motion capture system for gait analysis, observing both the agreement and the relationship between the same parameters acquired with the two systems. The Bland–Altman test quantifies the agreement between two measurements by studying the mean difference and constructing the LoAs. Ideally, we would expect that the average of the difference would be equal to zero. This has been easily tested by a 1-sample t-test used to test if the mean value of the difference differs significantly from zero. We found fundamental information on the presence of a fixed bias in all the parameters acquired with the Kinect, except for speed.

We exceeded this by subtracting the fixed bias from the measured Kinect parameter. The proportional bias affects a measure when the difference in values resulting from the two methods increases or decreases in proportion [97]. A proportional bias affected the base width measure obtained with the Kinect system. A similar deficiency in base width measurement has already been documented in healthy subjects [98]. A possible explanation of this could be that base width is the only parameter computed in the x-axis and, while Kinect uses a depth camera to measure distances on the z-axis, x- and y-axes are computed

with the RGB camera. Therefore, different types of transformations are needed to obtain the joints' positions in the camera space.

LoAs are often challenging to understand because they are highly context-dependent. It is critical to assess if the LoAs are clinically acceptable. Comparing with other studies, we found that temporal parameters' LoAs are acceptable while spatial parameters are characterized sometimes by larger but still acceptable limits [99, 100, 101].

Although the biases can be overcome, the results obtained show that only base width had a proportional bias, but all the parameters showed a fixed bias except the speed. The presence of different biases means that the spatio-temporal parameters measured by the Kinect cannot be used interchangeably with those parameters acquired with a standard motion capture system used for gait analysis in clinical practice. However, we found linear relationships between all the parameters acquired with the two systems, suggesting that a gait acquired with the Kinect cannot be used for gait analysis as a replacement for a standard motion capture system but can still provide essential clinical information. Indeed, we observed significant correlations between the SARA score and the speed, the stride length and the step length parameters acquired with the Kinect, which are consistent with the correlations found between the SARA score and the parameters acquired with the standard motion capture system [78].

Finally, we used all the Kinect dataset (from 51 subjects, patients + H) to test the ability of obtained parameters, that is, the technology-based biomarkers, to differentiate clinical conditions (PA, CA, H) and classify motor disability, such as to predict a potential use of Kinect-based evaluation for early patients stratification. The SVM algorithm gave the best results in this examination, showing an overall accuracy of gait score classification of 90.4%, a sensitivity greater than 85% and a specificity of 80% and more, for each class respectively, as reported in Table 2.6 and Table 2.7 (accordingly Figure 2.17 (a) shows separated clusters); conversely, the recognition accuracy of PA/CA/H was about 70%, the sensitivity and the specificity of each class were respectively about 63% and 60% (overlapped clusters are shown in Figure 2.17 (b)), consistently with previous studies disclosing some overlap in gait patterns of different clinical conditions [102]. These results, therefore, suggest that a machine learning approach to measures derived from a Kinect-based evaluation could provide an immediate index of Ataxia severity, useful for clinical applications, although the contribution of other disturbances (movement disorders, spasticity, weakness), the final result was not isolated in this context.

Finally, Kinect-based assessment has been successfully used on a population of young subjects, suggesting that this methodology can also be applied to pediatric patients, which is known to be poorly collaborative with standard evaluation systems, as our previous study initially disclosed [76].

2.4.7 Conclusions

Our findings pave the way to developing a novel and cost-effective Kinect-based IoTaaS for gait assessment in ataxic patients capable of providing technology-based biomarkers useful for clinical application in the field of Ataxia. Indeed, we confirm that the Kinect technology can reliably derive space-time gait parameters, as the comparison with the standard motion capture system showed. In particular, our data suggest that speed, stride length and step length are the most promising parameters for evaluating and following-up patients into a clinical setting, relying on Kinect-based gait analysis. Although further confirmatory studies are now necessary, possibly using two or more Kinects in the setup, this preliminary experience indicates a new tool for quantitative assessment of Ataxia and stratification of patients (even young), which are critical unmet needs in this field.

Indeed, a Kinect-based IoTaaS could represent a potential tool that allows assessing the progression of gait even in the phases of the disease in which assistance is needed and permitting an intensive home follow-up, beneficial for some advanced patient. Besides, the SVM algorithm demonstrated the accuracy of such a system in distinguishing different clinical conditions, further highlighting the potential applications of this novel tool; however, the main limitations of such methodology should be taken into consideration (restricted tracked area, frontal view acquisition, low sampling frequency, which collectively preclude monitoring in real-life or unconstrained context) [103].

Overall, these findings encourage the implementation of fully technology-based systems for objective assessment of Ataxia, even in remote contexts, as we conceptualized in SaraHome [72].

3 Network Management

In an Internet f Things (IoT)-based system, billions or even trillions of devices are connected to the global network infrastructure, contributing to the big data phenomenon. Aside from data volume, the velocity, variety, and veracity of these data will significantly burden traditional networking infrastructures. Cloud Computing (CC) has been integrated with IoT to address limitations in existing IoT networks (e.g., computing and storage resources). However, cloud-centric IoT systems might not be suitable for delay-sensitive and computationally-intensive applications due to end-to-end latency, resource availability, bandwidth. Consequently, there has been a shift to Edge Computing (EC), where computation and storage resources are pushed closer to data sources. However, massive migration between edge and cloud imposes communication delays and results in latency, bandwidth, and inefficient energy consumption issues. Thus, there have been attempts to explore the potential of network softwarization paradigms, such as Software Defined Networks (SDN) and Network Function Virtualization (NFV). Furthermore, things in next-generation IoT systems are expected to be extremely heterogeneous in platforms, resources and connectivity. As a result, the amalgamation of SDN and NFV in Cloud-to-Thing (C2T) interplay is, undeniably, promising in improving the Quality of Service (QoS) for complex IoT-driven applications. This chapter seeks state-of-the-art approaches, methodologies, key technologies, and typical IoT applications benefiting from edge cloud in the design, development, deployment, and innovative use of SDN and NFV edge-cloud integration in next-generation IoT infrastructures.

3.1 Introduction

Currently, around 10% of enterprise-generated data is created and processed outside a traditional centralized data centre or cloud. By 2022, Gartner predicts this figure will reach 50% [104]. Simultaneously, the exponential increase in the number of users and the proliferation of the Internet of Things (IoT) devices and associated data streams has put significant stress on the network and edge layer, leading to the emergence of new data confidentiality problems and network performance.

In recent years, the ongoing evolution of Cloud Computing (CC) and the adoption of the microservices-based architecture led to a drastic change in application building, deployment, and delivery to tackle those problems. Microservices-based architecture is tailored to today's software, which requires a high degree

of flexibility and dynamism. Therefore, it combines many sources of information, devices, applications, services, and distributed microservices DμS, into a flexible architecture where applications extend over multiple endpoint devices and coordinate with each other to produce a continuous digital experience.

However, one of the significant issues for applying the microservice architectural style in cloud/edge environments is represented by the network configuration over the fog. Each specific μS or μS requires particular network capabilities that can not be easily carried out by configuring the physical network due to the complexity of existing systems' infrastructures.

This problem can be solved through network softwarization or virtual network technologies that enable creating several virtual overlay networks that meet the requirements of each specific μS and DμS. Network softwarization offers many advantages for Information and Communications Technology (ICT) operators, as demonstrated by the recent keen interest of industry and academics regarding the virtualization of telecommunication networks for CC and Edge Computing (EC) infrastructures [105]. This chapter's main contribution is to provide a targeted analysis that can help cloud/edge architects choose the most suitable overlay network solution. Specifically, this chapter's objective is to propose an innovative system able to face softwarized network connectivity problems for the execution of distributed microservices. In particular, we have created an innovative solution based on Kubernetes, i.e., a portable, extensible open-source platform for managing containerized microservices that facilitate both declarative configuration and automation, to which various overlay network technologies have been associated, namely Open Virtual Networking (OVN), Weave Net and Flannel. We deployed two different distributed microservices-based respectively on the File Transfer Protocol (FTP) and Constrained Application Protocol (CoAP). Then, to quantify the implemented system's network performances, we realized a benchmark based on the transfer times registered using each implemented microservice.

Experimental results compare the different overlay network technologies' performances by considering Cloud-to-Cloud, Cloud-to-Edge, and Edge-to-Edge scenarios. Thus, analyzing the deployed system's performances, we noticed that the OVN based solution is better than others. Indeed, it presents lower execution time and offers more functionalities for managing microservices geographically distributed. Furthermore, independently of their performance analysis, choosing the best network overlay is not an easy task because it mostly depends not only on the user's needs but also on the scenario specifications.

The rest of the chapter is organized as follows. Section 3.2 highlights several relevant works form literature. Section 3.3 discusses the motivation behind this contribution. Section 3.4 provides an overview of the analyzed overlay network technologies. Our reference architecture based on Kubernetes for managing microservices in both cloud and edge layers is discussed in Section 3.5. Section 3.6

provides the implementation workflow for the proposed system. Experimental results comparing the different overlay network technologies in Cloud-to-Cloud, Cloud-to-Edge, and Edge-to-Edge environments are discussed in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.2 Related Work

In recent years, several scientific works have been proposed regarding networking in CC/EC. Hence, communication networks such as SDN and NFV are the key enabling technology that provides ample opportunities for highly flexible network operations in this context [106, 107].

Reliable design for virtual network requests with location constraints in Edge-of-Things (EoT) is proposed in [108]. In particular, the problem of mapping multiple Virtual Networks (VNs) with geographic location constraints onto a physical network is studied while considering the survivability and reliability requirements of each VN request in EoT-based data centres. In [109], the authors propose PerSoNet, a Virtual Private Network (VPN) that automatically creates and manages private, authenticated overlay links across personal devices of social network peers and automatically manages SDN rules in software switches for packet forwarding, name resolution and mapping (for IP addresses and DNS names), and device network access control.

The integration of SDN with IoT, edge, and cloud computing for dynamic distribution of IoT analytics and efficient use of network resources, is proposed in [110]. In particular, an experimental IoT-aware multilayer transport SDN and edge/cloud orchestration architecture that deploys IoT-traffic control and congestion avoidance mechanisms for the dynamic distribution of IoT processing to the edge of the network is presented based on the actual network resource state. [111] proposes a use case to illustrate the concept of NFV as a new paradigm in designing telecommunication networks. Notably, the authors implement an overlay network between vehicles and the cloud, based on distributed Docker containers, orchestrated by Open Baton. A V2V data offloading for cellular networks based on SDN inside a mobile EC architecture is presented in [112]. A study on joint computation offloading and resource allocation optimization in heterogeneous networks with mobile EC is presented in [108]. A novel collaborative vehicular EC framework is presented in [113]. In [114], the authors attempt to introduce a new concept of an ecosystem for Multi-access Edge Computing (MEC) based on the combination of ultra-broadband mmWave communications.

The task offloading problem in an ultra-dense network aiming to minimize the delay while saving the battery life of user's equipment leveraging SDN is investigated in [115]. In [116], an SDN-based edge/cloud interplay is presented to handle big data streaming in the industrial IoT environment. In particular, a multi-objective evolutionary algorithm using Tchebycheff decomposition for

flow scheduling and routing in SDN is presented. An EC-aware Non-Orthogonal Multiple Access (NOMA) technique can enjoy uplink NOMA's benefits in reducing mobile EC users' uplink energy consumption. The collaborative computation offloading for multi-access EC over fibre-wireless networks is presented in [117], presenting an approximation collaborative computation offloading scheme and a game-theoretic collaborative computation offloading scheme. A full-duplex aided user virtualization for mobile EC in 5G networks is presented in [118]. In particular, a novel framework with a user virtualization scheme in the SDN virtualization cellular network is discussed, in which radio resources are virtualized along with computation and storage resources. An integrated framework that enables a dynamic orchestration of networking, caching and computing resources to improve the performance of applications for smart cities is discussed in [119]. A virtualized heterogeneous network framework aiming at enabling content caching and computing is discussed in [120].

Differently from the previous scientific works, in this paper, we test and compare different network overlay technologies in a real cloud/edge system based on the Kubernetes platform.

3.3 Motivation

EC is a method to optimize CC systems by performing data processing at the network's edge, closer to users. It allows minimizing the communication bandwidth needed between IoT devices and the central cloud datacenter by performing analytics and knowledge generation at or near the data source. Doing this computation closer to the edge of the network, organizations can analyze essential data in near and real-time - a need for organizations across many industries, including manufacturing, healthcare, telecommunications and finance. EC reduces latency from a networking point of view because data do not have to traverse the whole network to reach the central cloud for processing, but only important data are sent over the network.

Figure 3.1 illustrates a scenario including both cloud and edge layers interconnected through the Internet. Such a scenario makes massive use of both hypervisor and container virtualization technologies to optimize virtual resources. The Cloud Layer includes several Cloud Regions CR_i, $i = 1, ..., L$ representing cloud systems deployed in different geographical areas. By definition, each CR_i exploits the hypervisor virtualization to manage several Virtual Machines VM_j, $1, ..., M$ and in turn, each VM_j exploits the container virtualization to run several Containers C_k, $1, ..., N$. In the end, each container C_k runs a specific microservice μS_k, $k = 1, ..., N$. The Edge Layer includes several Edge Regions ER_x, $x = 1, ..., O$ representing edge systems deployed in different geographical areas. Each ER_x can manage several Edge Devices ED_y, $y = 1, ..., P$, e.g., Single Board Computers (SBC) (such as Arduino, Raspberry), smartphones, and tablets. Each ED_y,

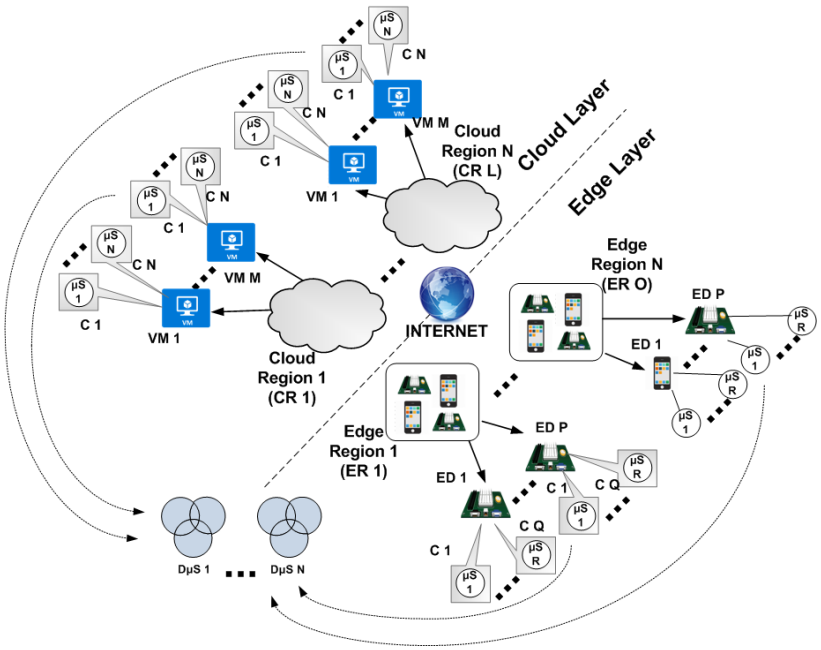


Figure 3.1: Reference scenario.

according to the type of device, can run either several Containers C_z, $z = 1, ..., Q$, such as in the case of Raspberry Pi 3 Model B+ [121], each one running a specific microservice μS_z, $z = 1, ..., Q$ or directly μS_z, such as in the case of devices that do not have the hardware/software capabilities to run a container engine. Also, several microservices μS_k and μS_z can be interconnected to arrange distributed microservices $D\mu S_l$, $l = 1, ..., R$ representing mesh-up services and applications including both central cloud and edge capabilities.

To optimize microservices, it is required to configure *"mesh networks"* of micro data centres that process or store critical data locally. Each microservice requires its specific network capabilities: for example, a CoAP real-time video streaming microservice will require higher throughput compared to an FTP microservice downloading batch data from the central cloud data centre. The possibility of changing the physical network configuration to meet the requirements of microservices is not so feasible due to the high number of microservices that quickly can appear and disappear and due to the complexity of the underlying physical networking infrastructure. A possible solution consists of adopting network softwarization or virtual network technologies that enable creating several virtual overlay networks fitting each specific microservice's requirements. Currently, many network softwarization solutions have appeared on the market. The question is: *which of them is the best one?* This contribution aims to provide an answer to this question considering three of the major virtual overlay network technologies: OVN, Weave Net, and Flannel. To the best of our knowledge,

a scientific work that compares such softwarized network technologies in a cloud/edge environment has not been proposed yet. To overcome such a gap, we will compare these technologies considering Cloud-to-Cloud, Cloud-to-Edge, and Edge-to-Edge communication scenarios.

3.4 Overlay Network Technologies

This section provides an overview of the three overlay network solutions assessed in our cloud/edge system testbed.

3.4.1 Open Virtual Networking

OVN, a sub-project within Open vSwitch (OvS), was announced in early 2015 and recently released the first production-ready version, version 2.6. OVN enables support for virtual network abstraction by adding native OvS support for L2 and L3 overlays and extends the existing OvS capabilities by adding native support for virtual network abstractions and security groups. OVN can be used as a plug-in for OpenStack, Docker and Kubernetes. OVN features include:

- Virtual networking abstraction for OvS, implemented using L2 and L3 overlays, but can also manage connectivity to physical networks;
- Flexible ACLs implemented using flows;
- Software-based L2 gateways;
- Networking for both VMs and containers, without the need for a second layer of overlay networking;
- Native support for NAT and load balancing;
- Native support for distributed L3 routing using OvS flows, with support for both IPv4 and IPv6;
- L3 gateways from logical to physical networks.

In terms of L3 features, OVN provides a so-called distributed logical routing. L3 features provided by OVN are not centralized, so each host executes L3 functions autonomously.

OVN allows users to control cloud network resources by connecting groups of Virtual Machines (VMs) or containers into private L2 and L3 networks quickly and without the need to provision Virtual Local Area Networks (VLANs) or other physical network resources.

3.4.2 Weave Net

Weave Net is a powerful cloud-native networking tool that creates virtual networks for connecting Docker containers across multiple hosts. It also enables the automatic service discovery of the resources. Weave Net has two different connection modes:

- *sleeve* that implements a User Datagram Protocol (UDP) channel to traverse Internet Protocol (IP) packets to containers. The main difference between Weave Net sleeve mode and Flannel UDP backend modes is that Weave Net treats all the container's packets as a unique total packet containing all those packets and transfer them via UDP channel, so Weave Net sleeve is technically faster than UDP backend mode;
- *fastdp* mode that implements also a solution based on VxLAN.

Weave Net creates a fully L2 network, including Ethernet's multicast packet replication to address all network nodes. In multicast communication, the data incoming from a single source is spread simultaneously to multiple sources; this is useful for communicating and sharing data between multiple endpoints. By integrating the Weave Net plug-in with Kubernetes, the users can create a micro-SDN and multicast information to a large number of containers simultaneously. Thanks to the micro-SDN capability, Weave Net offers containerized applications with a simple network and full-service discovery without any configuration, coding, or external dependency.

3.4.3 Flannel

Flannel is a networking solution for Kubernetes developed by CoreOS. It can be used as an alternative to existing SDN solutions. Flannel works by assigning each host an IP subnet used by Docker to allocate IPs for each container. Flannel holds two different network models:

1. *UDP backend*, which is a simple IP-over-IP solution. It uses a TUN device to encapsulate each IP fragment in a UDP packet, forming an overlay network. The UDP encapsulation adds 50 extra bytes to the frame.
2. *Virtual Extensible LAN (VXLAN)* is faster than the UDP backend. VXLAN extends the data link layer (OSI L2) by encapsulating MAC-based L2 Ethernet frames with network layer (OSI L4) UDP packets. From an overhead encapsulation perspective, VXLAN is efficient when compared to UDP.

To integrate Flannel with Kubernetes, each node must get its subnet through Flannel service - *flanneld*. At this point, Docker also needs to be configured to use the subnet created by Flannel. To maintain a mapping between the allocated

subnet and the IP address, Kubernetes runs on each host, an *etcd* server and a flannel service. The *kube-apiserver* component can get IP mappings and assign service IPs accordingly. Subsequently, Kubernetes will create iptables rules, which will allocate static endpoints and load balancing. If the node fails or the Pod is restarted, the Pod will receive a new local IP, but the service IP originally created by Kubernetes will remain the same in this way, allowing Kubernetes to route traffic to the appropriate set of Pods.

3.5 Cloud/Edge Layers Management

In this section, we describe a possible architecture that implements our reference scenario illustrated in Figure 3.1. In particular, after an overview of the Kubernetes platform, we describe how it can be adopted to develop distributed microservices in a CC/CE environment.

3.5.1 Kubernetes Overview

Nowadays, users require 24-hour applications. The constant development of new versions of such applications and the need to upgrade them can lead to their inactivity. Application development follows these pipelines, as applications are updated and released quickly by eliminating downtime. Another essential requirement is to intelligently scale-up applications while growing demand by ensuring particular network capability to guarantee particular Service Level Agreements (SLAs). Kubernetes is a platform designed as a highly available cluster whose purpose is to meet those requirements. Its abstraction allows us to deploy and scale applications that benefit from this service and have to be packaged or containerized to decouple them from the underlying host utilizing the container virtualization technology. This approach is different from the previous one, where deployed applications were installed on specific physical or virtual machines as deeply integrated packages in the host. The purpose of Kubernetes is to automate the deployment of application containers on the various cluster nodes. As defined on the official website, *"Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications"* [122]. From a networking point of view, the components are interconnected, as shown in Figure 3.2. The main Kubernetes components are *master* and *workeror minion nodes*. The master node manages the whole platform making global decisions, for example, performing task scheduling and detecting and responding to system events. It interacts with the etcd cluster to store data regarding the various operations, and it is also responsible for starting up and access Pods on minion nodes. A Pod is a group of one or more containers that include shared volumes, IP addresses, and information about how to run them [123]. In other words, a Pod encapsulates a group of containers that are scheduled on the same host. It consists of a single container or a small number

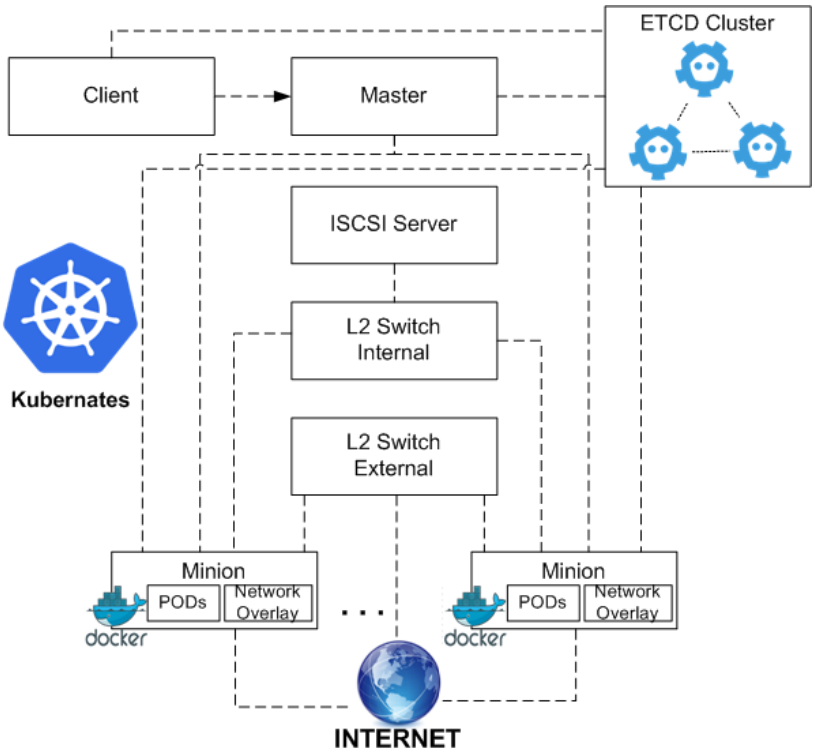


Figure 3.2: Basic Kubernetes architecture.

of tightly-enclosed containers that share the same resources. It represents the smallest planning, distribution, and horizontal/vertical scaling unit created and managed in Kubernetes.

The minion nodes interact with the master node via HTTPS or HTTP. They provide the Kubernetes runtime environment using Docker as a container engine to create Pods. Docker is an open-source platform for developers and system administrators that automates application deployment by leveraging container virtualization, such as *cgroups* and *namespaces*. Container isolation allows us to perform simultaneously several containers on the same machine without interrupting each other. It is also possible to establish safe connections between containers to make them interactive. Containers placed in the same Pod can reach each other via local addresses and communicate using interprocess communications such as SystemV semaphores or POSIX shared memory. The same is not valid for containers located in separate Pods that have to use an overlay network. Minion nodes interact with an internal L2 switch connected with an ISCSI Server, allowing for mounting a volume into a Pod. Moreover, an external L2 switch connects minion nodes over the Internet.

From a networking perspective, in Kubernetes, it is possible to identify three communication modes between the various components:

1. *Pod-to-Pod Communication.* In a Kubernetes cluster, each Pod is assigned

an IP address in a flat shared networking namespace that all other Pods and hosts can access within the cluster. This forms a networking model where every pod can communicate with the network just like in a VM;

2. *Pod-to-Service Communication.* Each service's IP addresses belong to a pool of addresses configured in the Kubernetes API Server using the *service-ip-range* flag. Microservices are deployed using virtual IP addresses accessible by various clients and are intercepted by a kube-proxy process running on all hosts to route the traffic to the correct Pod. A service is an abstraction that defines a logical set of Pods and a policy to access them (e.g., load balancing). The Pod group covered by the service is specified using a label selector;

3. *External-to-Internal Communication.* Previous communication modes can access a Pod or a service within the cluster itself. To access a Pod from outside the cluster, it is necessary to adopt an external load balancer to reach all cluster nodes. Each request to reach the correct node is recognized as part of a particular service and routed by the load balancer to the appropriate Pod through kube-proxy.

The coordination of dynamically allocated ports is challenging because each application must be associated with a port through a flag so that the API servers have to know how to insert dynamic port numbers into configuration blocks. Moreover, microservices have to know how to find each other. To overcome these issues, Kubernetes uses different approaches to each one addressing particular requirements:

1. all containers can communicate with all other containers without NAT;

2. all nodes can communicate with all containers (and vice-versa) without NAT;

3. the IP address of all containers belonging to the same Pod is the same.

These assumptions allow affirming that all containers within a Pod share the same namespace and are reachable to each other through the localhost address. This model is not only less complex overall, but it is principally compatible with the desire for Kubernetes to enable low-friction porting of applications from VMs to containers [124]. This implies that containers within a Pod must coordinate port usage, but this is different from having processes in a VM. We call this the IP-per-Pod model. This is implemented in Docker as a Pod container that holds the network namespace open while containerized microservices join that namespace with Docker's *–net=container:<id>* function. In this case, a port will be allocated on the host node, and traffic will be forwarded to the Pod. The Pod itself is blind to the existence or non-existence of host ports. To implement

this networking model, Kubernetes does not natively provide any multi-host networking solution but relies on third-party plug-ins: OVN, Weave Net, and Flannel.

3.5.2 Distributed Microservice Management

Figure 3.3 shows an example of DμS running in a cloud/edge environment based on Kubernetes. CR 2 runs in VM 1, a Kubernetes master node directly connected with a master gateway (GW) and two minion nodes running Docker containers containing microservices. The master GW connects through an overlay network CR2 with CR 1 and CR 2 through their respective minion GWs. CR 1 and 3 are similar to CR 2, with the only difference that they run only minion nodes. The CR 2 master GW is also interconnected through the overlay network with ER 4 and ER 5, including different IoT devices. In the Edge Layer, the Docker container virtualization is partially adopted according to the hardware/software capability of involved IoT devices. For example, considering ER 4, ED 1 (e.g., a smartphone) and ED 2 (e.g., an SBC not supporting Docker) directly run μS(s), whereas ED N (e.g., an SBC supporting Docker) runs containerized μS(s). Moreover, a DμS is arranged interconnecting four μS(s) respectively deployed in regions CR 1, CR 3, ER 4, and ER 5.

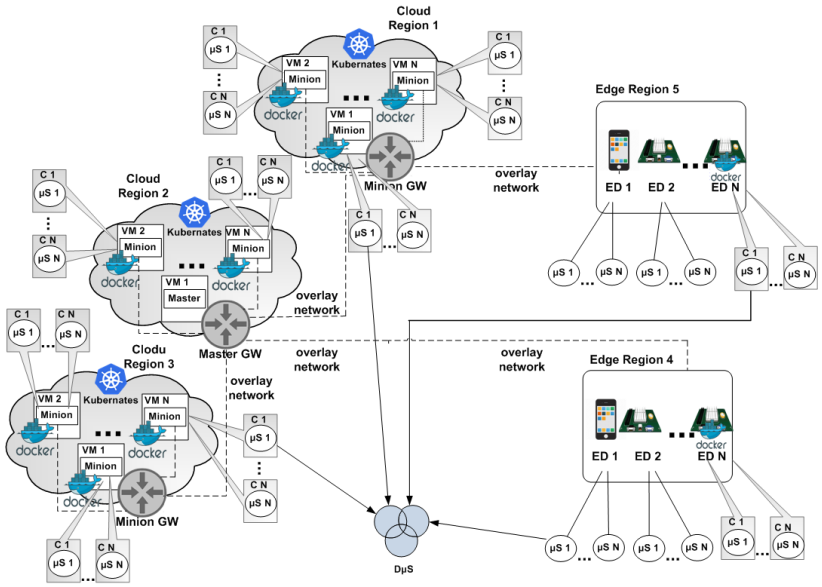


Figure 3.3: Example of distributed microservice in a Cloud/Edge environment based on Kubernetes.

3.6 Workflow

The first step is to create the Docker containers needed to deploy the microservices and configure the Kubernetes cluster. To perform these tasks, for each level of a specific microservices, a Dockerfile has been created for each tier of a specific microservice. This contains the commands needed to create related images automatically. Listing 3.1 shows a Dockerfile snippet for the CoAP server deployment.

Listing 3.1: Dockerfile snippet - CoAP server.

```
FROM centos
MAINTAINER Alina Buzachis
ADD CoAPthon-master /dir
ADD files /dir
WORKDIR dir
RUN python setup.py install
CMD ["python", "coapserver.py", "-i", "127.0.0.1", "-p", "5683"]
```

After executing the Dockerfile, we have a real container ready for the start and stop operations. Therefore, the container must be loaded on the Docker Registry using a push operation to allow recovery, simple and scalable development. Thus, the different images are available in the Registry and ready to be orchestrated by Kubernetes.

Listing 3.2: CoAP Pod YAML file.

```
---
apiVersion: v1
kind: Pod
metadata:
    name: coapserver
    labels:
        apps: coapserver
spec:
    containers:
        - name: coapserver
          image: alinab/coapserver:alfa 0.1
          imagePullPolicy: IfNotPresent
    ports:
        - containerPort: 5683
    nodeSelector:
        name: minion1-amd64
---
```

To create Pod and Service objects, it is necessary to execute the following commands: (1) *kubectl create -f server_pod.yaml* and (2) *kubectl create -f server_service.yaml*. Furthermore, analyzing the Listing 3.2, we notice that the

CoAP server Pod is scheduled on the *minion1-amd64 node*. To access the CoAP service externally, thanks to the *NodePort* mapping defined in the Listing 5.3, the 5683 port is mapped externally.

Listing 3.3: CoAP service YAML file.

```
---
apiVersion: v1
kind: Service
metadata:
    name: coapserver
    labels:
        apps: coapserver
spec:
    type: NodePort
    ports:
        - protocol: UDP
          port: 5683
    selector:
        apps: coapserver
---
```

3.6.1 Cluster Configuration

Once the containers could be accessed and managed by our platform, we needed to create the Kubernetes cluster, a cluster with a master and two minion nodes, and distribute the various Pods that will encapsulate all the containers. Also, the creation of the cluster, thanks to the simplicity of using Kubernetes, can be done using Vagrant through the *vagrant up* command.

At the end of this procedure, we will have a Kubernetes cluster consisting of a master node and two minion nodes that communicate with each other through a proposed network overlay. On the latter, using the *kubectl* command-line interface, it is possible to deploy and test the relative microservices. In this context, we have developed four proposed scenarios in which a different network overlay is applied to each of them: Flannel, OVN, Weave Net, and Calico. Very interesting is OVN; it can create L3 gateways. To have this type of configuration, a gateway must be configured on each node of the cluster. In this way, it is possible to pin the Pod subnet traffic to go out of a particular gateway. Additionally, the master's GW is configured to control the Pods belonging to a specific subnet. In this way, the flow is directed to go out through the master's GW. To do this, we can provide *–rampout-ip-subnets = "10.10.2.0/24,10.10.3.0/24"* option to the *gateway-init* command, as shown is Listing 3.4 during the gateway initialization phase.

By analyzing the gateway initialization script in Listing 3.4, we note that we share a single network interface for both management traffic (e.g., ssh) and

cluster's North-South traffic. Specifically, we attach the physical interface to an OvS bridge, move its IP address, and route to that bridge. If we choose *enp0s9* as the primary interface, with the IP address PUBLIC_IP, we create a bridge called *br-enp0s9*, add *enp0s9* as a port and move the PUBLIC_IP address to *br-enp0s9*. Then, we add the subnets that this gateway should manage and a default gateway GW_IP.

Listing 3.4: Gateway initialization command.

```
sudo ovn-k8s-overlay gateway-init --cluster-ip-subnet="10.10.0.0/16" \
    --bridge-interface br-enp0s9 \
    --physical-ip "$PUBLIC_IP"/"$MASK" \
    --node-name="kube-gateway-master" \
    --rampout-ip-subnets="10.10.2.0,10.10.3.0" \
    --default-gw "$GW_IP"
```

This is an essential feature because it represents the starting point for realising a network flow segmentation, the object of future work.

3.7 Experiments

In this section, we present the experimental results of the deployed system. In particular, we analyze the network performances in terms of transfer times collected using different overlay networks and running two different microservices (μS): FTP and CoAP based, respectively. Moreover, the experiments have been conducted accordingly to three scenarios: (i) Cloud-to-Cloud, (ii) Cloud-to-Edge and (iii) Edge-to-Edge. Mainly, these experiments aim to calculate the overhead introduced by the overlay networks in each proposed scenario and compare them.

Unfortunately, we have not found any similar work in the scientific community to compare our system and other solutions. In all the experiments conducted, we compare the performance of three overlay networks: (i) OVN, (ii) Weave Net and (iii) Flannel with the case where we did not apply any overlay networks. To evaluate the system's performances, for each proposed scenario, we collected 30 subsequent experiments and calculated the average times and the 95% confidence interval for both services implemented in four different configurations with increasing payloads.

3.7.1 Testbed Configuration

As previously described, we propose three different scenarios: i) Cloud-to-Cloud, ii) Cloud-to-Edge and iii) Edge-to-Edge. As a public cloud, we used GARR's platform (an Italian no-profit organization founded by several research organizations).

Scenario #1: Cloud-to-Cloud In this scenario, our testbed is made up of 3 VMs: one acts as a Kubernetes master node, and the others as Kubernetes minion nodes on which are scheduled the FTP/CoAP DμS and the service's client, respectively. Each VM presents the following HW/SW configurations: 4 VCPUs @ 3.2 GHz, RAM 8 GB, OS Ubuntu Xenial 16.04 with Linux Kernel 4.4.0-98-generic.

Scenario #2: Cloud-to-Edge Our testbed is consists of 2 VMs running in the cloud in this hybrid scenario, acting respectively as Kubernetes master and worker nodes. An edge device also acts as a Kubernetes worker node. On the edge worker node is scheduled the FTP/CoAP μS while the cloud worker runs the service's client. We remark that in this scenario, the VMs' HW/SW configurations are the same as those adopted in the Cloud-to-Cloud scenario, while as edge devices, we have used the Raspberry Pi 3 Model B+ with Raspbian Stretch Lite and Kernel 4.9.

Scenario #3: Edge-to-Edge In this scenario, our testbed consists of three Raspberry Pis: a Kubernetes master and 2 worker nodes. On the worker nodes, the FTP/CoAP DμS and, respectively, the service's client are scheduled.

We remark that in all presented scenarios for the Kubernetes cluster configuration, we adopted the Kubernetes 1.17 version while for container deployment, we used Docker version 17.05.0-ce, build 89658be. In all experiments, we configured the following overlays network: i) OVN based on the Open vSwitch 2.7.0-1 version - overlay mode, geneve tunnel type, ii) Weave Net v2.1.5 - pcap mode (sleeve), and iii) Flannel - v0.7.1-amd64 - vxlan mode.

3.7.2 FTP Microservice

Our first analysis allows us to verify the scalability of our deployed system for each configuration scenario (i) Cloud-to-Cloud, (ii) Cloud-to-Edge, and (iii) Edge-to-Edge. Specifically, we increased the payload starting from 1 MB up to 1 GB. Hence, in the next, the figures show the time performances registered using the FTP μS in the three proposed scenarios.

Cloud-to-Cloud Scenario

Figure 3.4 illustrates the collected results obtained with the FTP experiment.

It can be observed that the performance of the Flannel overlay network is worse than the other configurations. In particular, the performance drop is because Flannel introduces a higher overhead compared to other overlay network solutions. The Flannel-based solution has slower transfer times for each payload.

This overhead is a critical issue as it increases latency, compromising the real-time sessions' quality. A further improvement can be seen using Weave Net, which, compared to Flannel, introduces less overhead. The best performance is

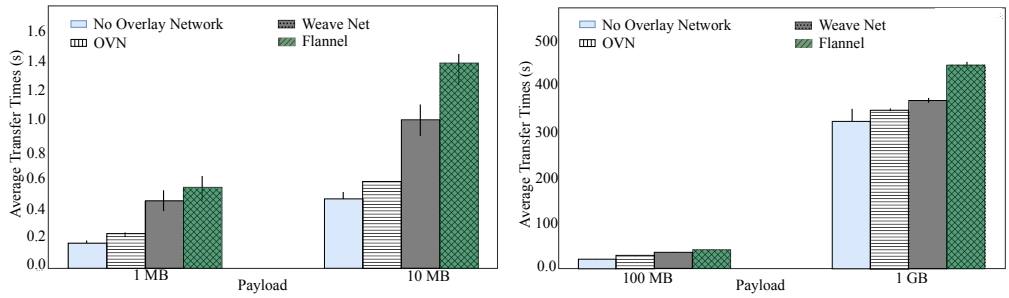


Figure 3.4: Cloud-to-Cloud Scenario: FTP μS Average Transfer Times using OVN/Weave Net/Flannel overlays.

obtained by applying the OVN overlay network. It introduces less overhead in all proposed configurations, as illustrated in Figure 3.7.

As a second consideration, as shown in Figure 3.4, transfer times worsen in Cloud-to-Edge scenarios and Edge-to-Edge scenarios, respectively. This increase in average transfer times and implicit decline in performance is mainly due to the lack of computational resources on edge devices. We get the worst performances in the Edge-to-Edge scenario.

Cloud-to-Edge Scenario

As in the case of the Cloud-to-Cloud scenario, even in the Cloud-to-Edge, we conducted the same experiments. Therefore, the obtained results can be seen in Figure 3.5. Also, for this scenario, the considerations previously made are valid; indeed, the performances registered to apply the Flannel overlay are significantly dropped respect to those obtained in the other configurations. Compared to the transfer times obtained in the previous scenario, here, time performances are slightly higher due to the lack of computational resources of the edge device.

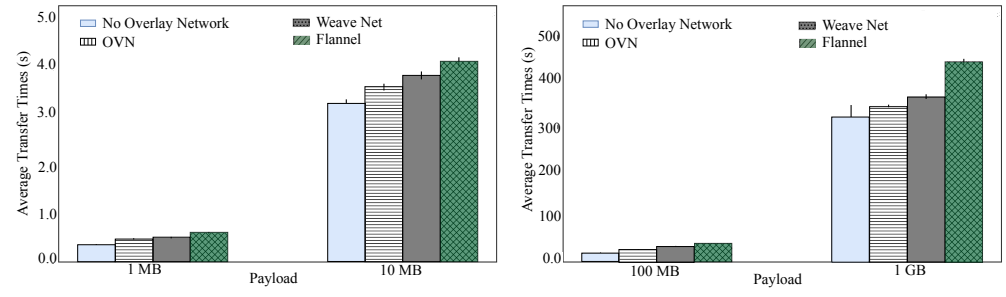


Figure 3.5: Cloud-to-Edge Scenario: FTP μS average transfer times using OVN/Weave Net/Flannel overlays.

Edge-to-Edge Scenario

Figure 3.6 shows the obtained average transfer time performances for the Edge-to-Edge scenario; the behaviour is the same as the one already presented. As we expected, due to the lower performances of the edge devices, transfer times are higher than others presented in the two previous scenarios.

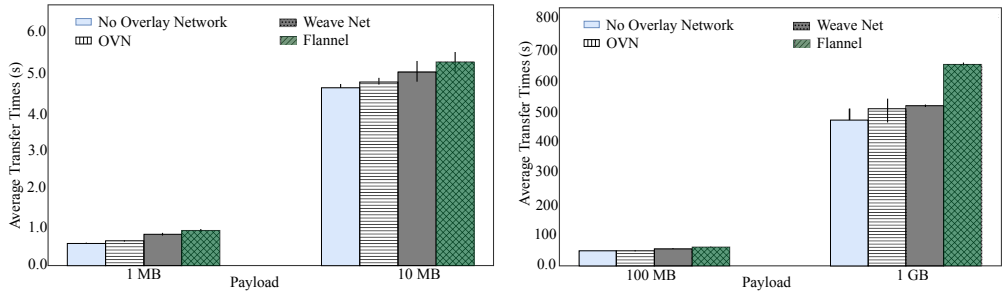


Figure 3.6: Edge-to-Edge Scenario: FTP μS average transfer times using OVN/Weave/Flannel overlays.

Following all the steps performed in the case of the FTP μS, also for the CoAP μS, we performed the same scalability tests in the three proposed scenarios. In these experiments, due to the nature of the CoAP, which is not suitable for large file transfers but only to exchange simple messages among hosts, we considered a payload file of 10, 100, 1000, and 10000 bytes.

Cloud-to-Cloud Scenario

Figure 3.8 (a) shows the behaviour of the average transfer times registered with the deployed service. Similarly to the performance obtained with the FTP μS, we note that Flannel introduces a higher overhead than other overlay network solutions. OVN presents the best time performances in all configurations. Indeed, it introduces less overhead in all proposed configurations, as illustrated in Figure 3.9.

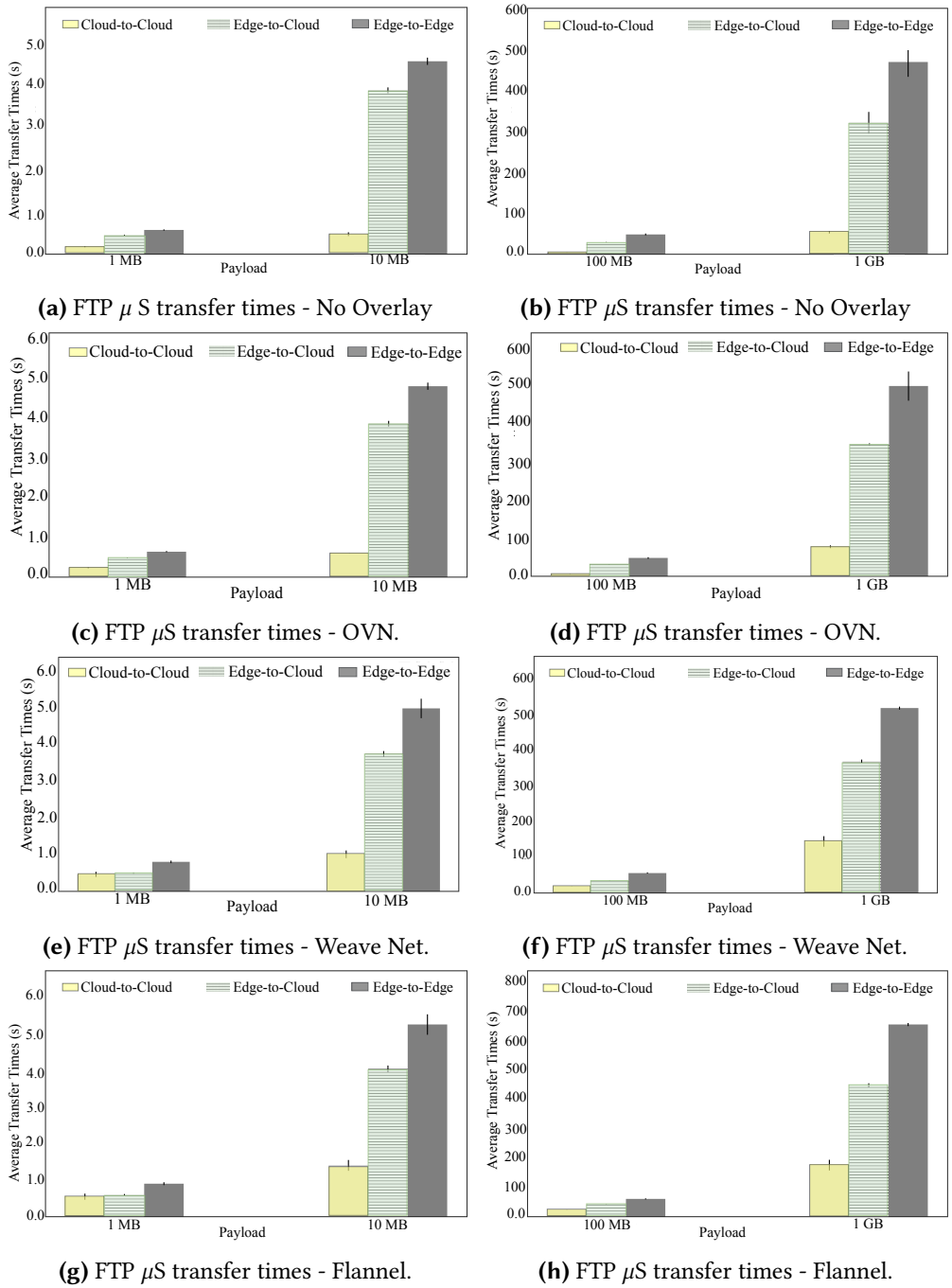
It is important to notice that the transfer times for experiments with payload equal to 10, 100, and 1000 bytes are very similar; this behaviour is due to the maximum size of the payload supported by CoAP that is equal to 1024 bytes.

For this scenario, further consideration highlights that the transfer times worsen in the Cloud-to-Edge and, respectively, in the Edge-to-Edge scenarios. This performance drop's motivation explained in the Cloud-to-Cloud scenario of the FTP μS is valid.

Cloud-to-Edge Scenario

Figure 3.8 (b) shows the obtained transfer time performances. Also, here, the best performances are obtained using the OVN overlay network, while the worst

are obtained with the Flannel overlay. As in the FTP Cloud-to-Edge scenario, the transfer times are slightly higher than those obtained in the previous scenario.



(a) FTP μ S transfer times - No Overlay

(b) FTP μS transfer times - No Overlay

(c) FTP μS transfer times - OVN.

(d) FTP μS transfer times - OVN.

(e) FTP μS transfer times - Weave Net.

(f) FTP μS transfer times - Weave Net.

(g) FTP μS transfer times - Flannel.

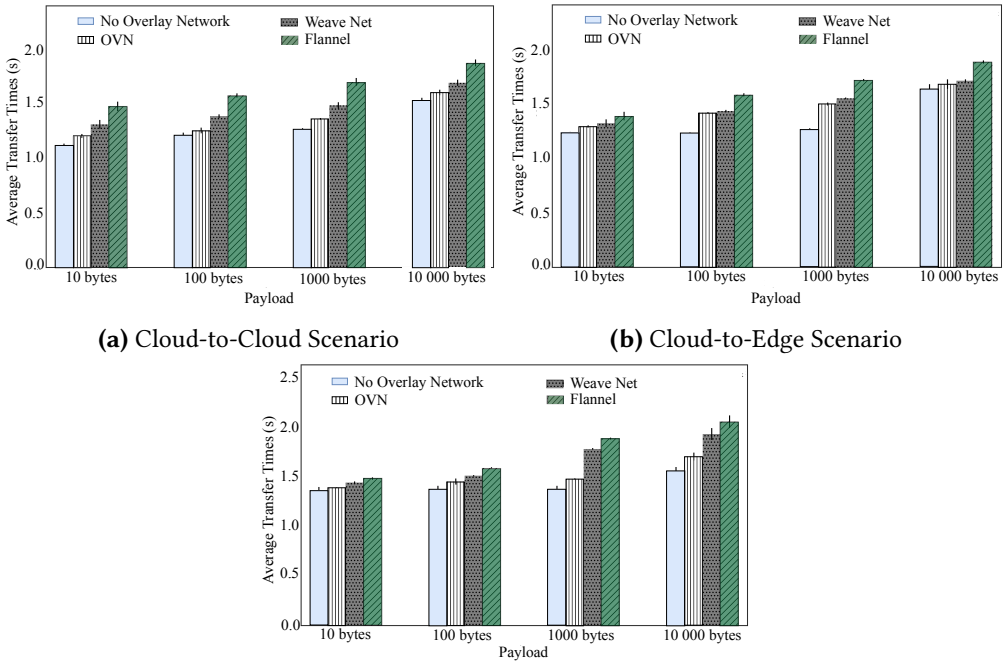
(h) FTP μS transfer times - Flannel.

Figure 3.7: FTP μS transfer times obtained using the proposed overlay networks.

3.7.3 CoAP Microservice

Edge-to-Edge Scenario

Figure 3.8 (c) shows the transfer times obtained in the Edge-to-Edge scenario. The behaviour is the same for the FTP Edge-to-Edge scenario.



(a) Cloud-to-Cloud Scenario

(b) Cloud-to-Edge Scenario

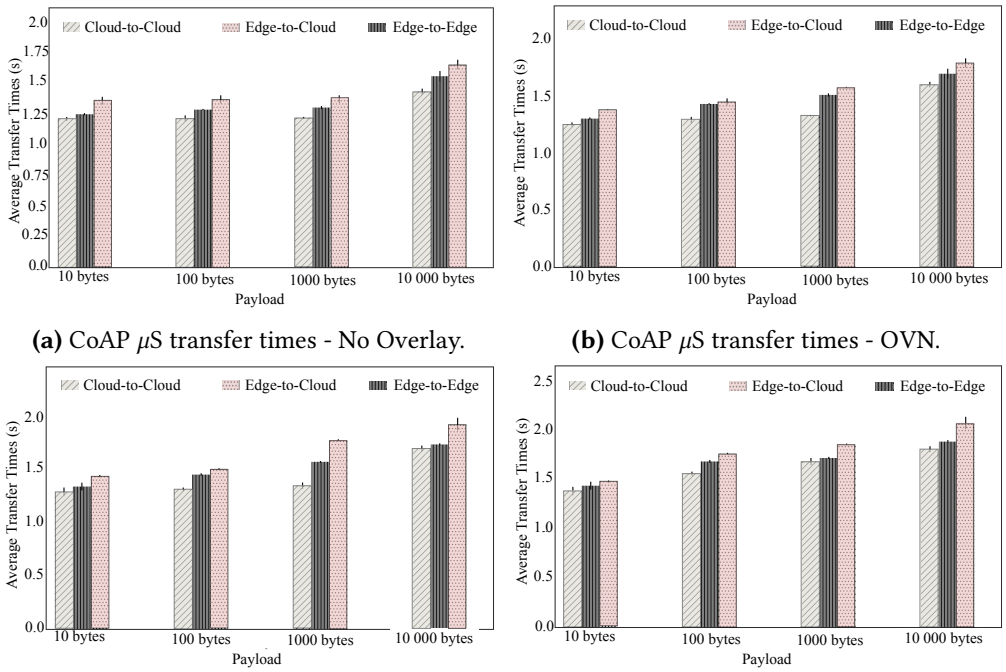
(c) Edge-to-Edge Scenario

Figure 3.8: CoAP μS average transfer times using OVN/Weave Net/Flannel overlays.

3.7.4 Discussion

The overhead introduced by the overlay networks is a critical issue, especially in real-time applications. Indeed, it could introduce a non-negligible latency and, in some cases, could also compromise the proper functioning of the sensor network. Moreover, we note a considerable overhead introduced by Flannel. This could cause problems in terms of *synchronization, transfer time, slow connections*, and so forth. These problems are the critical elements of the proper functioning of today's software. Taking the example of FTP μS, a not negligible latency means introducing a quality deterioration, for example, in a real-time session. In the case of the CoAP protocol, this aspect leads to a lack of synchronism and subsequently to the non-execution of quantifiable operations by sensors present within a network. This could have a disastrous impact in situations where the change (through code injection) of a specific variable is required.

In summary, Flannel allows the creation of a real bridged network on the



(a) CoAP μS transfer times - No Overlay.

(b) CoAP μS transfer times - OVN.

(c) CoAP μS transfer times - Weave Net.

(d) CoAP μS transfer times - Flannel.

Figure 3.9: CoAP μS transfer times obtained using the proposed overlay networks.

host with an associated subnet address and uses a host Linux routing table to forward container packages to this bridge. Flannel does not allow the overlapping between the bridged subnet and the host's routing table. Unlike Flannel, Weave Net does not use the host's routing table to differentiate packages from containers; it uses the pcap feature. Moreover, Weave Net allows allocating to container the same IP address as the host. Compared to the previous overlay networks, OVN provides a layer 2/3 virtual networking along with a set of services such as firewall, flexible security policies (ACLs), distributed L3 routing, IPv4, and IPv6, native support for NAT, load balancing, DHCP, L2, and L3 gateways.

In conclusion, from an implementation point of view, Flannel and Weave Net are easier and more intuitive than OVN, but this does not mean that the choice easier to implement is also the best. To choose the best overlay network, we need to evaluate the type of scenario and the complexity and, respectively, the design requirements, i.e., in a system where the implementation of gateways or ACLs is not required, we should exclude OVN and use a simpler one, as well as in a scenario where no specific bandwidth usage or fast connections are required, we can also use a simple overlay network; this also applies vice-versa. Despite its complexity, taking advantage of the OVN's features such as L3 gateways and logical flows, we can solve the network's crucial problem: segmentation. We can select the path that a particular type of packet must follow and differentiate the streams: mice and elephant flows. In this way, we obtain a coloured network that

allows us to optimize bandwidth consumption and limit latency. Furthermore, there is the possibility to change the path to be followed by specific packages dynamically; in this case, congestion is limited.

3.8 Conclusions and Future Work

Nowadays, the need for efficient pervasive services and applications has pushed ICT operators to move part of their services from the central cloud into an intermediate layer, closer to users, defined edge. In this context, the concept of microservice architecture is gaining more and more consensus among the industrial and academic communities because it allows the development of emerging cutting-edge and efficient Cloud/Edge Computing systems. However, one of the major problems in achieving this system is the configuration of softwarized networks.

In this chapter, considering a reference cloud/edge system based on Kubernetes, we explored different softwarized network technologies: OVN, Weave Net, and Flannel considering Cloud-to-Cloud, Cloud-to-Edge, and Edge-to-Edge communication scenarios. Experimental results have shown that OVN is the best potential solution as it introduces the lowest overhead. Moreover, by exploiting the OVN's features like L3 gateways and logical flows, we can tackle the network segmentation problem.

We plan to study the dynamic orchestration of overlay networks integrating the emerging Osmotic Computing paradigm in future work. In this way, we obtain a coloured network that allows us to optimize bandwidth consumption and limit latency. Furthermore, there is the possibility to change the path to be followed by specific packages dynamically; in this case, congestion is limited.

4 Security Management

The IoT stages a new technology that empowers both virtual and physical objects to be connected and communicate with each other and produce new digitized services that improve our quality of life. The IoT system provides several advantages; however, the current centralized architecture introduces numerous issues involving a single point of failure, security, privacy, transparency, and data integrity. These challenges are an obstacle in the way of the future developments of IoT applications. Moving the IoT into one of the distributed ledger technologies may be the correct choice to resolve these issues. Amongst the common and popular types of distributed ledger technologies is the blockchain. Integrating the IoT with blockchain technology can bring countless benefits. Therefore, this chapter provides a comprehensive discussion of integrating the IoT system with blockchain technology. The Blockchain-as-a-Service (BaaS) for the IoT is presented to show how various blockchain technology features can be implemented as a service for various IoT applications. Moreover, two innovative use case scenarios highlight the feasibility and benefits of using blockchain technologies (e.g., Hyperledger Fabric, Ethereum) in the continuums. Further, the first use case scenario proposes a Hyperledger Fabric-based BaaS for intersection management and faces the problem of preventing vehicular collisions in intersections by proposing a Multi-Agent Autonomous Intersection Management (MA-AIM) system. The second use case proposes an Ethereum-based BaaS solution for Health Information Exchange (HIE) systems.

4.1 BaaS-based Multi-Agent System for Intersection Management

4.1.1 Introduction

Cooperative Intelligent Transportation System (C-ITS) offers a novel approach in providing different transportation modes, advanced infrastructure, traffic and mobility management solutions by revolutionizing the way people commute in smart cities. It uses several electronics, wireless, and communication technologies to allow consumers to access a smarter, safer, and faster way to travel. Market reports estimate an annual growth rate of 25.1% in the automotive industry for the coming five years. From USD 72.05 Billion in 2016 and $82 billion in 2020 [125] it is expected to reach USD 220.76 Billion by 2021 [126]. The major proponents of this growth are smart cities, the need for public security and

safety and the government's initiatives to improve present-day transportation infrastructure. Concerning smart mobility, the new methodology that automated industry advocates introduce the integrated design guidance and control systems for Autonomous Vehicles (AVs).

Internet of Things (IoT) technologies in the automated industry enabled AVs to become comprehensive Cyber-Physical Systems (CPS) with communication, control, and sensing capabilities. On the other hand, AVs will be able to interact through the Edge of the network [127] and/or the cloud [128] with intelligent, secure traffic monitoring and orchestration devices installed in the smart cities able to make: i) roads safer for both drivers and pedestrian, and ii) reduce traffic.

Cloud-to-Things (C2T), among others, can revolutionize the Intelligent Transportation System (ITS) sector to impact vehicle safety, congestion, and travel behaviour. Therefore, in the last decade, vehicles have been manufactured with on-board diagnostic ports to retrieve vehicle controller diagnostics, Electronic Control Units (ECU) and On-Board Units (OBUs) to receive data from several on-board sensing devices. In this context, AVs represent a potentially disruptive yet beneficial change to our transportation system, having the potential to impact vehicle safety, congestion, and travel behaviour. In recent years, many car manufacturers such as Daimler, Tesla, and Nissan have successfully presented cars with automatic driving in urban areas [129]. In this scenario, intersections are a very challenging open issue. In fact, in urban areas where there is a high density of vehicles, intersection management is not trivial. According to a recent National Highway Traffic Safety Administration report, 40% of crashes typically occur in intersections [130]. Furthermore, many intersection vehicle accidents occur in metro areas where high-density neighbourhoods, numerous businesses, and pedestrian traffic converge; any intersection can be dangerous. AVs can take a lot of different actions when approaching an intersection. Several research centres are developing algorithms to solve one of the crucial aspects of autonomous driving, i.e., intersections management, to avoid collisions and traffic congestion. In ITS, vehicular ad-hoc networks (VANETs) have significant roles in the communication process for ITS. Each vehicle is regarded as a node in a VANET, where a vehicle can transmit information and provide request services in a Vehicle-to-Intersection (V2I) and Vehicle-to-Vehicle (V2V) manner. There are implementations based on Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), Vehicle-to-Human (V2H), or in general Vehicle-to-Everything (V2X) communications, which are currently a focus of research and standardization in the USA, Europe, and Asia. Because many of these interactions transmit sensitive data such as identification, position, and speed of the vehicle, high security and privacy level are prerequisites for the broad acceptance of these communication systems. In particular, communication security and privacy are the main aspects to consider: the physical security of the microcontroller, key injection, privacy mechanisms implemented by government agencies, and policy questions around

security. This field is becoming even more important with the proliferation of self-driving cars, prone to failures and cyber-attacks. Since an intersection management system can transmit sensitive data such as identification, position, and vehicle speed, security is important.

This contribution investigates the current C-ITS solutions for intersection management and faces the problem of preventing vehicular collisions in intersections by proposing a Multi-Agent Autonomous Intersection Management (MA-AIM) system based on V2I/I2V communication able to safely manage AVs crossing through an intersection leveraging both C2T continuum paradigms and blockchain facilities. Specifically, in this contribution, we present (1) a fully functioning AIM system that combines V2I/I2V communication and blockchain is implemented; (2) to increase the intersection robustness in terms of consensus, several IMAs are run at each intersection; (3) careful performance analysis is carried out to demonstrate the applicability of the BaaS-based AIM system is real scenarios. In particular, we implemented a proof-of-concept integrating the AIM 4.0 simulator with Hyperledger Fabric (HF) (implementing blockchain capabilities) and with Node-RED (implementing V2I/I2V communications in an EoT environment). To validate our system's performance, we conducted a series of experiments with different road traffic conditions, different number of intersections, and a different number of IMAs. The collected results helped us quantify how the HF framework's overhead impacts the overall system's performance and its suitability for real-time use.

The rest of this contribution is organized as follows. Related works are discussed in Section 4.1.2. Motivation is discussed in Section 4.1.3. In Section 4.1.4, we provide an overview of the adopted enabling technologies, whereas the design of the proposed MA-AIM system is described in Section 4.1.5. Implementation highlights are described in 4.1.6. Experiments are discussed in the 4.1.7. Section 4.1.8 concludes the paper and also providing lights to the future.

4.1.2 Related Works

In ITS, the rise of AVs has attracted attention in both industrial and academic fields. In this context, the ITS industry is focusing much of its attention on the concepts of *connected vehicles* (United States) or *cooperative ITS* (Europe). These concepts rely on data communication among vehicles (V2V) and/or between vehicles and the infrastructure (V2I/I2V) to provide the information needed to implement ITS applications.

With the rapid development of (IoT and CC, connected vehicles are set to become a massive industry over the next few years. [131] discusses a model for the next generation of ITS, which focuses on dynamic decision making of connected vehicles based on a Swarm Intelligence (SI)-based algorithm. Specifically, the authors propose a communication framework among connected vehicles for sharing information of traffic flow firstly, and secondly, connected vehicles are

regarded as artificial ants that can self-calculate to make an adaptive decision following the dynamics of traffic flow. An IoT cloud system for traffic monitoring and alert notification based on OpenGTS and MongoDB for car accident prevention is discussed in [132]. In [133], the authors propose reliable V2V/V2I communications to adaptively select parameters for improving communication efficiency by exploiting the surrounding environment information using the spectrum environment map. In [134], considering a vehicular ad-hoc network, a clustering method for gathering data related to vehicle movements is proposed. The purpose of this scientific work is to extend the green wave. In particular, traffic lights, communicating with vehicles, can make decisions, and change line priorities. In [135] an Intelligent Intersection Control System (ICS) to control traffic flow in intersections is proposed. Specifically, a hybrid fuzzy-genetic controller responsible for evaluating the appropriate action for each vehicle is implemented.

The concept of the Internet of Agent has recently been introduced as a potential technology that pushes intelligence, data processing, analytics, and communication capabilities down to the point where the data originates [136]. The authors introduce an approach for a Decentralized Home Energy Management System by applying the Internet of Agent concept. Moreover, by applying the Internet of Agent framework, connected appliances are regarded as smart agents that can make individual decisions by agreeing over the exchange of operations on competitive resources. In [137], the authors propose an ICS based on a Multi-Agent System (MAS) structure to control AVs in the intersections for better road utilization. Therefore, a central Intersection Manager Agent (IMA) is implemented at each intersection while each vehicle is controlled by a Driver Agent (DA). The proposed approach is verified using a developed simulation environment by testing both PID and Fuzzy controllers. As well as, in [138] discusses a Multi-Agent AIM (MA-AIM) system based on V2I/I2V communication to securely manage vehicles crossing where a central Intersection Manager Agent (IMA) is implemented at each intersection while a DA controls each vehicle. In [139], an approach to managing the crossing of autonomous vehicles through an intersection, avoiding collisions, and decreasing the waiting time at the intersection is proposed. This approach is implemented using V2I communication, and the safe trajectory of autonomous vehicles for the Autonomous Intersection Management is determined using discrete mathematics. An SI-based model for AVs is discussed in [140]. In particular, by sharing their position and other data, vehicles can set-up the speed and find the best path to a destination.

Although AVs supports a better convenience for society, it also suffers from some concerns. Security is the primary concern in AVs technology due to its high exposure to data and information communication. Among security, trust, data accuracy, and communication data reliability in the communication channel are the other issues in IV communication. Therefore, several studies on security

vulnerabilities have been conducted. In [141], the potential cyberattacks specific to automated vehicles are investigated. Moreover, in [142], a blockchain-based crypto Trust point (cTp) mechanism for IV communication to provide IV data security and reliability is discussed. The cTp mechanism accounts for the legitimate and illegitimate vehicle's behaviour and rewarding, thereby building trust among the vehicles. Blockchain technology is even adopted to provide a secure and reliable way to communicate among AVs with trust [143]. Moreover, a preliminary study on blockchain-based ITS (B2ITS) to develop real-time ride-sharing services is discussed in [144].

In summary, there is a relevant number of recent research works focusing on ITS applications. This contribution aims to enhance the state of the art by proposing an autonomous intersection management system using the AIM 4 simulator through V2I and I2V communication leveraging C2T continuum and blockchain facilities.

4.1.3 Motivations

This section explains the current problem in the state of the art concerning intersection management and motivates this research.

One of the main advantages of autonomous vehicles is the ability to even out human imperfection being more efficient in avoiding accidents safer since they can interact and react much faster and are prone to fewer errors than humans. One of the most critical aspects of managing autonomous vehicles is their behaviour in proximity to intersections.

The future vehicles will be equipped with On-Board Units (OBUs) to enable connectivity among vehicles and Road Side Units (RSUs) to provide collision avoidance and congestion control. These safety features will be realized with the wireless Dedicated Short Range Communications (DSRC) that will not only enable broadcasting of Basic Safety Messages (BSM) (e.g., V2V) but also provide the means to communicate with the infrastructure such as the traffic lights, railroad crossing (e.g., V2I).

AIM systems reduce the waiting and passing time of vehicles. In this scenario, AVs are assumed to have V2I communication capabilities, which can interact with the IMA. AVs communicate with the IMA through request and response messages that embody vehicle id, speed, position, and approval/wait messages. We assume that each intersection is managed by one dedicated IMA, which elaborates the vehicles' requests.

Architecturally, each AIM has a single centralized IMA responsible for handling the traffic. This centralization could introduce some drawbacks such as bottleneck, single point of failure, and security risks. The bottleneck issue can occur when a high density of vehicles makes request messages to IMA, causing a high load, which translates into lower response times and collisions. There also could be illegal vehicle requests that could cause the IMA to serve a non-existent

vehicle request.

For example, a hacker could exploit the IMA centralization by sending an illegitimate request for passage. This request for a non-existent vehicle would consume the space-time slot assigned by the IMA. In this way, hacking the IMA, the hacker gets the traffic control, and a vehicle can lie on its position in the traffic queue and ends up getting priority. The hacker will also be able to deceive the vehicle into colliding with another one. For example, we consider a scenario with two cars that send requests to this IMA for passing through the same area at the same time. The compromised IMA will send an approval response to both cars, which could ultimately lead these cars to a collision. Furthermore, the hacker could exploit a Denial of Service (DOS) attack so that vehicles would end up waiting at the road until the IMA recovers, and this could cost human lives and the reputation of the vehicle company.

For instance, with the increasing privacy concerns of data, privacy and reliability issues exist in building an effective VANET network. Ideally, all messages must be forwarded anonymously in VANETs since they usually contain AVs' sensitive information, as said before. However, forwarding messages anonymously does not assure the reliability of the messages, given that the fact that the IMA's centralization can be exploited for malicious scopes, as explained before.

Therefore, the fundamental aspects of being considered in automatic intersection management are authentication, privacy, security, reliability, availability, and scalability. These characteristics can only be satisfied by using a decentralized approach and a blockchain implementation. The permissioned blockchain of HF guarantees authentication and security of communications with the system. HF supports Transport Layer Security (TLS) to secure communication between nodes. TLS communication can use both one-way (server only) and two-way (server and client) authentication. Furthermore, using Hyperledger Composer, only through appropriate id cards will it be possible to forward requests to peers. In this system, for hackers, the ability to acquire access is to detect a blockchain node, and an authentication process can mitigate this in place. Moreover, these nodes are added statically to the network, unlike public blockchain networks, where anyone can join and listen to transactions.

4.1.4 Enabling Technologies

This section provides an overview of three enabling technologies on which our MA-AIM is based, e.g., Traffic Simulator, blockchain, and Node-RED.

Traffic Simulator

A variety of traffic simulation software has been developed to model traffic networks. For example, SimTrafficis and Simulation of Urban Mobility (SUMO) allow visual simulation of a surface street traffic network. Recently a new tightly

integrated traffic simulation/wireless communication system was developed, called AIM4.

AIM was developed by the Learning Agents Research Group, which is part of the AI in the Department of Computer Sciences at the University of Texas at Austin, and it is a traffic simulator that aims to create a scalable, safe, and efficient multi-agent framework for managing autonomous vehicles at intersections. AIM is an advanced management system that includes a scheduler and a simulation manager to detect vehicles' collisions and traffic congestion. A vital part of AIM is an intersection manager's infrastructure component, which manages the road crossing safely. AIM requires V2I communications, where each Driver Agent (DA) present on each vehicle requests to the IMA to cross through an intersection. In AIM, the central coordinator IM leads to a centralized architecture with some drawbacks such as a single point of failure, security risk, and bottleneck. A cyberattack on the intersection manager could let the hacker send manipulated responses to the cars, raising doubts in trusting the AIM. A compromised intersection manager could also lead to an accident.

The security risk of a central coordinator could cause a lack of trust among the entities involved. The bottleneck issue might not be a critical one, but it questions the AIM's main necessity. The AIM was developed to reduce the travel time of the vehicles crossing the road junction when the number of vehicles on the road increases, the load on the intersection manager will be high and could lead to a slower response.

Blockchain Technology

Blockchain is a novel technology that was first used as a public ledger for the Bitcoin cryptocurrency. It consists of consecutive chained blocks, replicated and stored by the nodes of a peer-to-peer network, where blocks are created in a distributed fashion using a consensus algorithm. Such an algorithm, together with crypto mechanisms, provides two properties of blockchain: 1) decentralization and 2) democratic control of data. This ensures that data on the chain cannot be tampered with maliciously, operations on the chain are non-repudiable and their origin is fully tracked. Blockchains are then typically classified into two main categories:

1. Permissionless: a block can be added to the blockchain by any peer who can join and leave the network as reader and writer at any time [145]. There is no central entity that manages the membership or which could ban illegitimate readers or writers. Examples are Bitcoin, Zerocash [146] and Ethereum [147].

2. Permissioned: only an authorized closed group of entities can write and read the respective blockchain. Here, a central entity decides and attributes

the right to individual peers to participate in the blockchain's write or read operations. Hence, consensus decisions are either taken unilaterally by this central entity or by a preselected group (so-called "consortium Blockchains"). Permissioned Blockchains can be further categorized into public and private. The most widely known instances of permissioned blockchains are HF [148] and R3 Corda [149].

The blockchain provides a tamper-proof data storage solution that makes it possible to store reports. A detailed analysis of available distributed ledger technologies was completed to identify HF as the best candidate for developing a blockchain-based architecture for vehicular applications. HF is the blockchain implementation proposed by IBM. It allows the creation of permissionless and permissioned solutions: users can communicate and interact with the company's blockchain by invitation only. HF presents a modular architecture and allows a division of roles between the infrastructure's nodes, the execution of smart contracts or chain codes, and the possibility to configure the consensus and the membership services. Hyperledger Composer is a set of comprehensive development tools to simplify blockchain applications' development by facilitating the integration of blockchain applications with existing business systems. An HF network includes:

- Peer nodes execute chaincodes, access log data, approve transactions and interface with applications;
- Ordering nodes ensure the consistency of the blockchain and deliver blockchain transactions to the peers of the network;
- Membership Service Provider Services (MSPs), generally implemented as Certification Authorities, manage certificates used to authenticate members' identities and roles.

HF allows creating channels and a group of participants with a separate log for their transactions. The HF register (ledger) combines a world state database and a transaction log history. The world state component describes the log's status at a given time and represents the distributed ledger's database. The transaction log component records all the transactions that brought the world state to the current state. By default, it is a LevelDB database of key-value type. An alternative to LevelDB is CouchDB, a NoSQL solution. HF allows choosing a consensus mechanism that best fits business needs. Currently, the consensus mechanisms made available by HF are SOLO, Kafka/Zookeeper, Byzantine Fault-Tolerant (BFT).

- SOLO is the sorting mechanism most commonly used by developers who experiment with HF networks. SOLO involves a single sorting node.

- Kafka is the recommended permissioned voting-based mechanism to use in production. It uses Apache Kafka, an open-source flow processing platform that provides unified, high-throughput and low-latency data feeds in real-time. The Kafka mechanism provides a Crash Fault-Tolerant solution (CFT) to the reordering.

- BFT prevents the system from reaching an agreement in the case of malicious or faulty nodes. Practical Byzantine Fault Tolerant (PBFT) is available for Fabric v0.6 and BFT-SMaRT available for Fabric v1.0 [150].

Node-RED

Node-RED is one of the best known flow-based programming tools for the Internet of Things. Its event-driven JavaScript model and the asynchronous execution of I/O operations allow developing scalable solutions for the real-time analysis of data flows, with a strong orientation to the IoT. Node-RED gives the possibility to connect different devices (with any sensors and actuators) and APIs and online services to create highly integrated and complex systems straightforwardly and intuitively. It is entirely written in JavaScript and runs on Node.js, the well-known platform based on Google's JavaScript Engine V8 for server-side applications.

4.1.5 Design

This section provides information on the design of our proposed BaaS-based AIM system.

Intersection Model

We have considered a symmetrical four-way intersection with a single lane for each incoming and outgoing street in this work. The model of the intersection we worked on is illustrated in Figure 4.1. In detail, this intersection is composed of four connected streets, where each street shares the common characteristic of driving direction. Hence, there are three possible choices of driving routes for each street that are composed of left, straight, and right directions (see Figure 4.1, V_b red car).

Simulation parameters For the simulation purpose, the vehicle's dynamics are configured according to the geometry of the intersection. For our simulation, we consider all vehicles having the same technical characteristics. Each vehicle has length $L_v = 4.5$ m and width $W_v = 1.85$ m, the maximum allowance of driving velocity is $v_{max} = 60$ m/s and the minimum is $v_{min} = -17$ m/s. The maximum acceleration is $a_{max} = 4.5\, m/s^2$ and the minimum is $a_{min} = -45\, m/s^2$.

To limit the number of vehicles communicating with the IM, a vehicle will start sending a message when it will reach the designated range of communication, which is R = 100 m away from the centre of the intersection as illustrated in Figure 4.1. We assume the distance between the vehicle and the intersection checkpoint L_{carr} = 25 m and the width of the intersection equal to L_{cross} = 4 m.

Reservation system The reservation system consists of two agents: Intersection Manager Agent (IMA) and Driver Agent (DA). For each intersection, there is at least a corresponding IMA. As well as, we assume each vehicle is equipped with a DA. IMAs are responsible for wirelessly communicating with each incoming vehicle's DA to manage its crossing through the intersection, based on the message requested arriving from the DAs. To prevent the message from crashing, the First-Come-First-Served (FCFS) principle is implemented for ordering the message queue. The IMA will do a service based on the sequence of the received message from the vehicle. Each DA is responsible for ensuring the corresponding vehicle will strictly and accurately follow the IMA's policy.

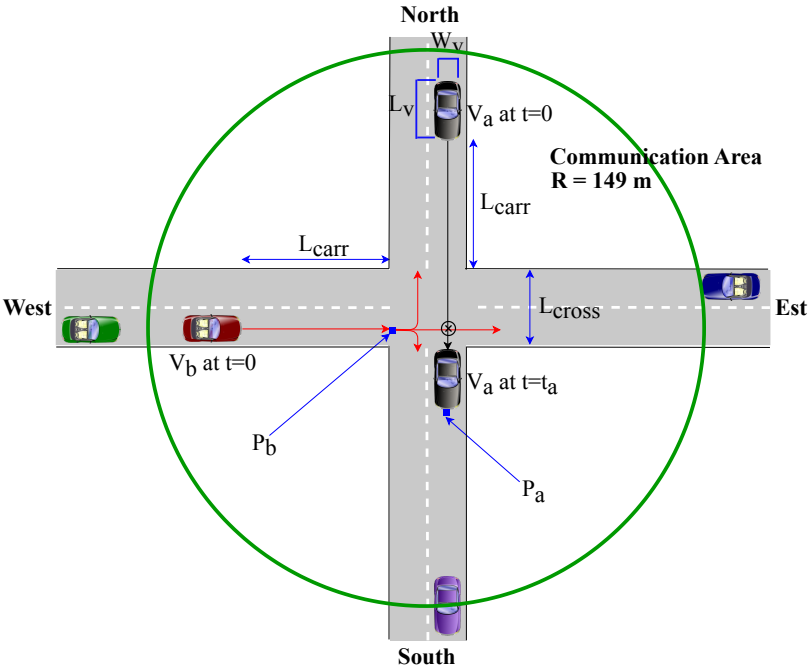


Figure 4.1: A real representation of a symmetrical four-way intersection with a single lane for each incoming and outgoing street.

To improve the system's throughput and efficiency, DAs request in advance the corresponding space-time occupied in the intersection to the IMA. Then, IMA checks whether the request can be satisfied according to the intersection control policy. If DA receives a positive response (e.g., reservation confirmation), it will ensure the vehicle will drive following the restrictions (if specified) com-

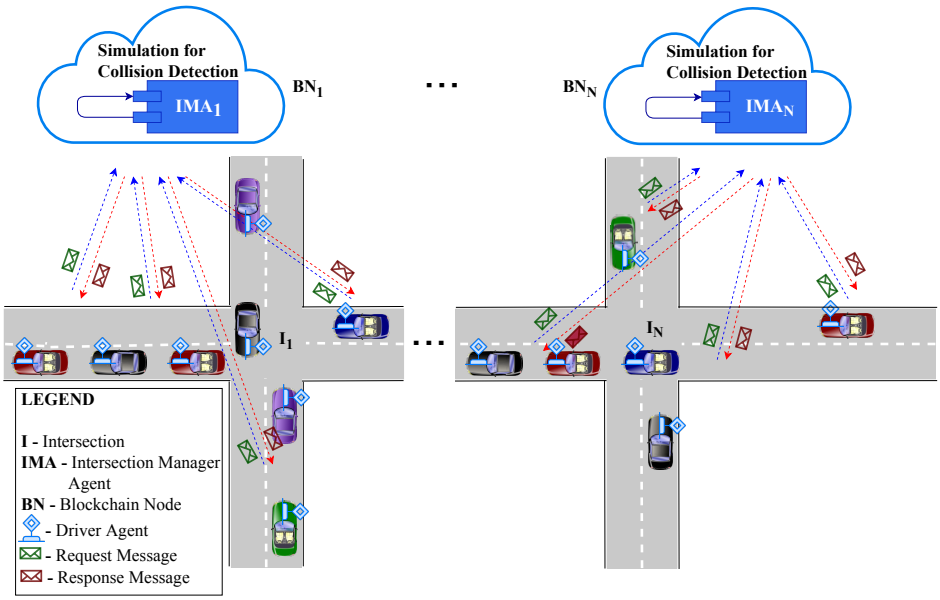


Figure 4.2: Blockchain-based Autonomous Intersection Management system.

municated by the IMA; when the crossing will be finished, the vehicle will send a message to the IMA to release the reservation. Otherwise, if DA will receive a rejection message, it will retry by sending another request later and ensure the vehicle will drive according to the IMA restrictions. The V2I communication is the instrument in which the message requested by the vehicle is delivered to the intersection manager and vice versa.

Intersection Control Policy For the AVs defined above to go through an intersection safely, we defined a simple interface for negotiating with the IMA and passing control between the IMA and the DA. The respective interface relies on the intersection management algorithm, which implements the Intersection Control Policy (ICP). The algorithm running on each IMA follows the FCFS policy.

The IMA continuously listens to vehicle requests and, with the information contained in the crossing requests, simulates vehicles' trajectories and responds according to the FCFS policy. In this way, FCFS will enable each AV to reserve in advance the space-time it needs to cross the intersection.

The request message contains a tuple with 7 parameters $r_i = (v_{in}, I_m, r_{ID}, t_a, v, l_{arr}, l_{exit})$, where v_{in} is the vehicle's identification number, I_m is the identification number of the IMA, r_{ID} is the identification number of the request, t_a is the estimated arrival time, v is the expected arrival speed, l_{exit} is the arrival lane from which the vehicle arrives at the intersection, and l_{arr} is the exit lane from which the vehicle leaves the intersection. The prototype intersection control policy divides the intersection into a grid of reservation tiles [151].

At each simulated time step, the intersection control policy determines which reservation tiles will be occupied by the vehicle. When an incoming vehicle reaches the designated line, it sends a reservation request to the IMA, indicating arrival time and velocity. The IMA simulates the vehicle's trajectory in the intersection and responds to the vehicle's corresponding DA with approval if the trajectory has no overlap with the reserved tiles regarding existing vehicles. If the request gets rejected, the vehicle prepares to follow the restrictions specified above or stop before the intersection line.

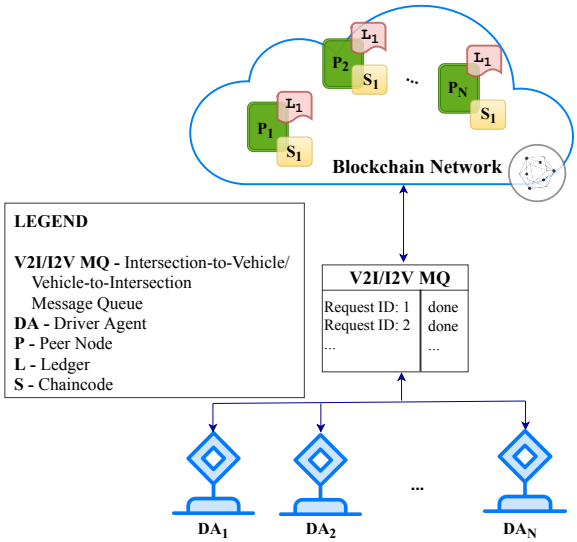


Figure 4.3: Driver Agent and blockchain communication in Proof-of-Concept flow.

Validate scenario We consider two vehicles in the communication range of an IMA. Vehicle V_a drives on the North Street and plans to go to South Street. On the other side, vehicle V_b drives on West street, and the destination is Est street. Trajectories of vehicles are overlapped. Therefore, the collision can occur when V_a turns left, and V_b goes straight if both vehicles arrive at the conflict point simultaneously. Both vehicles send a request to IMA asking to cross the intersection. As shown in Figure 4.4, each request is forwarded to the Intersection Control Policy (ICP), which is part of IMA and performs the trajectory simulation. The simulation results are sent to the Postprocess, which is in charge of generating and sending the appropriate response to the vehicle.

To avoid a collision, IMA extracts the proposed parameters from the crossing request to estimate the arrival time to the intersection and the leaving time with the maximum and minimum acceleration allowance back to the applicant vehicle. Consequently, the IMA status is updated, and the new time slot is transmitted to the applicant vehicle.

In a collision-free simulation, the response would contain the timing index

calculated together with the maximum and minimum acceleration allowance. Otherwise, in a collision, the rejection message would contain restrictions to decrease the velocity or stop to avoid the collision. For instance, we assume that

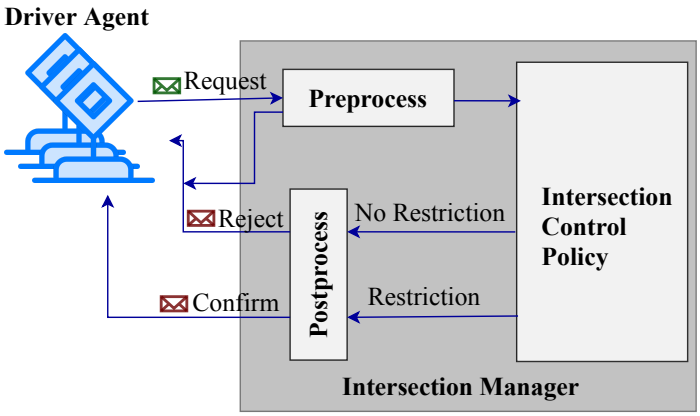


Figure 4.4: AIM simulator main components.

vehicle V_a moves at constant speed and it will arrive at point P_a at time $t = t_a$ with Equation (8.1), where S_a is the distance between the vehicle and P_a point.

$$t_a = \frac{S_a}{v_a} \tag{4.1}$$

As previously said, $L_{carr} = 25\,\text{m}$, $L_{cross} = 4.5\,\text{m}$, and $L_v = 4.5\,\text{m}$, and v_a is the velocity of vehicle V_a. Therefore, the total time necessary to reach the point P_a is given by Equation (4.2).

$$t_a = S_a/v_a = \frac{L_{carr} + L_{cross} + L_v}{v_a} \tag{4.2}$$

In the present case $t_a = 8.5\,\text{s}$ (we are assuming a constant speed of 4 m/s). If the vehicle V_b is not allowed to cross the intersection, it will reach the P_b point after a time equal of greater than t_a. Therefore, vehicle V_b should decelerate to cross the intersection in safety.

$$S = \frac{1}{2}at^2 + v_0 t + S_0 \tag{4.3}$$

Time t will be equal to t_a, space S will be equal to the distance between vehicle V_b and point P_b, speed v_0 will be equal to vehicle V_b speed (assumed constant), S_0 is the assumed starting point of 0 Equation (4.3). So obtaining the following reverse formula for a Equation (4.4).

$$a = 2\frac{S_b - v_b t_a}{{t_a}^2} = 2\frac{S_b - v_b \frac{S_a}{v_a}}{\left(\frac{S_a}{v_a}\right)^2} \tag{4.4}$$

The speed limit and the distance of the communication have to guarantee an acceleration or deceleration applicable. Otherwise, if the vehicle V_b is stopped, it must restart to reach the end of the intersection. Therefore, we suppose that the initial speed of v_i is 0, and the maximum acceleration a. The time necessary to cross the intersection is given by Equation (4.5) and Equation (4.6).

$$S = \frac{1}{2}at^2 + v_i t \tag{4.5}$$

$$t_c = \sqrt{\frac{2S}{a}} \tag{4.6}$$

Design of the Blockchain-based AIM System

According to Figure 4.2, which illustrates the architecture of the whole system, among the components used, a vital one is the IMA. It manages the crossing safety of vehicles through bi-directional communication among them.

The endpoints of this communication are DA and IMA. To have IMA accessible, it should be installed on every road intersection, and it is responsible for deciding which vehicle gets priority to cross the intersection.

As already explained in Section 4.1.3, the main security aspects in an AIM system to be considered are authentication, privacy, and security, reliability, availability, and scalability. To overcome those issues, a decentralized approach is needed; blockchain technology is a potential solution.

We propose to make intersection management a remote service that runs on a blockchain network with a certain number of nodes. We assume that these nodes do not need to be present physically at the intersection location but at a remote location like cloud/edge. First of all, we must establish the following analogies:

- Blockchain nodes vs IMAs: In the blockchain, nodes are connected to the peer-to-peer network; IMAs could also connect to form the peer-to-peer network.
- Chaincode vs ICP: The chaincode is the software program that is deployed on the network. Similarly, the implementation of the ICP could be deployed on the blockchain network.
- Users vs Vehicles: For a transaction in the blockchain, a registered account (public-key encryption) is required. Correspondingly, vehicles would need to possess a registered account to send a request and receive a response.
- Transactions vs Vehicles request/response: Each transaction expresses a request or response between the IMA and vehicles.

In the blockchain, only a valid user can communicate with the nodes. In this case, the users are the vehicles, and the nodes are the IMAs. Thus, the vehicles (users) should possess a valid public and private key to send digitally signed messages to the IMA. These public keys, along with the corresponding vehicle's identity, should be stored by the IMAs as well. Using this public key encryption for digitally signed messages, the IMA can trust the messages are produced only by a valid vehicle rather than an impersonation.

Figure 4.3 illustrates a proof-of-concept flow of the interaction between DAs and the blockchain network, respectively. In detail, the architecture comprises several DAs (DA_i, with i=1...N) and a blockchain network composed of several peer nodes (P_j, with j=1...N) each of which hosts a ledger (L1) and a chaincode (S1). Each peer node hosts an IMA; this configuration allows eliminating the traditional AIM architecture's centralization by overcoming the single point of failure.

As said above, ICP can be deployed as the chaincode and is invoked on all blockchain transactions, which are the vehicle's request and response in this context. Thus, each blockchain network peer (IMA) runs an ICP. This design choice allows us to increase reliability and availability by ensuring a high level of fault tolerance. As well as, we also increase the system's scalability by decreasing the computation time. In fact, in case of an IMA failure between this request/response handling, the vehicle will switch the IMA and send the request again. To make the vehicles aware of the IMA's status of failure (system down), the blockchain framework will timeout this transaction (passage request) and notify the corresponding vehicle. From the blockchain, these requests/responses are stored to form blocks.

Thus, the system's communication flow implies that each DA initializes the process by sending a request message to one of the available blockchain peer nodes (IMAs) running the chaincode (ICP). Based on the request, the chaincode (ICP) performs the simulation to check whether it results in a collision. IMA stores the passage of this vehicle and forwards the response to the DA. In the case of collision-free, the blockchain peer nodes combine to form a block through mining. Otherwise, in case of collision, it sends a reject message to the DA, and a request message is created again for a different time.

4.1.6 Implementation

This section outlines the implementation flow needed to integrate the blockchain network within the AIM simulator to allow peer nodes to serve as IMAs.

A variety of traffic simulation software has been developed to model traffic networks (e.g., SimTrafficis or SUMO), allowing visual simulation of a surface street traffic network. Recently a new tightly integrated traffic simulation/wireless communication system has been developed, which is called AIM 4[1]. AIM 4

1 http://www.cs.utexas.edu/~aim/

was developed by the Learning Agents Research Group, which is part of the AI in the Department of Computer Sciences at the University of Texas at Austin, and it is a traffic simulator that aims to create a scalable, safe, and efficient multi-agent framework for managing autonomous vehicles at intersections. An advanced management system includes a scheduler and a simulation manager to detect vehicles' collisions and traffic congestion. A vital part of AIM is an infrastructure called IMA, which manages the road crossing safely.

The most prevalent blockchain implementations (e.g., Bitcoin and Ethereum) are not suitable for our proposed system as they demand many resources (computation and bandwidth), while in a smart city scenario, vehicular networks require low latency to information exchange. Furthermore, conventional blockchains require from a few seconds up to minutes to insert new information into their distributed ledger (depending on the used consensus algorithm). In previous work, Lunardi et al. [152] presented a lightweight permissioned blockchain that creates blocks on demand. While Li et al. [153] and Sharma et al. [154] presented solutions using blockchain for smart cities, their approach incurs long delays for adding and retrieving information to and from the blockchain. Dinh et al. [155] presented a benchmarking framework, called BLOCKBENCH, for evaluating private Blockchain systems. According to the results obtained and our latency, throughput, and scalability requirements, we adopted Hyperledger Fabric (HF) to implement our blockchain network. HF is the blockchain framework proposed by IBM we have chosen for our experiment, and its capabilities are explained in Section 4.1.4. Hence, we used an HF permissioned Blockchain using the Kafka consensus mechanism.

The smart contract or chaincode is the software program installed or deployed on the blockchain network nodes to run on specific events or transactions. Our solution design is to implement the IMA as a smart contract.

Figure 4.3 illustrates the communication flow in the system where requests from DAs are sent to the smart contract (IMA) on the Blockchain network. Based on the request, the smart contract (IMA) performs the collision test and responds to the DA.

The AIM 4 simulator's source code has been appropriately modified to implement the Hyperledger Protocol, a communication protocol that stores and implements the V2I/I2V requests/responses to/from the IMA implemented through a Node-RED flow. Node-RED is one of the best known flow-based programming tools for IoT. Its event-driven JavaScript model and the asynchronous execution of I/O operations allow developing scalable solutions for real-time data flows analysis. Node-RED gives the possibility to connect different devices (with any sensors and actuators) and APIs and online services to create highly integrated and complex systems straightforwardly and intuitively.

Blockchain Network Setup

The implementation choice for the architecture and scenario described in Section 4.1.4 and Section 4.1.5 goes towards a permissioned blockchain. The reason is given by the fact that the network requires access control. In particular, a PKI refers to a centralized (root) management controlling all the intersections.

Proper matching for this type of configuration is represented by a HF setup where each organization is a jurisdiction, and the ordering service resides at the central level. The latter would run next to the CA to issue certificates for blockchain nodes (peers in the Fabric denomination) in the jurisdictions. Alternatively, a dedicated CA service can run in parallel. A peer is responsible for endorsing transactions submitted to it and runs a smart contract that can be different across jurisdictions. This allows a certain degree of freedom in the policies implemented in each jurisdiction and decentralizes the computational load of the misbehaviour detection process.

In this context, the DAs represent the client that interfaces directly with the peer in a jurisdiction and ordering service.

The functioning of the setup described above is the following:

1. Transaction submission. A DA submits a transaction proposal relating to crossing the peer node associated with the intersection in its jurisdiction.

2. Transaction verification. The peer verifies the transaction's validity, simulates the execution of the related smart contract functionality, and returns an endorsement result to the DA.

3. Transactions ordering. The DA sends the transaction proposal and the related result to the ordering service. The last orders all the transactions coming from different DAs, build the blocks, and broadcast them to all the different jurisdictions' peer nodes.

4. Transaction commitment. Upon verifying their validity, peer nodes write the blocks of transactions received from the ordering service in their local copy of the Blockchain.

However, all the peer nodes receive and synchronize the transactions that occurred in all the jurisdictions. From the description above, it emerges that the endorsement of transactions that occurred at the jurisdiction level is done by one peer node in that jurisdiction, and the DA submits the transaction. To make the process more robust in terms of consensus, a first option could be to require the endorsement of more than one peer node in the same jurisdiction. To this aim, in this setup, we focused on a single jurisdiction with several peer nodes; several IMAs manage each intersection. For instance, we used an HF business network topology with varying peers (hence, varying number of IMAs), 1 orderer, and 1 organization to support this setup. Before creating a Business

Network Definition (BND), a Peer Admin Card must be generated, which is necessary to control all network peers. This can be achieved by executing the following command, as illustrated in Listing 4.1.

Listing 4.1: Peer admin card generation commmand.

```
composer card create -p DevServer_connection.json \
    -u PeerAdmin -c "${CERT}" -k "${PRIVATE_KEY}" \
    -r PeerAdmin -r ChannelAdmin \
    -- file PeerAdmin@fabricgp.card
```

Thus, by analyzing the command, we notice it requires the following parameters: DevServer_connection.json contains the network configuration; ${CERT} indicates the certificate path relative to the Admin user; ${PRIVATE_KEY}, on the other hand, indicates the path to the Admin user's private key. These files were previously generated with the *cryptogen* command and are located within the crypto-config directory. To use the newly created id card, we need to import it with the command shown in Listing 4.2.

Listing 4.2: Peer admin card usage commmand.

```
card import -- file ./ PeerAdmin@fabricgp.card
```

Next, we can create our BND; the most straightforward way is using the Yeoman generator to create a corporate network skeleton. This will create a directory containing all the components of a business network. The network model skeleton can be generated with the command shown in Listing 4.3.

Listing 4.3: Network model skeleton creation commmand.

```
yo hyperledger-composer:businessnetwork
```

Generally, a business network consists of resources, participants, transactions, access control rules and, optionally, events and queries. The template file (.cto), written using the modelling language of Hyperledger Composer, contains the class definitions for all resources, participants, and transactions in the business network, as illustrated in Listing 4.4. It also contains an access control file (*permissions.acl*) with basic access control rules, a script file (*logic.js*) containing transaction processor functions, and a *package.json* file containing the business network's metadata.

Listing 4.4: "org.aim.net.cto"

```
namespace org.aim.net
participant ImParticipant identified by id {
    o String id
    o String name
}
asset IntersectionAsset identified by id {
    o String id
```

```
    o Integer vin
    --> ImParticipant im
    o String startTime
    o Integer reqID
    o Double atTime
    o Double atSpeed
    o Integer arrLane
    o Integer depLane
    o String status
}
transaction ImTransaction {
    o Integer vin
    o Integer im
    o Integer reqID
    o String startTime
    o Double atTime
    o Double atSpeed
    o Integer arrLane
    o Integer depLane
    o String assetId
}
transaction ImResponse {
    --> IntersectionAsset asset
    o String status
}
event ImEvent {
    o Integer vin
    o Integer im
    o Integer reqID
    o String startTime
    o Double atTime
    o Double atSpeed
    o Integer arrLane
    o Integer depLane
    o String assetId
}
```

Listing 4.5: ImTransaction implementation.

```
async function ImTransaction(tx) {
    let assetRegistry = await
        getAssetRegistry('org.aim.net.IntersectionAsset');
    let asset=getFactory().newResource('org.aim.net',
        'IntersectionAsset', tx.assetId);
    let event=getFactory().newEvent('org.aim.net', 'ImEvent');
    asset.vin=event.vin=tx.vin;
    asset.im=getCurrentParticipant();
    event.im=tx.im;
```

```
        asset.startTime=event.startTime=tx.startTime;
        asset.reqID=event.reqID=tx.reqID;
        asset.atTime=event.atTime=tx.atTime;
        asset.atSpeed=event.atSpeed=tx.atSpeed;
        asset.arrLane=event.arrLane=tx.arrLane;
        asset.depLane=event.depLane=tx.depLane;
        event.assetId=tx.assetId;
        asset.status="";
        await assetRegistry.add(asset);
        emit(event);
    }
```

Listing 4.6: ImResponse implementation.

```
    async function ImResponse(tx) {
        let asset = tx.asset;
        asset.status = tx.status;
        let assetRegistry = await
        getAssetRegistry('org.aim.net.IntersectionAsset');
        await assetRegistry.update(asset);
    }
```

Thus, in Listing 4.4, the ImParticipant class is used to identify each IMA with an id and a name. As well, the IntersectionAsset class allows us to store all the vehicle requests in the Blockchain. Each request is identified by an id and is correlated with the IMA that handled them. Next, the ImTransaction class defines a transaction model necessary to create the initial asset of requests forwarded by vehicles and set the initial status to "Pending"; the implementation is illustrated in Listing 4.5. Then, the ImResponse class defines a transaction model that allows updating the status of a request that has been evaluated to "Accepted" or "Rejected", whereas ImResponse is shown in Listing 4.6. Finally, the ImEvent class defines the event that is generated by the ImTrasaction transaction.

After the business network has been defined, it is necessary to pack it into a .bna (business network archive) *aim-network@0.0.1.bna*. To install the business network, the command illustrated in Listing 4.7 must be executed.

Listing 4.7: Command to install the business network.

```
    composer network install --card PeerAdmin@fabricgp \
    --archiveFile aim-network@0.0.1.bna
```

Finally, to run the business network, it is necessary to execute the command shown in Listing 4.8.

Listing 4.8: Command to run the business network.

```
    composer network start
        --networkName aim-network \
```

```
    --networkVersion 0.0.1 --networkAdmin admin \
    --networkAdminEnrollSecret adminpw \
    --card PeerAdmin@fabricgp \
    -- file networkadmin.card
```

Thus, by analyzing the command, a *networkadmin.card* id card is created inside the directory, necessary for the management of the business network that must be imported with the command shown in Listing 4.9.

Listing 4.9: Command to import the network admin card.

```
composer card import -- file networkadmin.card
```

AIM Simulator Setup

By integrating the Hyperledger Protocol with the AIM 4 simulator, we can set the following parameters from the configuration panel:

- *Server Address* - defines the address of the server where the Node-RED flow is deployed;
- *Traffic level* - defines the number of vehicles that are generated every hour for each lane;
- *Number of N-b S-b roads* - defines the number of intersections in the map, which may be 1 or 2 for our protocol;
- *Number of seconds per Simulation* - defines the duration in seconds of each simulation;
- *Number of Simulations* - defines the number of times that simulation must be repeated.

For instance, to support the Hyperledger Protocol integration, two fundamental methods have been implemented into the AIM 4 simulator: deliverV2IMessages and deliverI2VMessages. The first method forwards the vehicle's requests to the Node-RED server.

Each request encloses the following information: *vin* - the identification number of the vehicle, *im* - the identification number of the IMA, *reqID* - the identification number of the request, *startTime* - the time stamp of the start of the simulation, *atTime* - the time at which the vehicle arrives at the intersection, atSpeed - the expected arrival speed, *arrLane* - the arrival lane, depLane - the destination lane and finally *assetId* - the unique identification code of the request. This information is needed to generate an ImTransaction transaction in HF, which will create an IntesectionAsset containing all request details. The initial request's status is set to *"Pending"*.

On the other hand, the second method (deliverI2VMessages) is responsible for forwarding to the vehicles the result of the request that can be “Accepted” or “Rejected”. In the case of “Accepted” request, the vehicle must cross the intersection, respecting exactly the parameters supplied during the request phase. In the case of “Rejected” request, the vehicle will have to re-formulate a new request as the one already submitted was found to be ineligible and follow the constraints indicated by the IMA. Therefore, we deployed 2 Node-RED flows that allow the management of 1 or 2 intersections with the corresponding IMAs.

One intersection Figure 4.5 illustrates the flow used to manage the system with one intersection. Each vehicle forwards its request through the POST method on the path “/rest” (“[post]/rest” block); the request generates a Blockchain transaction through the “ImTransaction” block, which creates a new asset containing the details of the request into the Blockchain. Then, once executed, the transaction generates an event that is captured by the “HL Event” block. The event encloses the details of the request that has been inserted in the Blockchain and which status that is initially set to *"Pending"* given the request has not been evaluated yet. At this point, the request is inserted in the queue (“queue” block) and then evaluated by the “evaluate request” block, which, after having carried out the necessary checks, updates the status of the request by forwarding the result both on the “insert” block of MongoDB and on the “ImResponse” block of Hyperledger Composer. The “evaluate request” block hosts the algorithm that evaluates all the vehicles’ incoming requests. The algorithm, written in JavaScript, performs simple checks to determine if a particular request has to be accepted or rejected. The “ImResponse” block generates a blockchain transaction that updates the status of the request to “Accepted” or “Rejected”. Vehicles will obtain the status of their requests by using the GET method (“[get]/rest” block), which forwards to the “find request result” node that searches the request in the database (“build response” block). The result is forwarded to the vehicle through the “http” block. Once the crossing is finished, the vehicle returns information using the POST method using the [post]/stats block; these are used for statistical purposes. For each vehicle, the following information is stored: *vin* - the vehicle identification number, *rstReq* - indicates when the first crossing request was made, *lastReq* - indicates when the last crossing request was made, *numReqs* - the total number of requests made before being able to cross the intersection, *rstDist* - the distance from the intersection to the moment of the first request; *lastDist* - the distance from the intersection to the moment of the last request, *arrLane* - the arrival lane, *depLane* - the destination lane, *accTime* - indicates when the crossing request has been accepted, and finally *vehicles* - the number of vehicles present near the intersection (including vehicles that have just crossed it).

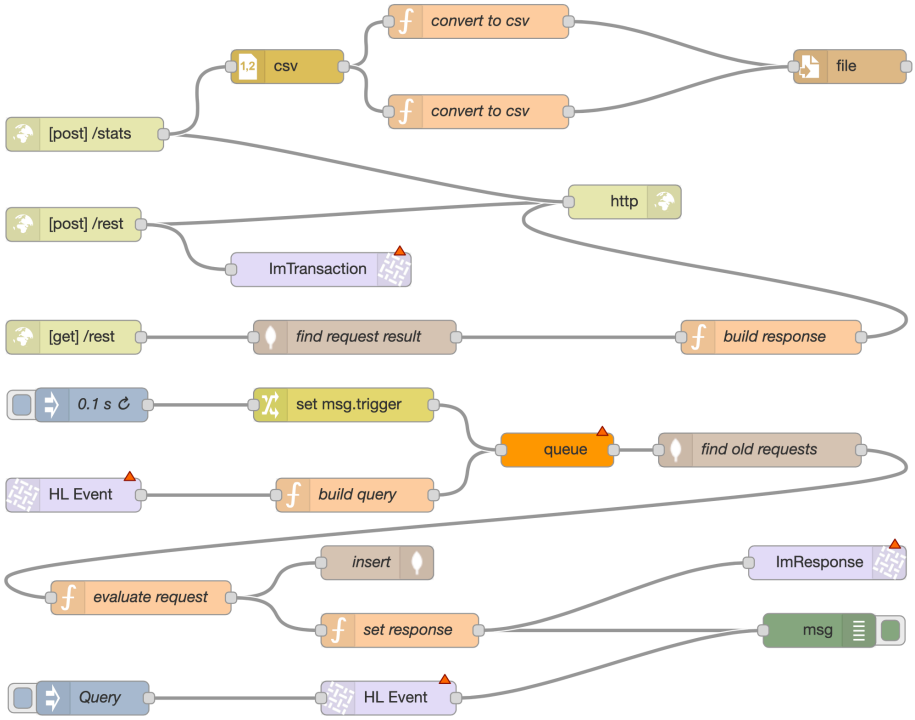


Figure 4.5: Node-RED flow - one intersection.

Two intersections Figure 4.6 illustrates the flow used to manage the system with 2 intersections. As in the previous flow, to concurrently manage the requests coming from the 2 intersections, two different paths have been created ("[post]/rest#1" and "[post]/rest#2" blocks) for the "http" endpoint. To concurrently manage the requests generated by the DAs within the range of both intersections that generate blockchain transactions, two "ImTransaction#1" and "ImTransaction#2" blocks have been used. Each one creates a new asset containing the details of the request into the Blockchain. In this flow, we also notice the presence of two distinct "ImResponse#1" and "ImResponse#2" blocks.

Each IMA has its own identity associated with the relative "Participant" of Hyperledger Composer. In this way, the transactions generate assets owned by the Intersection Manager that handled the transaction. The "HL Event" block captures the events generated by the requests from both intersections and then are separated into two queues ("queue#1" and "queue#2" blocks) to be evaluated in parallel by the two "evaluate request" blocks.

In this configuration, a "set lane value" JavaScript block has been used to monitor the number of vehicles in the lane which connects the two intersections for statistical purposes. As in the previous flow, two endpoints ("[post]/stats#1" and "[post]/stats#2" blocks) have been created to collect statistical data.

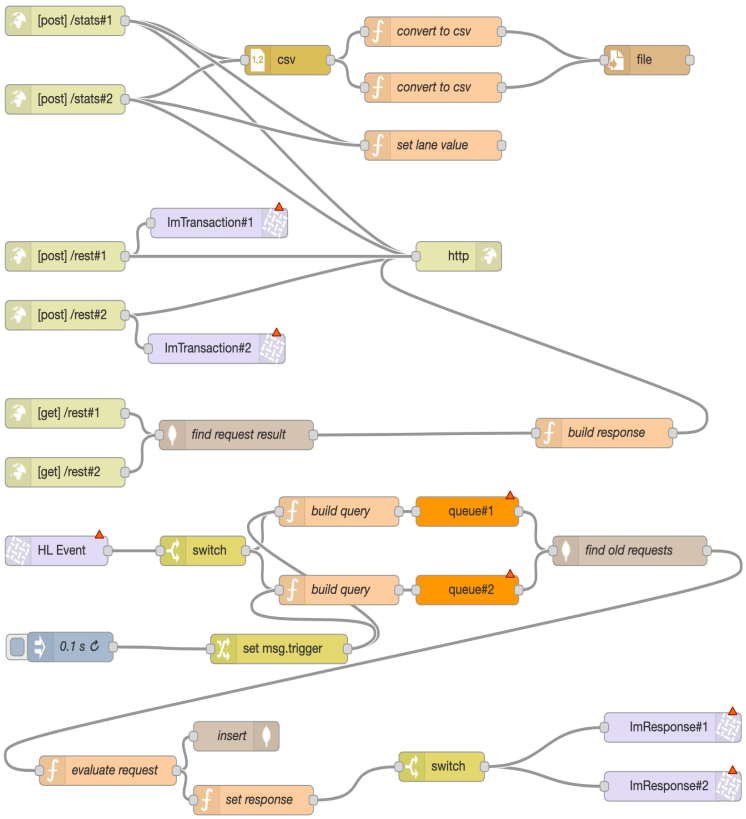


Figure 4.6: Node-RED flow - two intersections.

4.1.7 Performance

This section thoroughly tests all of the proposed BaaS-based AIM system features to demonstrate its feasibility. The experiments carried out allow quantifying the system's performance using the different number of intersections and road traffic configurations, respectively. Since V2I/I2V communications depend on the network's status, we concentrated our attention on how the overhead introduced by blockchain impacts the performance of our proof-of-concept and on its suitability for real-time use. Hence, we distinguish between 2 different performance scenarios. We start Scenario #1 by analyzing the impact of both the number of intersections and the number of vehicles per hour per lane (vehicles/hour/lane) on the system performance. In Scenario #2, we analyze the impact of the number of intersections, the number of peer nodes and the number of transactions sent per second on the system performance.

Experimental Setup

The experiments were carried out for both scenarios using a prototype version of the simulator, which was fully described in our previous work [138].

For the simulation purpose, the vehicle's dynamics are configured according to the geometry of the intersection. The intersection in the simulation is the same as the one shown in Section 4.1.5. As previously mentioned in Section 4.1.5, each vehicle has length L_v = 4.5 m and width W_v = 1.85 m. Each intersection contains one lane traveling in each direction and the maximum allowance of driving velocity is v_{max} = 60 m/s and the minimum is v_{min} = -17 m/s. The maximum acceleration is a_{max} = 4.5 m/s^2 and the minimum is a_{min} = -45 m/s^2. The length of each incoming street is 500 m, and the V2I communication range with the manager is set to R = 149 m. Specifically, we analyzed two different scenarios configured according to the parameters specified in Table 4.1 and Table 4.2.

Table 4.1: Scenario #1: Simulation Parameters.

Parameter	Set of Values	Unit
Traffic Level	{200, 400, 600, 800}	vehicles/hour/lane
Simulation Time	15	minutes
Number of Intersections	{1, 2}	-
Number of Peer Nodes	{1}	-

Hence, according to Table 4.1, in Scenario #1, the Blockchain-based system has 1 peer node, and the simulator is configured with (i) one intersection and (ii) two intersections, respectively. We, therefore, performed a scalability analysis by varying the number of vehicles/hour/lane from (a) 200, (b) 400, (c) 600, up to (d) 800, respectively. As reported in Table 4.2, in Scenario #2, the Blockchain-based system is configured with varying number of peer nodes from (a) 1, (b) 4, (b) 8, up to (d) 12 respectively. We performed a scalability analysis by varying the number of transactions per second (tps) from (a) 10, (b) 20, up to (c) 30, respectively.

Table 4.2: Scenario #2: Simulation Parameters

Parameter	Set of Values	Unit
Transactions Per Second	{10, 20, 30}	tps
Simulation Time	15	minutes
Number of Intersections	{1}	-
Number of Peer Nodes	{1, 4, 8, 12}	-

For each experiment, we fixed 15 minutes of simulation time and calculated the number of vehicles served, waiting time, response latency, throughput,

distance, and the number of requests. To have accurate results, we performed 30 subsequent iterations and calculated mean values and 95% confidence intervals, respectively.

Each performance is assessed according to the following criteria:

1. *Number of completed vehicles* - defines the total vehicles served during the simulation.
2. *Waiting time* - is the time interval a vehicle waits at the intersection before crossing.
3. *Response time* - is the time interval between when a vehicle first asks for permission to cross the intersection and when it is granted permission.
4. *Latency* - is the increase in travel time due to the presence of an intersection. Travel time is defined as the period between the moment when a vehicle gets through the intersection and the moment when a vehicle enters the V2I range. Then delay can be calculated as the difference between the real travel time under intersection control and the estimated travel time if the vehicle travels at the speed limit for the entire journey.
5. *Throughput* - quantifies the number of vehicles that pass through the intersection during a simulation.
6. *Distance* - represents the distance from the center of the intersection where the IMA has confirmed the crossing request sent by the vehicle.
7. *Number of requests* - totalizes the number of requests a vehicle does before getting a confirmation from the IMA.

Testbed Configuration

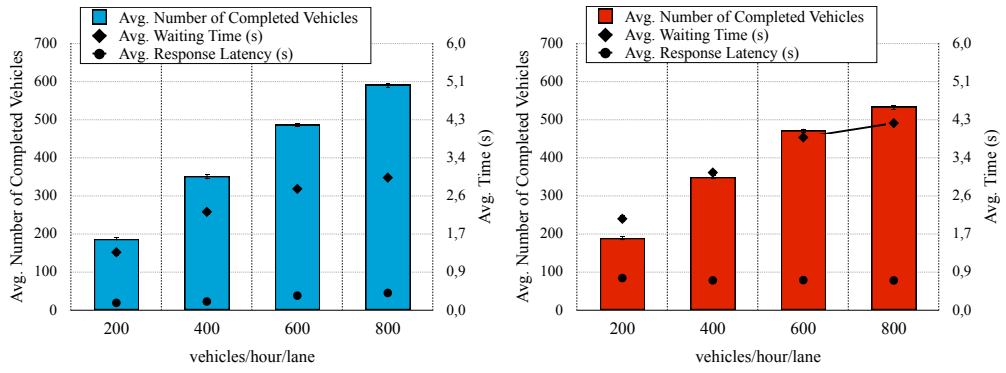
The hardware and software characteristics of each server of our testbed have Intel(R) Core(TM) i7-6920HQ @3.4 GHz, 4 GB of RAM and 64-bit Ubuntu Server 18.04.01 LTS. We used the Node-RED 0.18 version, Docker 17.03.0, MongoDB 3.6 version, Hyperledger Fabric v1.1, and Hyperledger Composer v0.19.11, v1.8 of docker-compose, v8.9 of node, v5.7.1 of npm and Python v2.7 and AIM 4. In Scenario #1, both existing and proposed systems are deployed on 2 servers. For instance, to set up the existing system, we deployed the AIM 4 simulator on one server and on another one the Node-RED and MongoDB instances. On the other hand, the proposed system was configured with an HF network having 1 peer node, 1 organization, and 1 orderer. Therefore, on one server we deployed the AIM 4 simulator and a peer node while the remaining HF's components and the Node-RED instance.

To set up Scenario #2, we implemented an HF network with 1, 4, 8, and 12

peer nodes, 1 organization and 1 orderer. In the configuration with one peer node, all the HF's components were deployed on one server. For the remaining configurations with 4, 8, and 12 peer nodes, respectively, we solely deployed the peer node on the different number of servers. The AIM 4 simulator was deployed to an additional server with the same software and hardware characteristics.

Scenario #1

One intersection Figure 4.7 illustrates the avg. number of completed vehicles, avg. waiting time and avg. response latency for both existing and proposed systems given certain road traffic configurations. Next, Figure 4.9 shows the avg. distance (m) and the avg. number of requests as well. For instance, for both with and without Blockchain systems, we varied the number of vehicles/hour/lane from 200 up to 800 and gathered the above-specified metrics. In Figure 4.7, it can be found that both systems can manage more or less the same number of vehicles until the traffic is around 500 - 600 vehicles/hour/lane; as the traffic gets heavier, the Blockchain-based system has worse performance and creates long queues at the intersection. According to Figure 4.7, the Blockchain-based system has a significantly high average waiting time per vehicle comparing to the existing one. It can be seen that the Blockchain-based implementation does not show a meaningful improvement, the average waiting time performances per vehicle are always greater (≈31%) and grow faster than those obtained with the existing one as well as, as the number of vehicles increases, the response latency of both implementations follows a linear trend. It can be seen that the avg. response latency of the Blockchain-based system is around 0.67 s while that collected with the other one without Blockchain is below the thousandth of a second.



(a) No blockchain **(b)** Blockchain

Figure 4.7: Scenario #1: one intersection (avg. number of completed vehicles, avg. waiting time (s), avg. response latency (s)).

We next examined the avg. distance and the avg. number of requests. For

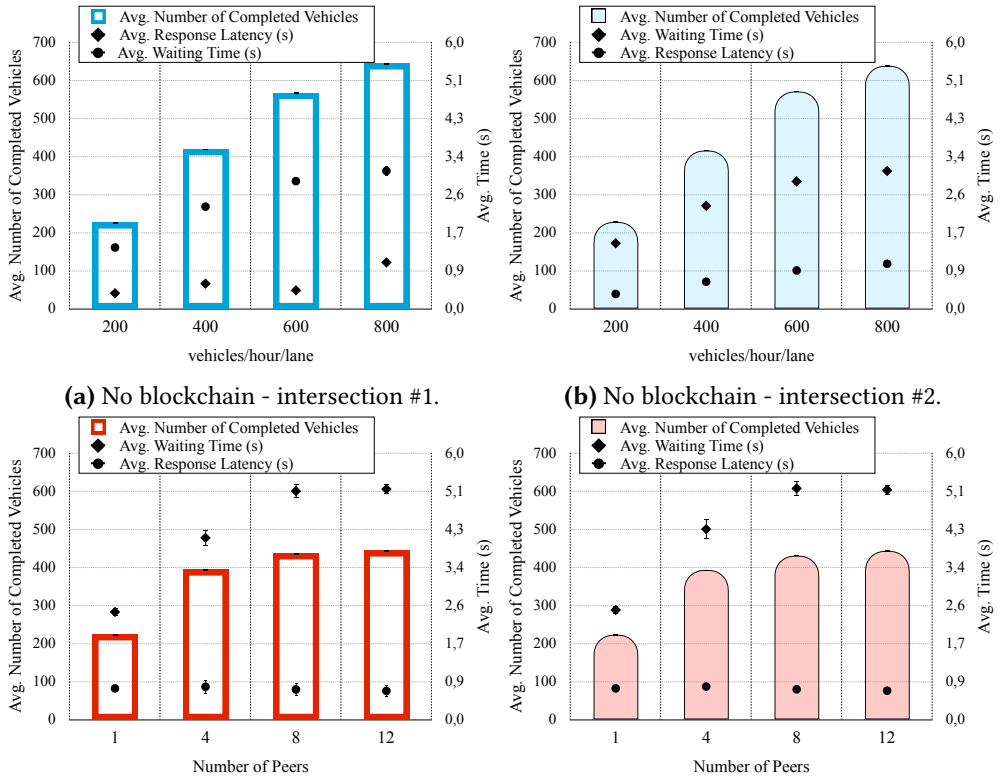
instance, as can be seen from Figure 4.9 (a), the results show that more the number of vehicles/hour/lane increases, more the avg. number of requests increases, while the avg. distance decreases. This can be explained by the fact that more crossing requests are received when the number of vehicles/hour/lane becomes important more the system cannot handle them; this produces long queues at the intersection. As a consequence, each vehicle will have to stop and make more than one crossing request. Thus, as a consequence, the avg. distance will decrease and the avg. number of requests will increase. Similar behavior is also registered with the proposed system (see Figure 4.9 (b)). In detail, with the blockchain-based system, there is an increase of ≈13% on average for avg. number of requests and a decrease of ≈21% on average for the avg. distance from the intersection.

Furthermore, by comparing the avg. number of requests of both existing and proposed systems, we notice both systems behave similarly when the number of vehicles/hour/lane is almost 500 - 600; exceeded this threshold, the avg. distance in the existing systems is higher and the avg. number of requests is lower respect to those in the proposed one. Besides, it has been noticed that the number of transactions sent per second has an impact on the output of the proposed system due to the overhead introduced by HF. The existing system performs better than the proposed one.

Two intersections We gathered the same metrics using two intersections as well. Figure 4.8 reveals the number of vehicles that have completed the intersection crossing during the simulation. It can be noticed that both implemented systems are able to manage more or less the same number of vehicles; while for one intersection, this was possible until the traffic was around 500-600 vehicles/hour/lane, for two intersections, this is possible up to a lower threshold of about 400 vehicles/hour/lane. Under heavy traffic conditions greater than 400 vehicles/hour/lane, as for one intersection, the existing system performs better compared to the proposed one.

In terms of avg. waiting time, it can be noticed that the existing system outperforms the proposed one. In the presence of heavy traffic, once the threshold of 400 vehicles/hour/lane is exceeded, the waiting time saturates and becomes constant. Because of the long queues formed in the intersections, everyone will wait more or less the same time before crossing the intersection. The same trend is also registered for the avg. response latency.

Next, we illustrate the avg. distance and the avg. number of requests for each intersection using the existing (see Figure 4.8 (a) and Figure 4.8 (b)) and proposed (see Figure 4.8 (c) and Figure 4.8 (d)) systems. As we expected, the trends are similar for both IM0 and IM1. This is coherent with what has been previously obtained. Indeed, as the number of vehicles/hour/lane becomes important the avg. distance decreases, and the avg. number of requests increases. As previously explained, this behavior is because the system cannot handle the incoming



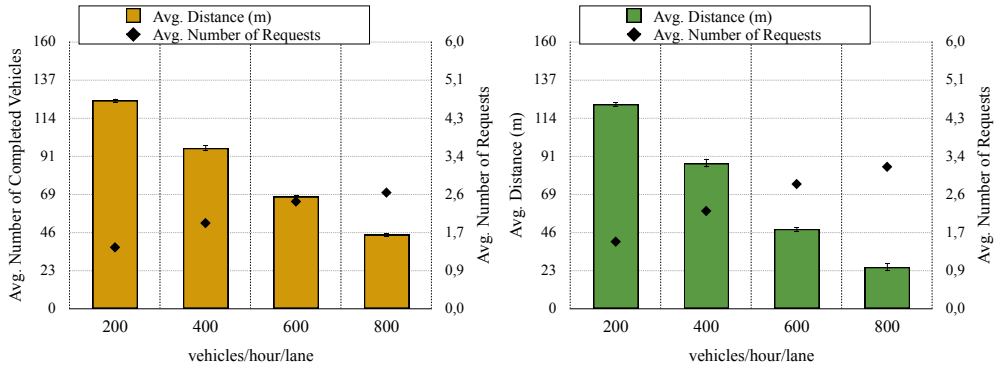
(a) No blockchain - intersection #1.

(b) No blockchain - intersection #2.

(c) Blockchain - intersection #1.

(d) Blockchain - intersection #2.

Figure 4.8: Scenario #1: two intersections (avg. number of completed vehicles, avg. waiting time (s), avg. response latency (s)).

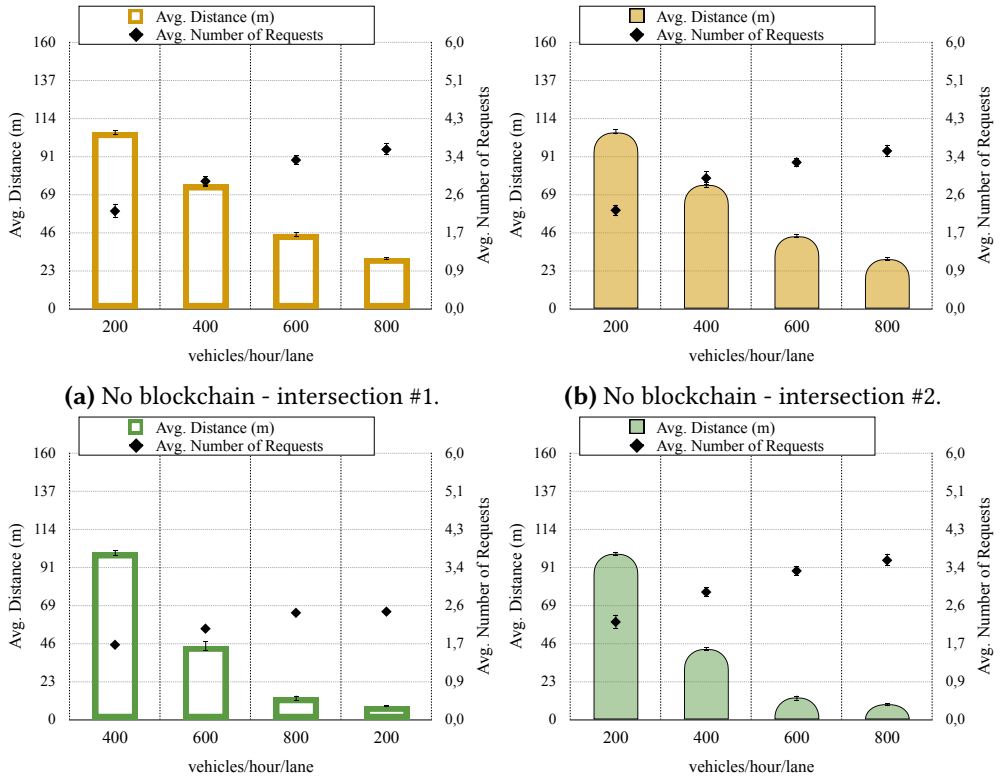


(a) No blockchain

(b) Blockchain

Figure 4.9: Scenario #1: one intersection (avg. distance (m), avg. number of requests).

crossing requests efficiently. Each vehicle has to make the crossing request more than one time and enqueue at the intersection. By comparing these results with those obtained with one intersection, we notice the avg. distance is lower and the avg. number of requests is higher as well. This can be explained by the



(a) No blockchain - intersection #1.

(b) No blockchain - intersection #2.

(c) Blockchain - intersection #1.

(d) Blockchain - intersection #2.

Figure 4.10: Scenario #1: two intersections (avg. distance (m), avg. number of requests).

fact that as the intersections are 2, both existing and proposed systems saturate and have limitations to manage them with the given hardware and software configurations.

Scenario #2

In the following, the impact of the network topology is studied. To do the assessment, the number of peer nodes has been fixed from 1, 4, 8 up to 12, and the input transactions number is varying. Figure 4.11 shows the avg. latency and the avg. throughput of the proposed systems with one intersection by varying the number of transactions per second (tps) from (a) 10, (b) 20, up to (c) 30, respectively.

It can be seen that, as the number of peers increases, the performance becomes worse as there are more servers, meaning that the system incurs some network overheads. Because HF is communication bound, having more servers means more messages being exchanged and higher overheads. The peer nodes use more time to handle the communication between them rather than to manage the incoming transactions. More communications are required to pre-vote and pre-commit a block when the number of peers (validators) becomes important.

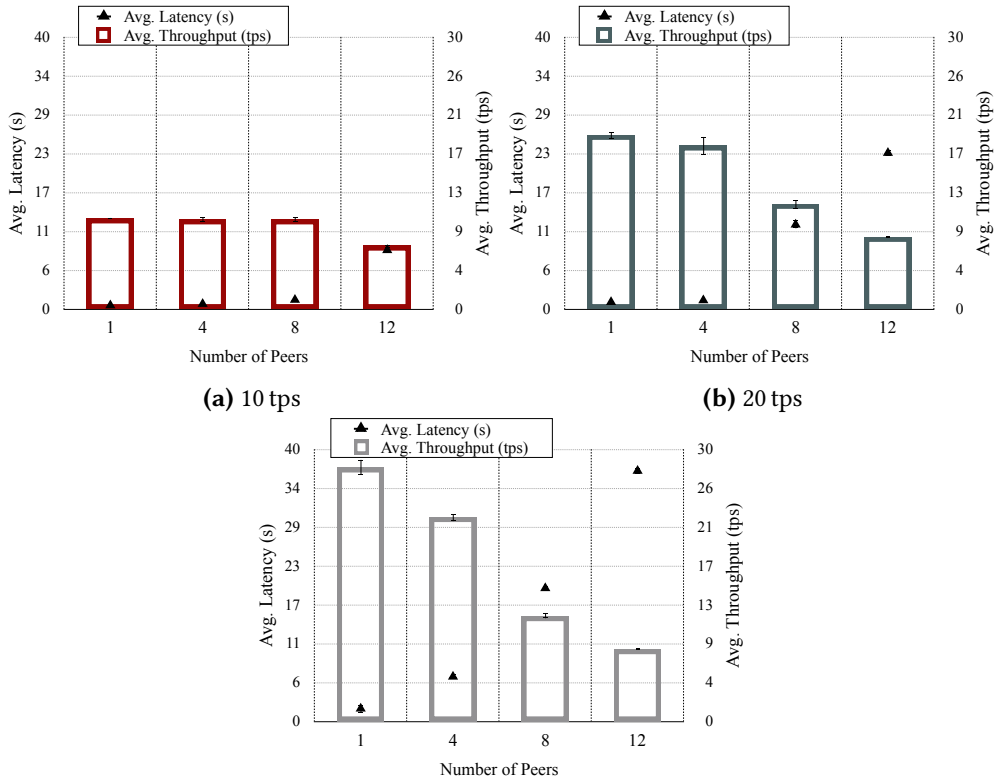


Figure 4.11: Scenario #2: avg. latency (s) and avg. throughput (tps).

Besides, it has been noticed that the number of transactions sent per second has an impact on the output of the blockchain. This is coherent with what has been previously obtained. Indeed, in a network containing an important number of validators, a validator node requires additional time to inform/get informed about the state of other nodes. That will undoubtedly delay the validation/creation of new blocks in the network. This can be explained by the fact that in a complete network, the validator nodes spend more time to synchronize their state with the rest of the network. As a result, such nodes have less time to accomplish the validation work.

Figure 4.11 (a) reveals the throughput and the avg. latency when the number of transactions per second (tps) is 10. We notice the system can handle all the incoming requests, and thus maintaining a high throughput value equals to 10 tps and a low avg. latency, until the system reaches 8 peers; exceeded the number of peers, the throughput is lower and the avg. latency increases.

As well, Figure 4.11 (b) and Figure 4.11 (c) illustrates the same metrics using several transactions per second (tps) equals to 20 and 30, respectively. In this configuration, we notice the system can maintain a high throughput and a low latency when the HF network has 1 and 4 peers, respectively. The configuration

with 12 peers has significantly lower throughput values; therefore, a major number of peer nodes cannot guarantee to have lower minimum delays and maximum throughput values than those obtained with 1 and 4 peer node/s. More precisely, this can be explained by the fact that more vehicles and more transactions are sent per second in the proposed system, the more the network accumulates some delays to validate transactions.

From the above results, it can be concluded that the performance of the blockchain HF technology will undoubtedly limit its usage in practice to specific use cases where the time required to validate a transaction is very high, and the number of validators joining the network is not very important.

4.1.8 Conclusions and Future Directions

New emerging ICT technologies such as IoT, CC, EC, and blockchain that are revolutionizing the whole ITS sector will represent the future research topics for both industry and academic contexts in the next years. In particular, blockchain technology's apparent advantages will open toward new favourable horizons regarding its integration in AIM systems.

This paper proposed a reliable and secure MA-AIM system based on both C2T paradigms and blockchain. The proposed systems use V2I/I2V communication to exchange information between AVs and an IMA placed on each intersection. Specifically, we implemented a proof of concept integrating the AIM 4.0 simulation tool with Node-RED and HF. We validated the proposed system's performance by conducting several experiments by varying the number of AVs crossing through one and two intersections. The collected results helped us to quantify the overhead introduced by the proposed HF-based implementation compared with an alternative solution not using blockchain.

In future works, we initially plan to study further the integration of both V2V and V2I communication capabilities to improve intersection management further. Moreover, particular attention will be given to optimising the algorithm to calculate the trajectory of vehicles approaching the intersection. In the end, we plan to investigate an Infrastructure-to-Infrastructure (I2I) communication scenario, including multiple intersections, by creating a federated private blockchain infrastructure.

4.2 BaaS-based Health Information Exchange System for Patient Monitoring

4.2.1 Introduction

The healthcare industry has been slower than others in adopting IoT technologies, but the rapid growth of IoT and wearable devices has opened up new possibilities in the realm of medical equipment, particularly for remote patient monitoring.

Monitoring devices can be seen as the Internet of Medical Things (IoMT) or healthcare IoT, characterized by seamless data exchange between connected devices followed by efficient data analytics. Electronic Health/Medical Records (EHRs/EMRs) are a key element of this new approach.

A new trend in eHealth information management arose in recent years, which pushes towards patient-centred interoperability [156]. Unfortunately, patient-centred interoperability brings new security and privacy challenges and issues that need to be addressed, and many of them cannot be resolved with traditional interoperability solutions [157]. Thus, it is necessary to look for novel or disruptive interventions that can facilitate the transition towards patient-centred interoperability. Blockchain is a promising technology that can have a key role in this direction and associated challenges.

To deal with security issues in health information systems, we propose a Blockchain-as-a-Service based solution for Health Information Exchange (HIE) or BaaS-HIE suitable for an EHRs/EMRs-IoMT scenario. Our proposed system addresses the four major issues discussed right now, i.e., i) patient privacy and medical record integrity, ii) slow access, granular access authorization and access control, iii) private and auditable healthcare data sharing, and iv) system interoperability. In our BaaS-HIE system, the concept of *smart contract* is crucial because it allows different administrations to implement trustless functionalities in the system. In this way, we have secure remote monitoring of patients, we provide medical professionals with real-time notifications on patients' status, and we propagate health notifications from multiple devices through a seamless system. Using smart contracts is a revolutionary approach that easily integrates new medical technologies in their health solutions. This contribution provides some experimental analysis to show the BaaS-HIE system's feasibility and discuss how the information system's performance is related to the size of EHR.

The remainder of this contribution is organized as follows. We provide motivations and related works in Section 4.2.2. In Section 4.2.3, the design of the BaaS-HIE system is presented, and our implementation is presented in Section 4.2.4. Experimental analysis is discussed in Section 4.2.5. Then, conclusions are provided in Section 4.2.6.

4.2.2 Motivations and Related Works

Digital and networked health services, such as EHR/EMR, play a key role in the storage, sharing and maintenance of patients' personal and medical records [158]. The inability to access medical data (generated by a doctor or device) quickly and efficiently is a long-standing problem in providing electronic healthcare solutions globally. Current EHRs/EMRs implementations assume that a patient meets professionals in a clinic (EMR) or in a political jurisdiction (EHR managed by a province or a state). This assumption is based on a practitioner-centric approach, and the opportunity to make systems interoperable is low.

In reality, patients do not interact with a single doctor or clinic. A survey published in 2010 already stated that patients meet 18.7 different doctors during their lives [159]. This study did not consider the broader population of health practitioners (i.e., pharmacists, physiotherapists, chiropractors). Patients also do not restrict themselves to a single EHR coverage area; they travel for leisure, work, and move for long periods. Now the situation is even worst because patients are increasingly interested in managing their health with the help of data generated by wearable devices [160]. This kind of Patient-Generated Health Data (PGHD) is easily shared with the device/service provider but not with health practitioners. Despite huge investments by practitioners, medical facilities, the provision of medical services and health management continue to be hampered by inaccessibility to information, poor interoperability, inconsistent security of data assets, cumbersome privacy controls, and exclusion of patients from access control to data resources that they are the legitimate owners. However, many shortcomings might lead to the loss of sensitive patient medical information. For example, using current approaches to healthcare systems management makes it difficult for patients to track what entity is accessing health data and its purpose. Therefore, this work's core motivation is to use blockchain for healthcare systems and address the current HIE systems' potential shortcomings.

Blockchain technology can be vital in these cases because it provides data ledger-based features distributed to all network entities. The involvement of blockchain across the healthcare sector has been a growing area of interest in both academic research and industry [161, 162, 163]. [162] claim ownership of the original attempt to employ blockchain for an EHR management system. However, there have been various attempts to address the security and privacy aspects of EHRs/EMRs in healthcare. In particular, in [164] authors studied the decentralization of privacy using blockchain to protect personal data. As well as, in [165], the authors proposed an application of the blockchain for authentication and validation of identity. [162] proposes an access architecture, Healthcare Data Gateways (HDG), for blockchain use in eHealth. [166] constructs an access control scheme, called ESPAC, to implement granularity authorization for data queries based upon attribute-based encryption (ABE) in eHealth. The use of blockchain for medical data access was demonstrated in a system called MedRec [167]. This decentralized system handles patients' EHRs and offers a possibility to retrieve their medical information from different service providers.

Therefore, there is substantial work in integrating blockchain technology in the healthcare domain. We identified four major issues to be addressed: (i) patient privacy and EHRs/EMRs integrity, (ii) slow access, granular access authorization and access control, (iii) private and auditable healthcare data sharing and (iv) system interoperability. Compared with the existing blockchain advances in eHealth mentioned above, we highlight the following novel contributions.

First of all, we address the four major issues addressed above by considering an EHRs/EMRs-IoMT scenario, as illustrated in Figure 4.12 and explained in Section 4.2.3, where the patient is monitored, and the data collected by the sensors are accurately formatted and aggregated using the patient's mobile device as Gateway/Aggregator point. Each IoMT device is authenticated within the patient's Gateway/Aggregator point. Our Baas-HIE system can facilitate this patient-centric transition by surmounting the current HIE systems' interoperability barriers, explicitly focusing on clinical data transaction volume, privacy, and security. A patient can monitor which entity is accessing the data and grant the accessibility permission to only the authorized entities accordingly. Therefore, the interoperability issues related to the HIE system are patient-centric and assessed adopting smart contracts. We also propose a DApp able to manage the user's authentication and authorization.

4.2.3 System Design

This section presents the model, which seeks to contemplate the four major issues highlighted above. The BaaS-HIE proposed model consists of five parties: i) constrained and unconstrained devices (IoMT), ii) patient (gateway/aggregator) point, iii) authorities (healthcare centre personnel (e.g.,.e doctors) and patients), iv) decentralized storage servers, and v) blockchain network.

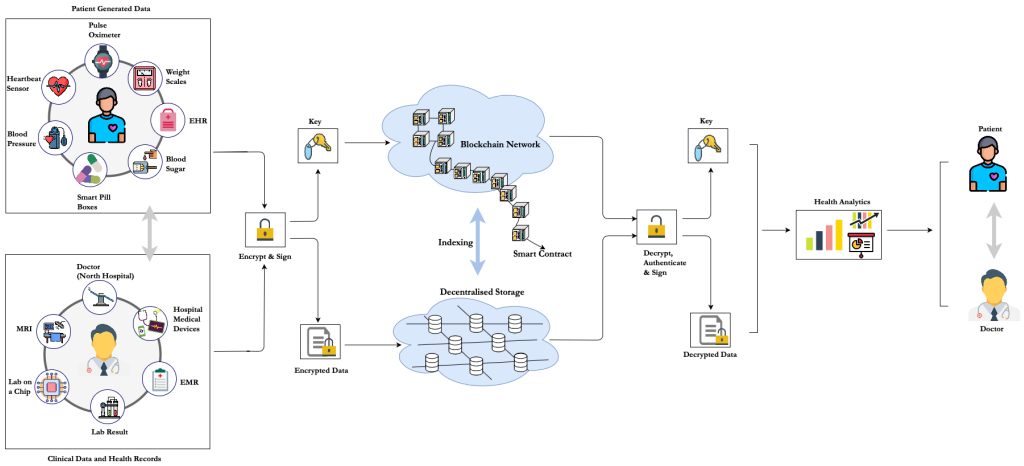


Figure 4.12: Baas-HIE proposed model.

This system uses a private and consortium-led blockchain, which means that only authorized viewers can read blocks and only designated nodes can execute smart contracts and verify new blocks. Restricting viewers to only invested parts as doctors and patients help reduce excessive information exposure by requiring authentication to access the application. We propose three scenarios for this model:

1. Patient collects data from IoMT devices to create an EHR from scratch;

2. Patient grants an authority to access an EHR;
3. Patient adds a new block to his chain due to sharing his EHR with a healthcare centre doctor;
4. Patient adds a new block to his chain due to visiting a doctor in a healthcare centre.

Therefore, according to Figure 4.12 we outline the system as follows. A patient remotely monitored by a doctor is equipped with various IoMT devices (such as a blood pressure monitor, pulse oximeter) to collect all health data from the patient. First of all, each IoMT device must be authenticated with the patient (e.g., gateway/aggregator point), therefore, it uses an Identity-Based Signature (IBS) to calculate the signature of the pseudonym and sends it to the patient. The patient verifies the signature by decrypting it using the public key. If it recovers the pseudonym, authentication is successful. Thus, the patient acts as an authority that certifies that the node has the private key corresponding to his public key. Consequently, he/she accepts data from this node.

The raw data is sent to a gateway/aggregator point, typically a smartphone or tablet, for the application's aggregation and formatting. The patient creates the EHR record from scratch (containing data collected from devices) and sends it to the relevant smart contract for full analysis along with customized threshold values. Next, the patient can also decide to share his EHR records with other doctors belonging to different healthcare centres.

The smart contract will then evaluate the data provided by adding a new block to the chain and send alerts to both patient and doctor if the measured data is outside the predetermined thresholds. The authorized authority could consult the patient's EHR through his blockchain decrypting EHR using the patient's public key when he granted the authorization. At the same time, after being treated in a healthcare centre, the doctor adds a new EMR including appropriate lab tests, diagnoses, and prescriptions, and all the information including the hash of the EMR, visit ID, patient ID, authority ID and date is encapsulated in one block. Patient treatments at different times will be generated in different blocks. A series of blocks are then generated according to the time sequence and a healthcare blockchain of this patient is constructed. Once the treatment is finished, the patient can deny further access to the doctor. The logic and state transition events are recorded as immutable data in the blockchain. Using these unique properties of blockchain, our system manages authentication, integrity, confidentiality, accountability, and health data auditing and sharing.

4.2.4 Implementation

As previously mentioned, our system is envisioned for storing and sharing healthcare data from medical institutions and individuals; it includes three main

components: i) a patient-centric user experience accessed through a browser and mobile app interface, ii) a blockchain network, iii) IPFS-based decentralized storage. Figure 4.13 provides an overview of our architecture. Each main component is described below. Firstly, we made the following implementation assumptions:

1. The constrained node is already authenticated with the patient gateway/aggregator point;
2. There exists a security policy allowing secure communications within the constrained network domain (and in particular between gateway/aggregator and the IoMT devices);
3. The gateway/aggregator is a trusted entity that is the owned system by the patient.

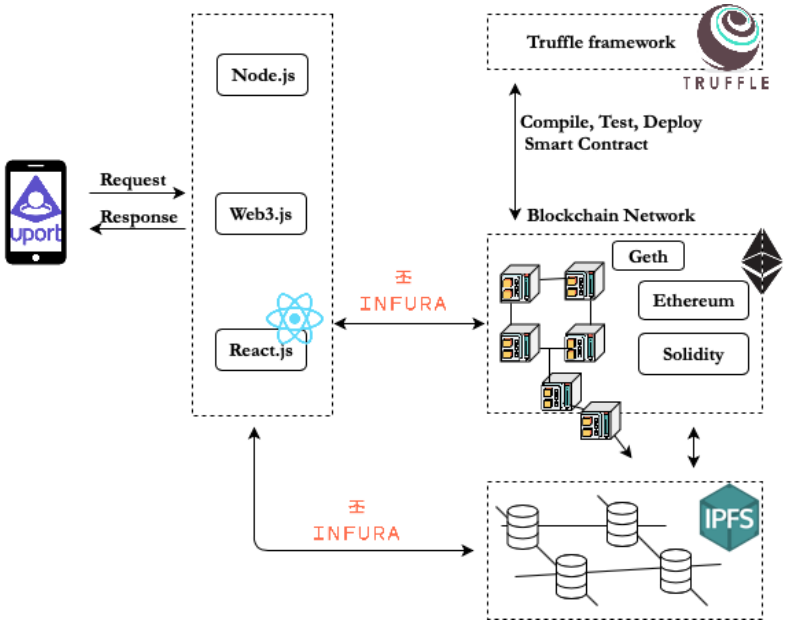


Figure 4.13: BaaS-HIE architecture.

Patient-Centric User Experience

The user interface (UI) is managed by a decentralized application (DApp) on smart devices responsible for communicating with smart contracts on the blockchain and managing user profiles. The information coming from the IoMT devices is aggregated and formatted in the back-end of the DApp and forwarded to the smart contracts, which are connected using a *web3.js* object. Patients choose whom to include in their circle of care and, at the granular level, what information is shared with each. For patients who need or want assistance from a trusted

party, the circle expands to include a caregiver who can act on behalf of the patient.

To implement the front-end of the DApp, we used React, a JavaScript library, for building user interfaces. The structure of the React application is relatively simple. The main React components are: *App.js*, which contains the login functionality and *Dashboard.js* where the user can enter and view his data. Because we adopted uPort, we used uPort's implementation of web3 for the contract interactions.

When *initAccount*() is called during the user's registration, uPort injects HTML into the page that contains a QR code. The user uses the uPort mobile application to scan the QR code. When finished, the user object is be populated with its uPort identity details that were requested.

An uPort identity is a complete digital representation of a person (or app, organization, device, or bot) that identifies him when interacting with smart contracts and other uPort identities, either on-chain or off-chain. The uPort mobile application generates a public and private key for a user and deploys smart contracts to represent their identity. This ability to make statements about themselves without relying on centralized identity providers is what makes uPort a platform for self-sovereign identity. uPort consists of a combination of Ethereum smart contracts, some of which are shared and some specifically owned by the identity owner, including:

- An identity is identified by an MNID encoded Ethereum address. MNID includes the Ethereum network ID and a checksum, making it safer to use in a world where users regularly transition between Mainnet, multiple test networks, and private chains;
- A straightforward optional proxy contract designed to live permanently with key recovery and management delegated to an upgradable Identity Manager contract.

This also issues a push notification connection between the mobile phone and the application, so each time a transaction is initiated from the app, the user will be prompted to accept the transaction on his mobile device.

Blockchain Ecosystem

We have chosen not to carry out our operations and smart contracts on the Ethereum public blockchain but on a separate private chain using Ethereum's protocol. This offers the freedom to experiment outside the set parameters of the Ethereum blockchain and eliminates the need to spend Ether. As a proof-of-concept, we implemented smart contracts using the Ethereum coding language Solidity. For implementation in Solidity, we coded our smart contracts using Remix, a website that has a compiler to test contract functionalities. Then, we

used *Truffle* - a framework for compiling, migrating and testing contracts.

Next, we set up the configuration file for deploying the contract using Ganache tools to locally emulate a complete blockchain on the top of the development host. We used *Clique* - Proof-Of-Authority (PoA) consensus protocol. We used Infura, a scalable back-end infrastructure for building DApps on the Ethereum blockchain, to connect to both blockchain and decentralized storage. Once the contract has been deployed, an account (address) is provided to record its binary code and can not be changed; only an authorized user can invoke contracts.

As well, the next function will call a transaction on the contract. Here we pass an address and *InterPlanetary File System* (IPFS) hash. The hash is passed as an argument and the address is set as the field that our contract receives as *msg.sender*. Note that the name of the $setHash()$ function is the same name we use in the Solidity contract. Next, the second function - $getContractHash()$ is used to get the IPFS hash.

By interacting with smart contracts, patients can prove their own EHRs/EMRs and manage the relationship with healthcare organizations. The functionalities are constructed by three types of smart contracts: (i) *Entity Contracts* (EC), (ii) *Relationship Contracts* (RLC), and (iii) *Data Contracts* (DC). The concept of these contracts is motivated by MedRec [167] and DeepLinQ [168].

Entity Contract (EC) We distinguish between two types of EC contracts, which are EC patients and healthcare centre ECs. A patient EC records the patient's encrypted data, including a digital uPort ID, patient name and relevant RLC addresses. A healthcare centre EC records the hospital name, its RLC address, and the government institution's relevant ID for identification. Patients can verify a hospital's identity by calling the EC's smart contract methods and vice versa. To protect themselves against malicious users, all participants are certified by the system administrator.

Relationship Contract (RLC) As well, there are two types of RLC: patient and healthcare organization RLCs. RLC records the contract addresses of medical data (MDC) it owns. Thus, a patient can share his medical records with selected healthcare organizations and a healthcare organization can send an MDC to a patient. For instance, we define two roles, patient and healthcare centre, of EC and RLC to achieve role-based access control so that transactions are more efficient and patients can better guarantee data ownership. We do not allow the sending of MDC directly between healthcare organizations. Therefore, authorized users of an MDC can access EHRs/EMRs. Any exchange of this type must pass through the data owner, the patient. This element of differentiation is crucial not only for efficiency but also because the core of privacy is based on the fact that a patient owns his EHRs. Access authorizations are managed individually for each patient's EHR and can be changed at any time via a mobile application.

Medical Data Contract (MDC) An MDC represents the data stored in the decentralized storage. We distinguish between two types of MDC: patient and healthcare organization MDC. A patient MDC includes the scratch-generated EHR containing data collected from IoMT devices; therefore, it stores metadata including patient ID, device ID, timestamp, threshold, and fingerprint of the record. Similarly, a healthcare organization MDC contains the EMR generated after the patient has visited the hospital. It stores metadata including hospital ID, division, physician name, date and time, diagnosis and relevant considerations, and the recorded fingerprint.

Therefore, there will be a patient MDC smart contract in our system, *MDC-Caller*, that the smart device will call to handle all data. Next, *MDCCaller* will create the appropriate individual contract for the specific device it is receiving data from. For example, if receiving heart rate data, the smart device will call *MDCaller.heartMonitor()*. This will call an object of the *MDCCaller* and the function *MDCCaller.heartMonitor()*. The smart device will pass the data and the specified threshold values as parameters. The function will then create a new *HeartMonitor* object and pass the same parameters to its *analyze()* function. Therefore, the patient creates the EHR from scratch containing the collected data and the result returned by the *analyze()* function, the subcontract will write this transaction on the blockchain. Before being added to the blockchain network, each EHR/EMR is cryptographed using OpenPGP. The EHR's owner's public key is used, and thus only authorized entities can access the contents of the EHR using its private key. Furthermore, if the measured data is outside the predetermined thresholds, an alert will be triggered to the patient's device and to the healthcare organization authorized to access that EHR. At the same time, if a patient wants to see a doctor belonging to a different healthcare organization (i.e., Org B), he can share his EHRs in Org A with Org B by adding Org B to the viewer list of the MDC. Subsequently, the contract will send the MDC address to the north hospital's RLC address, which can be obtained via the EC corresponding to the hospital ID.

IPFS Decentralised Storage

To maintain the system's performance and economic viability, EHR/EMRs are stored in decentralized storage, and the data URI hash is the data element of the block involved in the chain. Transactions involving data are signed with the owner's private key (patient or healthcare organization). The hash is created based on the contents of the files. If something in the file changes, a different hash is returned, effectively making the files tamper-proof. Using the URI hash and private key cryptography, we minimize each block's size and provide an additional layer of data security. Therefore, IPFS is the perfect solution for Ethereum DApps, since storage on Ethereum is expensive. Given that IPFS

data are readable by anyone, to ensure confidentiality, the EHR/EMR content is encrypted using the OpenPGP API.

4.2.5 Performance

To validate the functionality and assess the performance of our prototype, several experiments have been done. These experiments furthermore improve the understanding of the usability of the blockchain in real-world applications.

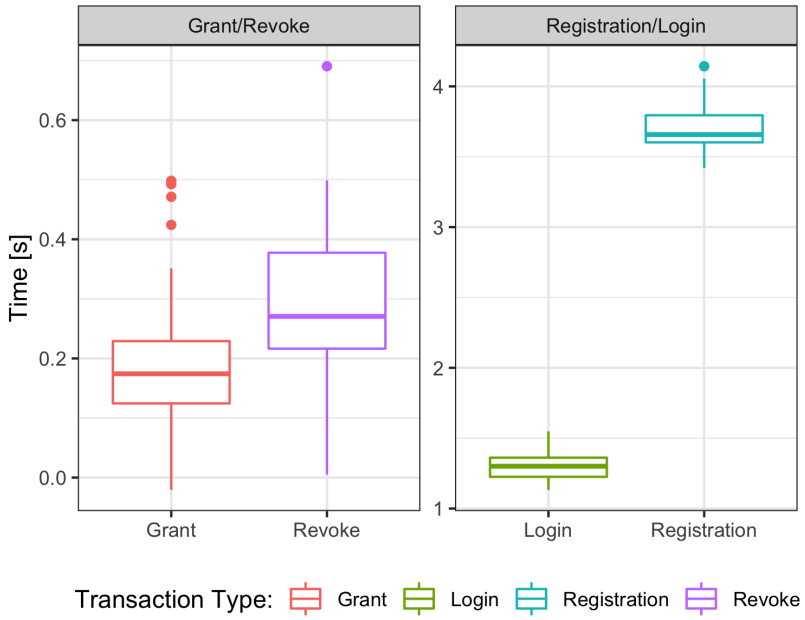


Figure 4.14: Average times [s] necessary to grant/revoke permissions and to register/login.

Experiments are performed on the server hosted locally on the local machine with processor Intel Core i7 processor running with a 2.4 GHz clock speed, 16 GB memory, and 1 TB SSD storage. Experiments performed on blockchain quantify the response time necessary to perform each of the following operations: i) registration, ii) login, iii) grant permission, iv) revoke permission, v) get data, and vi) upload data, respectively. In order to have truthful results, we performed 30 subsequent iterations for each experiment. Figure 4.14 encloses the collect response times for each experiment related to grant/revoke permissions and respectively to register/login. We can notice that the average response times necessary to revoke and grant permissions are comparable; the grant transaction takes on average 0.32 s while the revoke transaction requires on average 0.21 s. In the case of registration, it takes on average 3.7 s. Login requires only 1.3 s. In the case of registration, the data needs to be inserted into the blockchain, and the invoke function must be executed. The response times necessary to upload

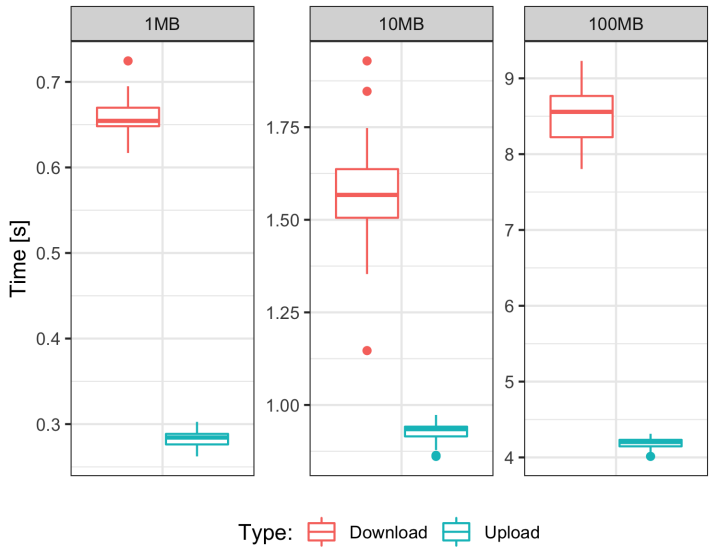


Figure 4.15: Average times [s] necessary to download/upload an EHR of 1, 10 and 100MB in size.

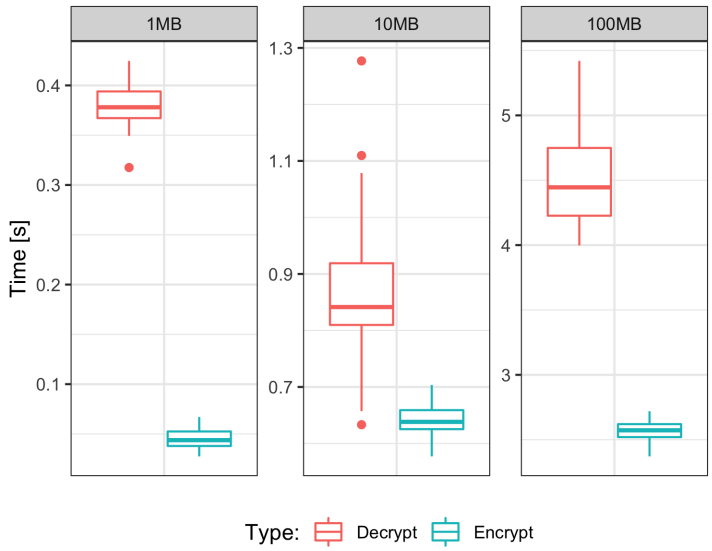


Figure 4.16: Average times [s] necessary to encrypt/decrypt an EHR of 1, 10 and 100MB in size.

an EHR of varying size from 1 MB up to 100 MB on IPFS and write its hash on the blockchain are shown in Figure 4.15. We can notice that the download time on average for each payload of a different size is greater than the upload one. In particular, for an EHR of 1 MB in size upload and download times are comparable; 0.283 s for upload and 0.66 s for download. For an EHR of 10,MB and 100 MB in

size, the average time for upload is noticeable greater than the average time for the upload; for instance, for an EHR of 10 MB in size, the upload time takes in average 0.928 s while to download it 1.567 s. Also, the download of an EHR of 100 MB in size takes in average 8.521 s, while uploading 4.187 s. The response time necessary to upload data also includes the time necessary to encrypt the data. As well as, the response time necessary to download data from the blockchain is greater than that necessary to upload the file on the blockchain because, in the case of data download, the blockchain checks whether the actor has the permission to access the specific subject and the time necessary to decrypt data (see Figure 4.16). Figure 4.16 illustrates the average times necessary to encrypt/decrypt and EHR of varying size from 1 MB up to 100 MB. We can notice that for an EHR of 1 MB and respectively 10 MB encrypt/decrypt operations are quite comparable, while for an EHR of 100 MB, the time necessary to encrypt/decrypt data takes on average a couple of seconds. Thus, this impacts the time needed to upload and download an EHR.

4.2.6 Conclusions and Future Work

This contribution vestigated the current state of the art of the healthcare systems, with a particular interest in HIE between different practitioners (e.g., healthcare organizations). Driven by the recent shift towards patient-driven interoperability, in which health data exchange is patient-mediated and patient-driven, we address the new associated security and privacy challenges, requirements, technology, incentives and governance necessary to ensure successful data sharing.

Therefore, this contribution aims to proffer blockchain technology as a suitable platform to overcome interoperability and security challenges of patient-mediated HIE of EHRs/EMRs among different practitioners. The proposed BaaS-HIE system enables HIE's interoperability and permission management in a secure, private, and auditable way by leveraging the blockchain technology's features in an IoT-EHRs/EMRs scenario. Finally, the experimental analysis carried out helped us to verify the feasibility of the system.

To conclude, future work related to this research will focus on integrating this system into a real use case. Furthermore, we are also planning to test other blockchain solutions to understand better which technology is best suited to this scenario.

Part II

The Confluence of Osmotic Computing in the Cloud-to-Thing Continuum

5 Osmotic Computing on the Rise

The widespread of the Internet of Things (IoT), e.g., mobile and sensor devices, has led to the creation of a wide range of highly distributed, federated, data-intensive, and latency-sensitive applications that are disrupting legacy-centralized computing moving cloud resources closer to users and edge. In this context, Osmotic Computing (OC) is a new paradigm that supports this by simplifying the dynamic management of microservices (called MicroELements or MELs) on resource-constrained edge/fog devices as well as cloud-hosted services in public/private, hybrid or multi-cloud, thus providing reliable IoT support with specified levels of Quality of Service (QoS). On the one hand, defining when and how microservices can be migrated from edge resources to cloud-based resources (and vice versa) and factors that influence such migration remains a challenge. Monitoring plays a central role in identifying when and where a specific microservice should be migrated. For migration to be effective, it is necessary to monitor the performance of the microservices properly. On the other, networking and security represent the enablers that allow the dynamically adjusting the overall microservices behaviour according to user/infrastructure requirements. Network management in a federated, highly dynamic, and distributed ecosystem is challenging. This chapter seeks basic concepts, methodologies and key technologies behind OC to tackle the challenges mentioned above.

5.1 Introduction and Motivation

As the Internet of Things (IoT) extends into several application domains such as healthcare, utility grids, cities, agriculture, transportation, industry 4.0, and disaster management, the need for investigating on-the-fly computation over the IoT data streams is even more critical. However, despite the rapid advances in IoT-related technologies, IoT applications' major bottleneck is the limited computing resources.

The situation is worsened by many IoT applications that need intensive use of resources for data analysis and application processing in real-time. A common way to overcome IoT devices' limited computing resources is to offload some tasks to systems with sufficient computing resources, e.g., cloud servers.

The offloading approach improves IoT applications' performance and reduces energy consumption but often introduces additional overhead. This could increase IoT applications' latency and network congestion, especially in those applications involving resource-intensive operations. These disadvantages are

the main driving factors that motivate the recent development of Edge Computing (EC). EC exploits devices with larger computational power, such as cloudlets or computing-enabled switches, at the network edge to support IoT applications. The use of EC decreases the latency and increases the bandwidth by offloading some tasks.

Today's digitalized ecosystem is federated and usually highly distributed, becoming complex and dynamic and extending through the cloud/fog/edge and IoT layers and the federated organizational boundaries. Therefore, in a federated organizational system where organizations collaborate to share resources, microservices, and exchange information, applications must be highly scalable and available and run on heterogeneous infrastructures.

In this panorama, to facilitate highly distributed and federated computing environments, a new promising paradigm to dynamically extend edge/IoT resources towards the cloud (and vice versa) is OC [169, 170]. Borrowing the main osmotic concepts from chemistry, OC assumes that each application can be slit into many dynamic MELs [171] that can be executed across different computing systems.

Hence, OC implies the dynamic management of services and microservices across cloud and fog/edge, addressing deployment, networking, and security issues, thus providing reliable IoT support with specified levels of Quality of Service (QoS).

Application delivery follows an osmotic behaviour where MELs in containers are deployed opportunistically in cloud and fog/edge systems. Because of the high heterogeneity of physical resources, the MELs' deployment task needs to adapt to virtual environments and involved hardware equipment. Thus, a bidirectional flow of adapted MELs from cloud to fog/edge (and vice versa) must be managed (see Figure 5.1). Moreover, the migration of MELs in the cloud/-fog/edge system implies the need for dynamic and efficient management of virtual network issues to avoid application breakdown or QoS degradation.

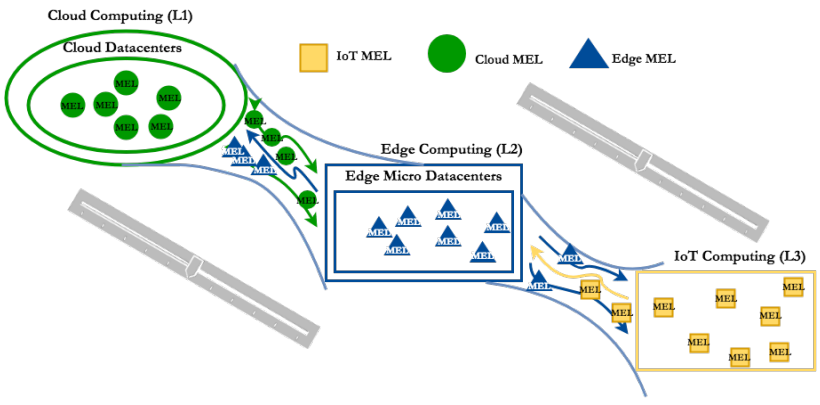


Figure 5.1: Osmotic Computing MELs offloading.

A breakthrough approach to address these issues is decoupling user data and applications from networking and security services. OC moves in this direction, providing a flexible infrastructure by offering an automatic and secure microservice deployment solution. Specifically, OC is based on an innovative application-agnostic approach, exploiting lightweight container-based virtualization technologies (such as Docker and Kubernetes) to deploy microservices in heterogeneous fog/edge and cloud.

However, using an osmotic infrastructure poses new challenges for IoT workflow application developers and operations managers as they need awareness of resource/device heterogeneity, virtualization software heterogeneity (e.g., hypervisor vs container), data analytic programming model heterogeneity (e.g., stream processing vs batch processing), geographic distribution and network performance uncertainties.

The growth of microservices as independent and distributed components of limited scope grouped in a federated architecture to "form" different applications and meet different business needs and the complex network model has introduced data protection and privacy problems. In fact, federated environments often are not manageable by a single organization. Each organization requires the ability to have proper administration and security policies to maintain data security while allowing selective sharing of resources. In this kind of environment, classical models of access control effectively protect resources while allowing users to access resources within their privileges. However, the management of such microservices is not trivial at all.

This highly dynamic and distributed ecosystem's network complexity further increases when federated systems and corresponding microservices extend not only through the cloud/fog/edge and IoT layers but also the federated organizational cross-boundaries. Consequently, this ecosystem requires suitable access control systems that enable participants to define and manage who can access their resources independently.

The rest of the chapter is organized as follow. Section 5.2 discusses the osmosis process. Osmotic MELs are presented in Section 5.3, while the *Software Defined Membrane* (SDMem) is highlighted in Section 5.4. Section 5.5 highlight the design on an OC ecosystem, while a proposed implementation in presented in Section 5.6. Section 5.7 concludes the chapter.

5.2 Osmosis Technique

In chemistry, *"osmosis"* represents the unrestrained net movement of molecules from a higher (mean solute concentration) to a lower water concentration (high solute concentration) [169]. As shown in Figure 5.2 (a), when a semi-permeable membrane separates pure liquid water and glucose, water moves from high to low water activity, as shown in Figure 5.2 (b). So, osmotic pressure is important

to be applied to a solution to stop water flow across a semi-permeable membrane, as shown in Figure 5.2 (c). It is presented in Equation (5.1) [172].

$$\pi = ic_{solute}RT \tag{5.1}$$

where π is the osmosis pressure law, i is a correction factor, c_{solute} is the molar concentration of the solution, R is the ideal gas constant, and T is the temperature in Kelvin.

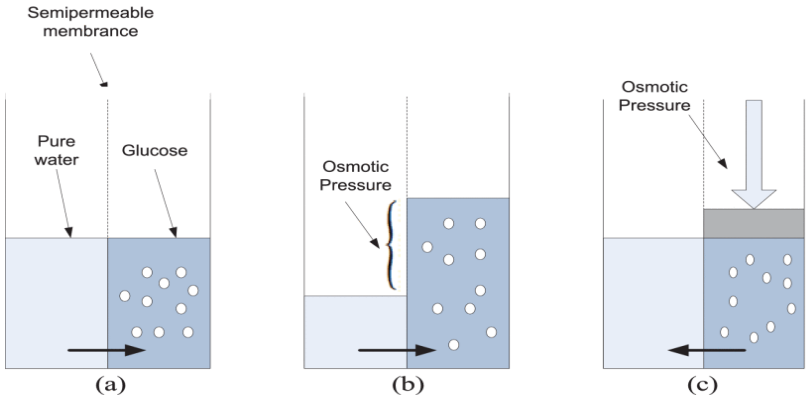
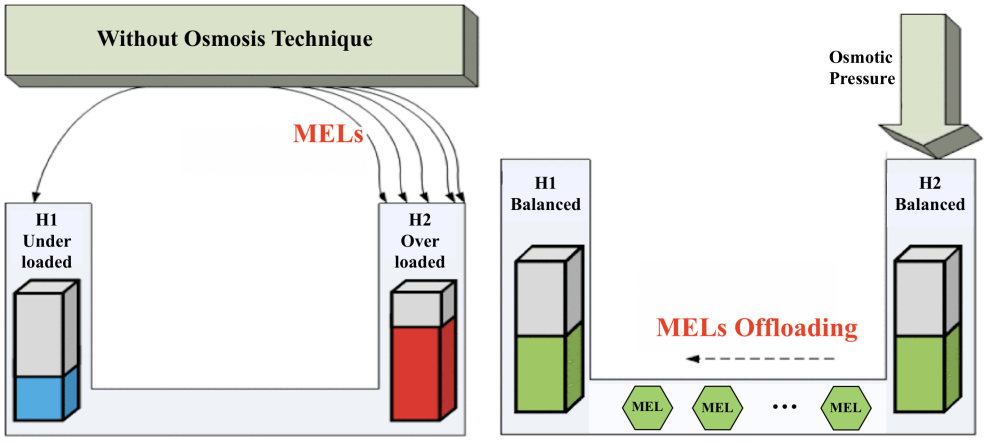


Figure 5.2: Osmosis technique and osmosis pressure. (a) Initial state (b) Equilibrium (c) External Pressure.

In osmotic environments, this process can be used to represent how MELs can be migrated across the cloud, as shown in Figure 5.3. Figure 5.3 (a) shows over and underloaded hosts as two liquids in a tube, one is pure water and the other is glucose. Figure 5.3 (b) shows how the osmosis technique affects hosts to migrate MELs between them to achieve a more balanced cloud system.



(a) Without osmotic technique **(b)** With osmotic technique

Figure 5.3: The effect of osmosis technique in load balancing.

5.3 Osmotic MicroELements

Each MEL contains a hierarchical structure and can split into two main abstracted components: MicroService (MS) and MicroData (MD). It can be noticed, the root element of the hierarchy is represented by the MEL (see Figure 5.4), whereas underneath, there are MS and MD. The leaves of the hierarchy are MUS & MOS and MUD & MOD, respectively.

Further, MOS is the MicroOperationalService (like an operating system), MUS is the MicroUserService (like a user application service), MUD is the MicroUserData (user data), and MOD is the MicroOperationalData (MS configuration). MS and MD are portable, mobile and cross-platform. In a real example, an MS is represented by a running Docker-based container. MD is a new entity wrapped in JSON. MD can be both passive and active. Passive data can be read and written on devices, while active data can be saved within databases (e.g., MongoDB) and queried.

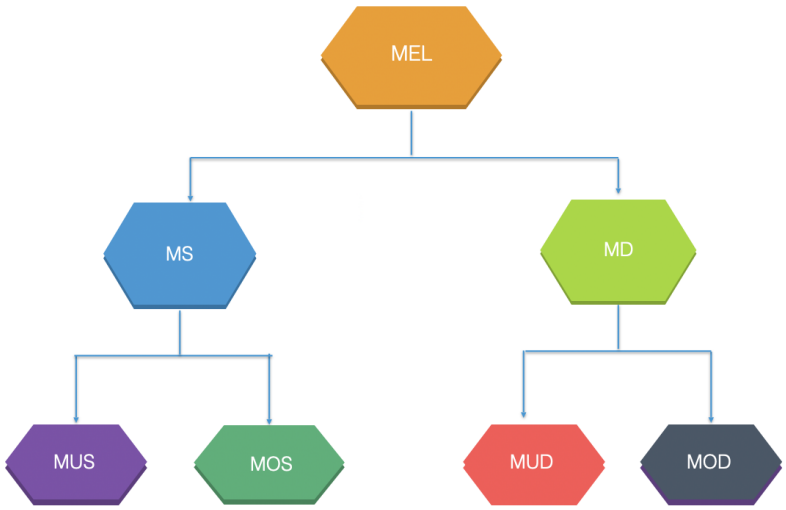


Figure 5.4: MEL hierarchical structure.

5.4 Software Defined Membrane Concept

MELs flow through an osmotic membrane, a software-defined component (that is SDMem) designed to monitor running MELs and manage their deployment and migration (see Figure 5.5).

In OC, SDMems can be set up in federated environments, with different administrative domains, and different applications and services may require different security constraints. SDMem is responsible to dynamically manage MELs osmosis by isolating independent flows according to security levels and QoS.

For example, in Figure 5.5, there are several flows of MELs between a cloud datacenter (L1) and a micro-cloud at the edge (L2). Such flows belong to three different logic applications/services characterized by specific security requirements. Thus, three independent SDMems are instantiated to isolate MELs and guarantee secure access to data and microservices. Similar considerations can be done in the osmotic management of MELs across cloud datacenters (L1) and IoT devices (L3) and micro-clouds at the edge (L2) and IoT devices (L3).

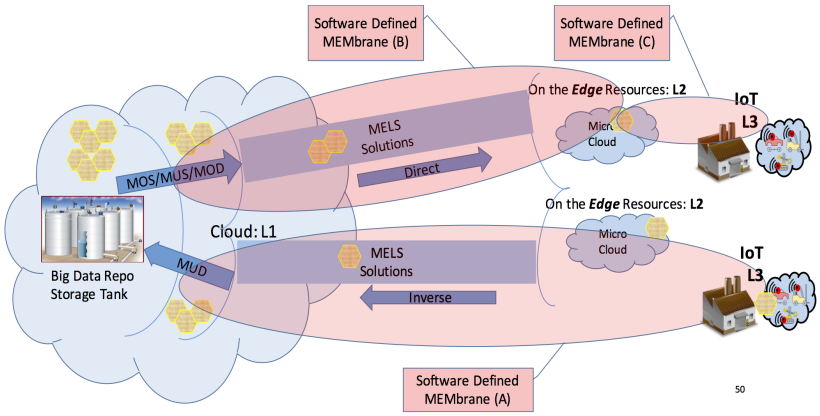


Figure 5.5: Software Defined Membrane in Osmotic Computing.

The SDMem leverages the latest technologies to be formalized and deployed, using abstractions and virtualization systems. The SDMem describes the services composition and workflow description, where its elementary elements are the MELs. SDMem mainly looks at Kubernetes for managing the multitude of containers deployed in the cloud and IoT. To specify security requirements and osmosis constraints, SDMem can be described using both YAML and JSON representations.

5.5 Osmotic Computing Ecosystem Design

Figure 5.6 illustrates today's federated ecosystem, including cloud, edge and IoT layers interconnected through the Internet. Such a scenario makes massive use of both hypervisor and container virtualization technologies to optimize virtual resources.

The Cloud Layer includes several Cloud Regions CR_i, $i = 1, ..., L$ representing cloud systems deployed in different geographical areas and owned by different organizations ORG_i, $i = 1, ..., L$. By definition, each CR_i exploits the hypervisor virtualization to manage several Virtual Machines VM_j, $j = 1, ..., M$ and in turn, each VM_j exploits the container virtualization to run several Containers C_k, $k = 1, ..., N$. In the end, each container C_k runs a specific microservice μS_k, $k = 1, ..., N$.

The Edge Layer includes several Edge Regions ER_x, $x = 1, ..., O$ representing edge systems deployed in different geographical areas. Each ER_x can manage several Edge Devices ED_y, $y = 1, ..., P$. Each ED_y, according to the type of device, can run either several Containers C_z, $z = 1, ..., Q$.

The IoT Layer includes several IoT Regions $IoTR_t$, $t = 1, ..., D$ representing IoT systems deployed in different geographical areas. Each $IoTR_t$ can manage several IoT Devices IoT_s, $s, = 1, ..., F$. Each IoT_y, according to the type of device, can run either several containers C_l, $l = 1, ..., L$, such as in the case of Raspberry PI B+ [121], each one running a specific microservice μS_z, $z = 1, ..., Q$ or directly μS_z, such as in the case of devices that do not have the hardware/software capabilities to run a container engine. Besides, several microservices μS_k and μS_z can be interconnected to arrange distributed microservices $D\mu S_l$, $l = 1, ..., R$ representing mesh-up services and applications including both central cloud, edge and IoT capabilities.

Each ORG_i, $i = 1, ..., N$ can define the microservices on the network, make the services discoverable, grant to the consumer organization, transfer access rights, and trace them. Besides, each ORG_i, $i = 1, ..., N$ needs to define their access control permissions for each distributed microservice.

Generally, access control data is implemented and stored in centralized servers and the centralized access control system manages the permissions of users to access the centralized data storage. Therefore, each ORG_i, $i = 1, ..., N$ manages its independent access control system. According to the centralized approach, all access requests go through a central authority, which decides whether grant or deny. However, this centralized approach has some drawbacks:

- *Single point of failure* - if the centralized access control fails, then no one can access the entire system;
- *Central authority* - all requests must be approved by the central authority, which can cause performance problems due to bottlenecks;
- *Limited transparency* - since there is centralized control in a collaborative environment, transparency becomes another challenging issue.

Traditional access control models like Discretionary Access Control (DAC) and Mandatory Access Control (MAC) do not adapt to distributed and heterogeneous systems. Role-based Access Control (RBAC) is more flexible and more used for access control models. According to RBAC, roles are assigned to users, and permissions are assigned to the roles [173]. Similarly, the Organization-based Access Control (OrBAC) model is an extension of RBAC that defines permissions independently from implementation and support organizations to define their security policies. Also, the OrBAC model does not fit a highly decentralized federated organizational environment [174]. Therefore, these classical access control models are not suitable and designed for heterogeneous

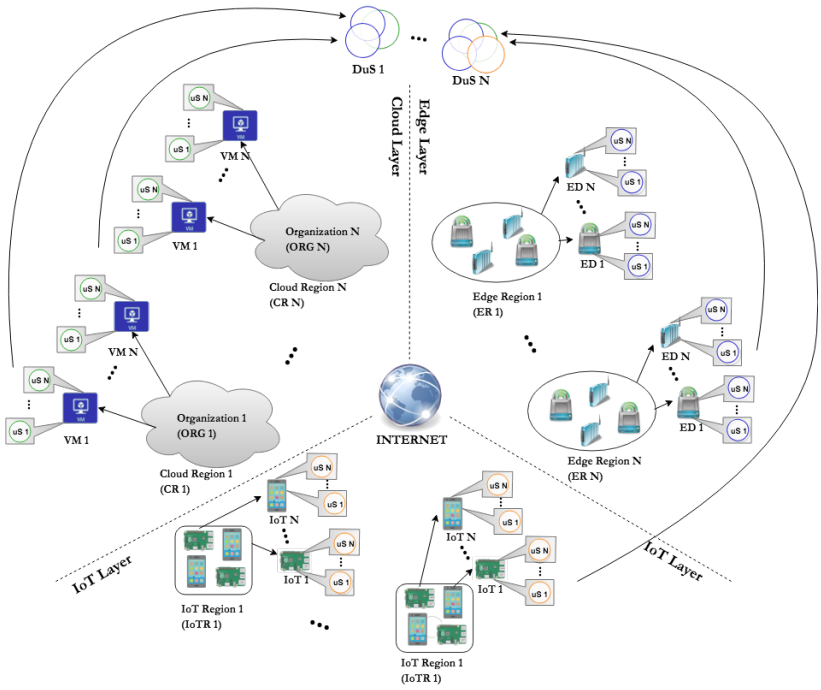


Figure 5.6: Federated ecosystem spanning on the Cloud, Edge and IoT Layers.

and distributed systems. Hence, there is a lack of suitable security models for a highly collaborative environment. Although some research has been done on decentralized access control [175], most approaches do not support fine-grained access control policies nor the transfer of access rights. Thus, building access control for the cross-boundary organization is an important and challenging issue.

5.5.1 Research Challenges

In this scenario, we have identified the following challenges that will be addressed in the following:

- *Federation Provisioning* (discovery, selection, allocation);
- *Service composability*;
- *Portability* (data portability, model abstraction and data migration to avoid vendor lock-in in federated environments);
- *Federation Networking Model* (network virtualization and connectivity, addressing, naming, multicasting);
- *Federation Security* (trust, authorization and identity management, federated policy and semantic interoperability, federation level agreement, legal issues, disaster recovery, multi-tenancy).

To facilitate provisioning in highly distributed and federated computing environments, we adopted the OC paradigm that enables the automatic deployment of MELs over the cloud, edge and IoT layers. Our methodology to solve the research questions consists of three phases:

Phase 1: MELs definition, deployment and orchestration We assume that MELs are digital resources defined as RESTful web services and that each federated organization will provide their own MEL for its digital artefacts. According to the osmotic principles, each MEL can be composed of two software components: (i) MicroService and (ii) MicroData. This division enhances the service composability concept.

A very challenging task in an osmotic ecosystem characterized by a high heterogeneity level is MELs definition, deployment and orchestration. Each federated organization must define its MELs using manifests like YAML files. The deployment of such artefacts can be automated by adopting an orchestrator tool like Kubernetes. Kubernetes is an open-source system that automates the deployment, scaling, and management of containerized applications (see Section 3.5 for further details). Each organization will set up its Kubernetes cluster to deploy and manage the defined MELs. Then, Kubernetes clusters owned by each organization will be federated to ensure federation provisioning and data portability.

Phase 2: Federation network management Network management in a federated, highly dynamic and distributed osmotic ecosystem is very challenging. The network represents an enabler that allows us to dynamically adjust the overall microservices behaviour according to user/infrastructure requirements. Kubernetes orchestrator does not provide an own networking solution; it relies on third-party network plug-ins called overlay networks. Both Software Defined Networking (SDN) and Network Function Virtualization (NFV) offer useful solutions for supporting network connectivity by enabling inter-domain and federated networks in an osmotic system. A popular SDN overlay network is Open Virtual Networking (OVN). OVN is treated in Section 3.4.

Phase 3: Federation security management In the highly dynamic and collaborative federated environment, classical models of access control effectively protect resources while allowing users to access resources within their privileges. Organizations need to have proper administration and security policies to maintain data security while allowing selective sharing of resources.

In OC, we adopt the SDMem and integrate it with blockchain facilities to tackle the security aspect. The SDMem is related to MELs flow management within the whole system, allowing organizations to manage security and privacy issues.

A blockchain is essentially a distributed and decentralized database of records

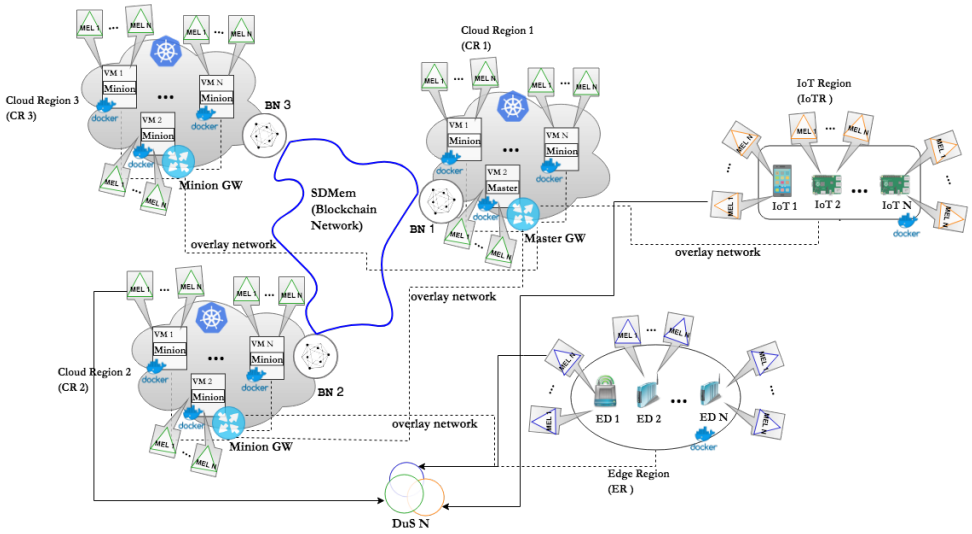


Figure 5.7: System architecture.

or public ledger containing chained blocks of transactions. Once the transaction is stored in the chain, it can be modified or deleted. Since the peer-to-peer networks do not depend on a central entity, blockchains have no single point of failure. Suppose the network nodes are sufficiently distributed across different authorities and geographical locations. In that case, the blockchain becomes resilient to network outages by natural disasters, physical or cyber-attacks, and falsifications from authorities.

In an ecosystem of federated organizations, SDMem allows each organization to enable grouping and filtering MELs based on their properties and use. SDMem allows MELs migrations according to constraints identified in the membrane guaranteeing isolation of one system from another. Therefore, security in OC involves two aspects: (i) MELs' design and specification; (ii) MELs' migration and management, subject to security and privacy policies. A private and permission-based blockchain could help us keep the shared data and historical records of the access permissions in this context. Each organization will manage its blockchain node in the blockchain consortium cluster. We propose the adoption of Hyperledger Fabric - a blockchain implementation proposed by IBM. Blockchain technology is detailed in Section 4.1.4.

Another advantage of blockchain technology is the built-in feature of defining and handling assets (resources). In the blockchain network, each organization can choose the type of resource that it wants to store, e.g., MELs definition manifests, configuration files, users' public key, and then define for each registered resource the access control list of permissions which enforces the federated policy and semantic interoperability, federation level agreement and legal issues. Using the blockchain as data storage makes this solution more robust, resilient, immutable, transparent, and traceable. With respect to the traditional database systems that

support Create, Read, Update, and Delete (CRUD) interfaces, blockchain is an append-only data storage, so it does not support update and delete actions; it supports the creation of new transactions. Tracking digital assets movements using blockchain technology enables the capability to verify asset's integrity and authenticity. Every node can have a complete and continuously updated copy of the ledger, and it allows them to be used for monitoring.

Another important aspect is authorization, identity management and multi-tenancy. Specifically, when an organization grants access rights for its microservices, the challenge is how the consumer organization can directly access the MELs without interacting with additional third party services. Each organization manages and certifies local users through its authentication and authorization server, and therefore it stores the users' public key on the blockchain nodes. In this way, organizations manage their public/private key pairs securely.

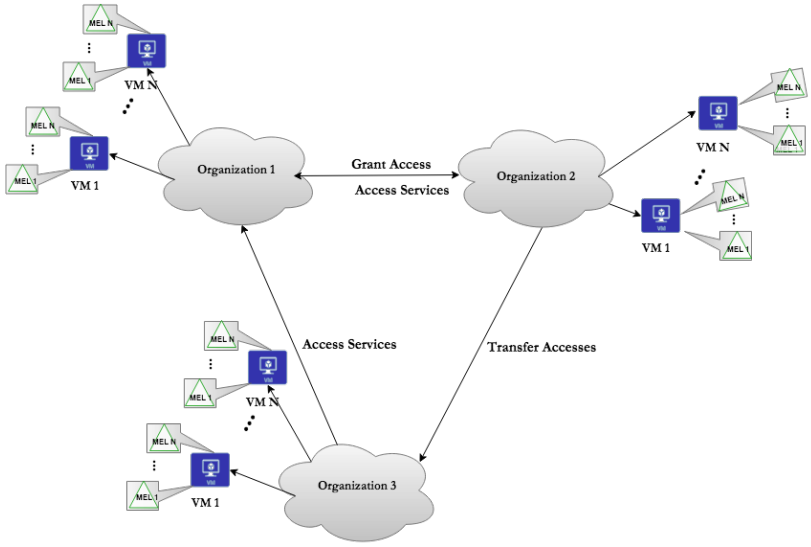


Figure 5.8: Access rights and transfer flow.

For example, we suppose that users are uniquely identified by their email address in this architecture. Organization administrators and internal access control systems have the flexibility to use their identification mechanism to identify and authorize locally. However, to achieve authentication and authorization across organization boundaries, they must have a mechanism by which all collaborative organizations agree on the Public Key Infrastructure (PKI) algorithms and definitions of it. Figure 5.8 shows how the organizations manage their users independently and exchange the public users' data on the blockchain network. Specifically, the blockchain-based SDMem allows organizations to define MELs on the network and expose them, grant access to the consumer organization, transfer access rights, and trace them, respectively. Figure 5.7 illustrates the architecture of the proposed MELs' orchestration approach involving the defini-

tion of an SDMem-based on Hyperledger Fabric (HF) technology able to satisfy the above-explained research question and to support the research methodology.

5.6 Osmotic Computing Ecosystem Implementation

The benefit of decomposing an application into different smaller microservices or MELs is that it improves modularity, making applications simpler and more resilient. However, the management of such MELs is not trivial at all.

The Osmotic Orchestrator represents the choreographic element of our osmotic system. Therefore, it encases the failover mechanism and interacts with Kubernetes to manage microservices re-/deployment (either on IoT/edge devices (e.g., Raspberry Pis) or cloud) through the manifests stored in a database system (e.g., MongoDB). In order to monitor the deployed microservices, we used Agento. Agento [176], an open-source project developed by the University of Messina, whose purpose is to monitor host and guest resources usage. Figure 5.9 shows an overview of the implemented osmotic architecture.

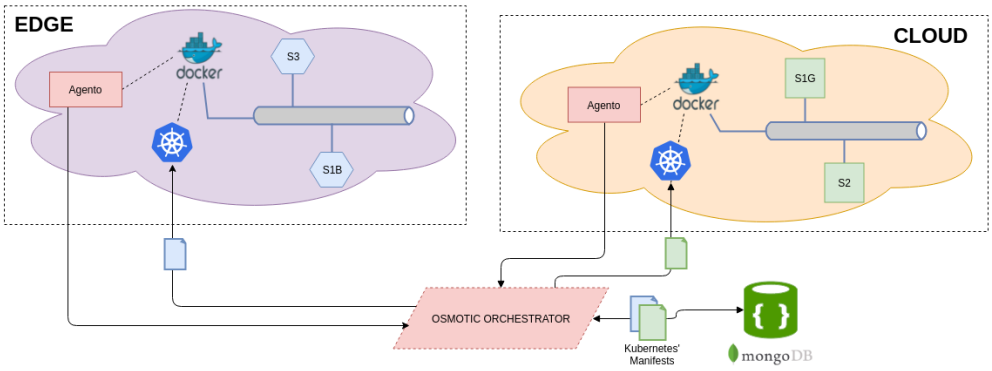


Figure 5.9: The architecture of our osmotic platform.

Kubernetes allows achieving applications and services orchestration flexibly, following a master-slave architecture. Thus, we deployed a master node on the cloud and four worker nodes: three running on IoT/edge device (i.e., Raspberry Pis) and the last one running on the cloud.

Master node The master node represents the brain of our Kubernetes cluster. It is initialized through the *kubeadm init* command. Moreover, to ensure the communication between the master and worker nodes, we used Open Virtual Networking (OVN). We chose OVN due to its efficiency and multi-platform support (further details on OVN are provided in Section 3.4). The Osmotic Orchestrator is implemented as a Custom Controller integrated within Kubernetes.

Worker nodes The worker or minion nodes are added to the cluster through the *kubeadm join* command. Of particular interest is the Kubernetes configuration on the Raspberry Pi minion node. Since kube-proxy runs in a DaemonSet, it is spread on all nodes.

However, in the kube-proxy DaemonSet, the image's architecture type used with the *kubeadm init* command is set to amd by default. Thus, the kube-proxy's image is *gcr.io/google_containers/kube-proxy-amd64*. Therefore, this causes issues when adding to the cluster nodes with different architectures, e.g., arm. To solve this issue, before running the *kubeadm join* command on the arm worker node, we must manually pull the *gcr.io/google_containers/kube-proxy-arm* image and re-tag it.

Another fundamental element of the Osmotic system is the osmotic application which has the following features: (i) highly horizontally/vertically scalable within the environment, (ii) 24 hours 24 available, (iii) fault-tolerant, and (iv) secure.

5.6.1 How to Implement an Osmotic Application

Therefore, after exposing the osmotic application's main characteristics previously, it is necessary to define a proper model to achieve these. The monitoring mechanism represents a crucial element of the osmotic workflow. Thus, we implemented it using Agento, which is deployed on each cluster node. It monitors each node (host) and the relative guests (containers or MELs) by gathering metrics related to CPU, memory usage and in and out network traffic.

To implement the scheduling mechanism, these metrics are sent to the Osmotic Orchestrator through a Publish/Subscribe bus event (e.g., Apache Kafka), which uses them to implement the corresponding orchestration policies.

The orchestration is implemented according to the following logic. When a container's resources usage metrics (CPU or memory) reach almost 90% of the allocated resources, the Osmotic Orchestrator applies a vertical scaling to increase that available container's resource by 10%. Specifically, if the host's resources allow the redeployment according to the new MEL resources requirements, the MEL will be redeployed on that node; otherwise, the MEL will be redeployed on another cluster node satisfying the new specified requirements.

For example, we suppose that the MEL is initially deployed on a Raspberry Pi device. If, at some point, the MEL's resources usage exceeds the 90% threshold, the MEL is redeployed with new increased specifications. As the Raspberry Pi has limited computational resources, if the new requested resources cannot be satisfied, the application is moved to a more powerful node like a cloud node. On the other hand, if none of the cluster nodes can satisfy the new specifications, a horizontal scaling mechanism is actuated. This mechanism is also applied vice-versa. The re/deployment of the MELs is defined through YAML files - called manifests. These are grouped in collections and stored in MongoDB.

As previously mentioned in **??**, another important feature of an osmotic ap-

plication is the 24 hours availability. However, the microservice redeployment causes its unavailability. Kubernetes currently supports two different deployment strategies: *RollingUpdate* and *Recreate*. The first one will replace each Pod with a new one in turn so as no outage will occur. This is great if we have developed our application with backwards compatibility so that new deployments can run alongside old ones. On the other hand, *Recreate* strategy destroys all existing Pods before they are replaced with a set of new ones. This is useful if we have introduced breaking changes, which could cause problems when Pods are running older and newer code simultaneously. The downside here is that there will be a brief outage between the old Pods being destroyed and the new Pods being scheduled.

Thus, to eliminate the downtime due to application redeployment, we have adopted the *blue-green deployment technique*, which occurs when we run two complete deployments of our service: a deployment in production and a deployment with the changes we would like to introduce. Thus, after we deployed the new version that meets the new requirements, we are ready to make production changes. Thus, we update the Kubernetes Service object, which plays the role of load balancer or router between the two Deployment resources, to send traffic to the new version by replacing the version label in the selector field. As shown in Listing 5.1, Listing 5.2 and Listing 5.3, we can then toggle the Label Selector on the Service to switch between blue and green, more specifically between 1.10 and 1.11 versions.

The blue-green deployment technique introduces a couple of advantages. One of them is that by creating an entirely new deployment, we can be more certain that nothing is broken before making changes to production; while the other one is that if something goes wrong, such as automated testing or undetected health checks, we can quickly and easily switch back as the previous deployment is still running.

Listing 5.1: Blue deployment - YAML manifest.

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-1.10
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: nginx
        version: "1.10"
    spec:
      containers:
        - name: nginx
          image: alexellis2/nginx-arm:latest
          resources:
          limits:
            cpu: "0.1"
```

```
                memory: "128Mi"
              ports:
              - name: http
                 containerPort: 80
---
```

Listing 5.2: Green deployment - YAML manifest.

```
---
...
metadata:
  labels:
    name: nginx
    version: "1.11"
spec:
  containers:
    - name: nginx
      image: alexellis2/nginx-arm:latest
      resources:
        limits:
          cpu: "0.5"
          memory: "256Mi"
...
---
```

Essentially, we have the A_{blue} deployment with a CPU limit of0.1 (100 milli cores), 128 Mi of memory and the service resource configured with the Label Selector on the A_{blue} service. If the CPU usage exceeds the 90% threshold, then is created the A_{green} deployment. When the A_{green} deployment Pod's state is running, we patch the Label Selector on the service to switch on green deployment.

In this context, Agento is applied to manage the blue deployment deletion. In fact, when the new (green) deployment is up, the old (blue) deployment can be deleted to free the host's resources. We need to wait for the old (blue) version to finish the requests sent to it; indeed, Agento will continue to monitor it. When no activity in terms of resources usage is registered, this one will be deleted.

The blue-green deployment mechanism is also applied to implement the osmotic application's security feature. Indeed, through the metrics reported by Agento, if anomalous behaviour is identified in terms of network traffic, the application is redeployed on a different cluster node. To do so, we used the Kubernetes node - *affinity* feature, as shown in Listing 5.4.

Listing 5.3: Service YAML manifest.

```
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  ports:
    - name: http
```

```
        port: 80
        targetPort: 80
    selector:
      name: nginx
      version: "1.10"
  ...
  ---
```

Listing 5.4: Affinity rules.

```
  ---
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
             - key: kubernetes.io/region
               operator: In
                values:
                   - region-1
                   - region-2
       preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
             matchExpressions:
                - key: additional-label-key
                  operator: In
                  values:
                     - additional-value
  ---
```

There are currently two types of node affinity, called *requiredDuringSchedulingIgnoredDuringExecution* and *preferredDuringSchedulingIgnoredDuringExecution*. The one is a *"hard"* rule, while the second is a *"soft"* one. Thus, requiredDuringSchedulingIgnoredDuringExecution would be *"only run the Pod on nodes with Intel CPUs"* and an example *preferredDuringSchedulingIgnoredDuringExecution* would be *"try to run this set of pods in availability zone XYZ, but if it's not possible, then allow some to run elsewhere"*.

Moreover, node's affinity rule highlights that the Pod can only be placed on a node with a *kubernetes.io/region* key label and the value is either *region-1* or *region-2*. In addition, among nodes that meet that criteria, nodes with an *additional-label-key* key label and *additional-value* value should be preferred.

5.6.2 Experiments

In this section, we present the experimental results obtained by evaluating our system in different usage scenarios. In particular, in the first scenario i) we conducted two kinds of analysis: Kubernetes' manifests split and store and respectively manifests' retrieval and recomposition in MongoDB. We carried out these tests in two different contexts: i) "local", where the Osmotic Orchestrator and MongoDB belong to the same Local Area Network (LAN) and ii) "remoteVPN", where the machines are interconnected through a Virtual Private

Network (VPN). In the second scenario ii), we evaluated the Pod deploy and delete time both on IoT and cloud nodes.

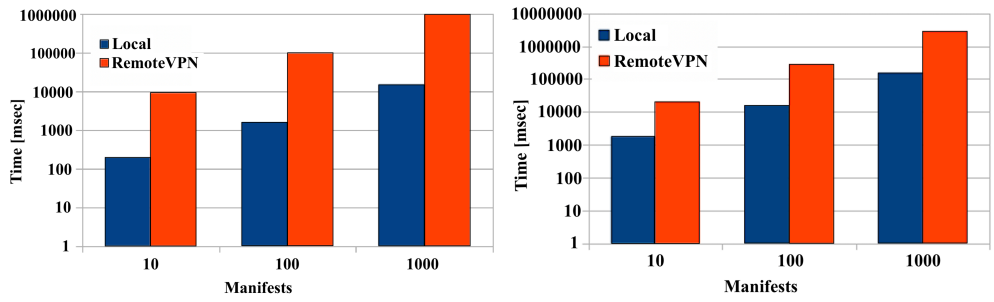
Experimental Setup

We deployed our osmotic platform on different machines. In particular, we implemented the Edge Layer using three Raspberry Pi 3 and the Cloud Layer using two blade servers with the following hardware/software configurations: CPU Intel(R) Core(TM) Xeon E7-8860V3 CPU @ 3.20 GHz, RAM 32 GB, OS: Ubuntu server 16.04 LTS 64 BIT. We adopted Kubernetes 1.9.3 version for the cluster configuration and for applications containerization - Docker 17.05.0-ce, build 89658be version. As a database system, we used MongoDB stand-alone instance running on a workstation equipped by CPU Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz, RAM 32 GB, OS: Ubuntu server 16.04 LTS 64 BIT, while for monitoring part we used Agento.

Results

In particular, to evaluate each scenario's performances, we collected 30 subsequent experiments and calculated the average time and the 95% confidence interval. We remark that this is the first concrete example of an osmotic application; for this reason, we have not found other implemented solutions to compare time performances. For the time evaluation in the (i) scenario, we considered a dataset composed of 10, 100 and 1000 Kubernetes manifests.

Figure 5.10 (a) shows the trend of the time performances obtained during the split and store phases on MongoDB. We notice that the "blue" bar is referred to the local context, while the "red" one is referred to the remoteVPN context. The computation time grows up according to the number of processed manifests in both scenarios. As we expected, the local scenario is faster than the remote one. As shown in Figure 5.10 (a), the trend is linear, and the values are acceptable.



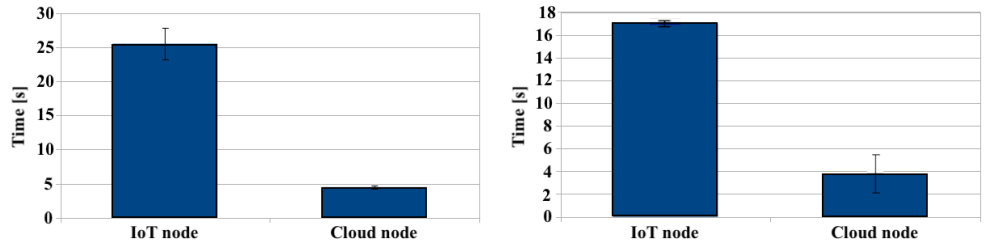
(a) Manifest split and store.

(b) Manifest recomposition.

Figure 5.10: Manifests operations.

Figure 5.10 (b) shows the trend of the time performances registered during the

recomposition phase. The behaviour is very similar to that already presented; indeed, as shown in Figure 5.10 (b), the trend is linear and also in this case the local scenario is faster than the remote one; computation time is slower than that obtained in the manifest split phase, but values are still acceptable.



(a) Service deploy time. **(b)** Service delete time.

Figure 5.11: Service operations.

To quantify the time performances necessary to deploy new Pods, we used the manifests retrieved from MongoDB. We remark that to have real-time performances, the Kubernetes' manifest complexity and data are random. Figure 5.11 (a) shows the trend of the time performances obtained during the deployment phase. Therefore, the times necessary to deploy microservices on IoT nodes are greater than that obtained on Cloud nodes. This behaviour is merely understandable; the IoT device has lower computation performances. Similar behaviour is shown in Figure 5.11 (b) which illustrates the time performance necessary to delete Pods. Service deployment and destruction present high elaboration times, especially in a real-time context, but this does not represent a critical issue for deploying an osmotic application. In fact, the Osmotic Orchestrator guarantees the failover feature by deleting the old Pods when the new ones are up and running.

5.7 Summary

The current challenges due to the IoT proliferation and the rapid development of new technologies in cloud environments led to the adoption of new paradigms able to act as integration layer among them, such as OC paradigm.

In this chapter, basic concepts, methodologies, key technologies behind OC ecosystems have been introduced. There is also presented an efficient orchestration approach to redeploy containerized microservices eliminating the application outage and promptly reacting to failures. Time performances for manifests management and microservices deployment and destruction proved the goodness of the implemented system.

6 A Gamified Flow Model Leveraging Osmotic Computing

The Internet of Medical Things (IoMT) is a sweeping revolution in the healthcare industry, with IoT quickly establishing itself as a critical part of modern healthcare. The rapid proliferation of such IoMT devices can bring limitations to the current cloud-IoT-centric infrastructures, which are not designed to handle huge volumes and velocity of data generated. To address this problem, it is necessary to revisit the network architecture, pushing some data, processing, and services directly on the network's edge nodes where the data originates, away from the centralized cloud. In this context, Osmotic Computing (OC) aims to provide a new paradigm for integrating a centralized cloud layer and edge/IoT layers. The deployment and migration strategies through the cloud and edge layers depend on the infrastructures and applications requirements. This chapter promotes the OC paradigm's basic principles and proposes a closed-loop OC flow model applied to a gamified cognitive rehabilitation use case. Moreover, the use case introduces a customized virtual reality system based on a serious game that allows the patient to carry out physical and cognitive rehabilitation therapies using a natural user interface based on Microsoft© Kinect.

6.1 Introduction

In the era of globalization and information age, healthcare industries are intensely promoting and adopting Information Communication Technologies (ICT) to improve patient care [177]. Internet of Things (IoT) has been one of the revolutionary technologies in recent times, which has catalyzed the paradigm shift in traditional healthcare methods. However, recent years have witnessed the rapid growth and adoption of the IoT paradigm, the advent of medical smart biosensors and self-monitoring systems, and research advances in big data techniques for manipulating large, multiscale, multimodal, distributed and heterogeneous data sets. These advances have generated new opportunities for personalized, precision healthcare services.

Even though the healthcare industry has been slower to adopt IoT technologies than other industries, thanks to the concept of seamless data exchange between connected devices followed by efficient data analytics, the Internet of Medical Things (IoMT) or healthcare IoT strives to cater to intelligent and personalized healthcare services. IoMT can help monitor, inform and notify not only caregivers but provide health professionals with real data to identify

problems before they become critical or to allow for earlier invention.

According to some estimates, spending on healthcare IoT solutions will reach a staggering $1 trillion by 2025 [178] and, hopefully, will set the stage for highly personalized, accessible, and on-time healthcare services for everyone. Allied Market Research's report predicts that the IoT healthcare market will reach $136.8 billion worldwide by 2021 [179]. Today, there are 3.7 million medical devices in use that are connected to and monitor various parts of the body to inform healthcare decisions.

In this context, ICT covered all these needs revolutionizing the traditional healthcare industry by proposing a new operating model to deliver services for universal access to medical transformation at a very low cost. The new operating model proposed by ICT leverages the Cloud Computing (CC) emerging technology to deliver flexible and scalable services enough to allow clients to request computing, storage and networking capacities without investing in new infrastructures. With this model, the eHealth Service Provider (HSP) will deploy and deliver healthcare services rapidly and on-demand [180].

Simultaneously, such IoMT devices' rapid proliferation can bring limitations to the current cloud-IoT centric infrastructures that are not designed to handle huge volumes and velocity of data generated. To tackle this issue, it is necessary to revisit the network architecture [181], pushing some data, processing and services directly on the edge nodes of the network where the data originates, away from the centralized cloud.

Since 2016 [182], a new promising paradigm for the integration between a centralized cloud layer and edge/IoT layers has been proposed. Osmotic Computing (OC), borrowing the name from chemistry, implies the decomposition of applications in microservices, called MicroELements (MELs), and their dynamic management through the cloud and edge data centres.

This scientific work promotes the OC paradigm's basic principles and proposes a possible OC flow applied to a real gamified cognitive rehabilitation use case. Moreover, this cognitive rehabilitation use case introduces a customized virtual reality system based on a serious game that allows the patient to carry out physical and cognitive rehabilitation therapies using a natural user interface based on Microsoft© Kinect.

The rest of the chapter is organized as follows. We survey related works in Section 6.2. The motivation is explained in Section 6.3. The design requirements are highlighted in Section 6.4 along with the real use case is presented in Section 6.5. Finally, conclusions and light to the future are discussed in Section 6.6. Section 6.5.2 shows the implementation of the proposed system. Finally, Section 6.6 concludes the chapter.

6.2 Related Works

Recent years have witnessed the rapid growth and adoption of the IoT paradigm, the advent of wearable biosensors, and research advances in big data techniques for manipulating large, multiscale, multimodal, distributed and heterogeneous data sets. Therefore, these advances have generated new opportunities for personalized precision healthcare services and the starting point for researchers to leverage the ICT innovation in the healthcare industry by proposing new smart healthcare systems (eHealth) [183].

Meanwhile, eHealth adoption introduced new challenges. In [184], the authors present the security challenges in the cloud for the eHealth domain and recently proposed solutions. In [185], the authors propose a new privacy-preserving scheme adapted to eHealth systems, satisfying all privacy requirements, as well as communication security and authentication. In [186], the authors identify the limitations in the current eHealth organization cybersecurity solutions, especially the network layer and propose a next-generation cybersecurity solution for eHealth organizations.

In [187] addresses all these important aspects of novel IoT technologies for smart healthcare-wearable sensors, body area sensors, advanced pervasive healthcare systems, and Big Data analytics. It identifies new perspectives and highlights compelling research issues and challenges such as scalability, interoperability, device-network-human interfaces, and security, with various case studies.

The fast proliferation of such IoMT provides enormous opportunities in terms of new approaches to innovate the ways that healthcare services are being provided to the patients and carrying out personalized therapies [188]. On the other hand, recent advances in computer gaming technology, digital games design, computer game engines stimulated active research in gamification of processes and activities in "serious" areas such as simulation of complex physical objects and phenomena, engineering, project management, healthcare, chemistry, planning, marketing, and other areas. Focusing on the field of health, the use of serious games can provide an additional mean to increase interest in training, education and evaluation of user performance. For instance, serious games can be designed to educate and train health care professionals to avoid medical errors or in rehabilitation processes to reproduce the repetitive tasks that have to be done by the patient.

Many studies have been conducted to verify the feasibility of serious games by leveraging Kinect-based systems. The playing experience and motivation, various high-quality graphic implementations can be found, mostly using Unity 3D, in one case Blender [189]. *"Kinect-o-Therapy"* [190] is a complex environment consisting of four mini-games, which are gamified versions of the generally prescribed exercises for arms and body. In [191], the authors presents a suite

of serious Kinect™ - based games for rehabilitation. In [192], presents a serious game-based system for dance notation training named LabanDance. The system tracks the full-body motion via the Microsoft Kinect of the user while performing the movement according to the dancing score. This article presents a serious game framework developed using the Unity 3D game engine and Kinect V2 sensor as a natural user interface [193].

In summary, there is a substantial amount of research done in the area of IoT and the area of eHealth, and it is expected that the integration between these two areas gives a major boost to the development of new systems that help to care for human health.

6.3 Motivations

The Internet of Medical Things (IoMT) is a sweeping revolution in the healthcare industry, with IoT quickly establishing itself as a critical part of modern healthcare. As IoMT expands, the need to investigate on-the-fly computation over the IoMT data streams is ever more pressing. The importance of data in delivering efficient and effective healthcare has long been obvious and has never been greater. Indeed, most IoMT applications are modeled as data transformation flows which includes: i) multiple interdependent, heterogeneous data analysis computational and programming models that perform various data tasks of data transformation from data ingestion to analysis, ii) virtualized/non-virtualized computational and network infrastructure, iii) means of communication of various types. Ultra-high reliability and very low latency communication are essential for the health care IoMT. The networks provide the framework for such communication and allow the carrying out of activities such as surgical visits and remote surgery. Any communication problem can result in a potentially catastrophic outcome for the patient.

The classical approach for the realization of this type of IoT application workflow is based solely on the Cloud Computing (CC) resources, which may be distant from the data sources, thereby leading to an excessive delay of detection of events (i.e traffic congestion). The emerging availability of different types and complex IoT devices, along with the large volumes of data that they produce, can reveal bottlenecks for the current Cloud-centric IoT infrastructure. The transfer of large streams of information in a reliable and fast way to centralized Clouds is a limitation of current systems [194]. A possible solution to augment the scalability of CCs/ECs lies in taking advantage of the ever-increasing computational and storage capabilities available at the network edge.

For example, the simple analysis of the data produced: the aggregation of the data must be mapped to the nearby Edge Computing (EC) resource, while the resource-intensive analytic task must be mapped to the CC resource, since it must execute a complex computational model of the flow. Currently, CC and EC

provide data processing and storage resources for IoT flows, but suffer from limited bandwidth and network latency, and do not support either latency-sensitive applications or applications that rely heavily on streaming data from IoMT data sources for real-time information intelligence (in the form of data ingestion and data analysis).

To facilitate the deployment of microservices in environments highly heterogeneous and distributed, we adopt the Osmotic Computing (OC) paradigm that enables the automatic deployment of microservices, called MicroElements (MELs), over inter-connected EC and CC infrastructures. In this way, we are able to maximize the resources usage of the underlying host's infrastructure and at the same time to adapt the system availability accordingly to the user requests, ensuring therefore the main requirements in terms of scalability, ease management, fault-tolerance, CI/CD, flexibility and interoperability.

6.4 Design Goals of the Osmotic Flow

Therefore, a modern architecture should be robust, reliable and easy to be managed. We identify the key requirements that drive the design of the osmotic flow.

Scalability and elasticity Because of the enormous amount of data that will be processed in real-time, scalability is a key design requirement for the IoMT flow applications. The osmotic flow should consider scalability and elasticity by design so that MELs can automatically grow and shrink based on data volume and velocity.

Flexibility The architecture must allow for different business requirements and existing assets to integrate and collaborate.

Fault-tolerance The system must be able to manage failures in order to avoid services downtimes.

Data management The on-going flow of data coming from the IoMT nodes must be processed accordingly to limit latencies and overheads. Thus, the osmotic flow should allow an easy-to-implement interface where data transformations should be easily defined, and at the same time, each transformation should be mapped without problems on EC or CC based on performance needs. As a result, the deployment process is performed transparently by the underlying runtime engine. However, application providers can customize the behaviour of the framework to target their specific performance needs better.

Efficient composition of data transformations The osmotic flow should support the composition of transformations and cross-flow data links to create complex flows easily. To this end, the osmotic flow should consider the possibility

to compose data streams from multiple devices and public IoMT applications, thereby promoting the principle of sharing and reusability. Our osmotic flow should enable the supplier to easily define new flows, which extract high-value information from the raw data without worrying about low-level problems related to their runtime execution, such as allocation of resources and distribution of flows, elasticity and governance.

Interoperability The modern healthcare systems span the cyber and physical world, involving many IoMT devices. Therefore, it is necessary to ensure interoperability between such devices.

Network awarenes The emerging IoMT environment calls for strong network awareness. The osmotic flow should minimize communication delays while performing the deployment of data transformation tasks to CC and/or EC [181].

Cost The HW/SW infrastructure and management costs must be minimized.

6.5 Use Case: A Closed-Loop Gamified Cognitive Rehabilitation Flow Model

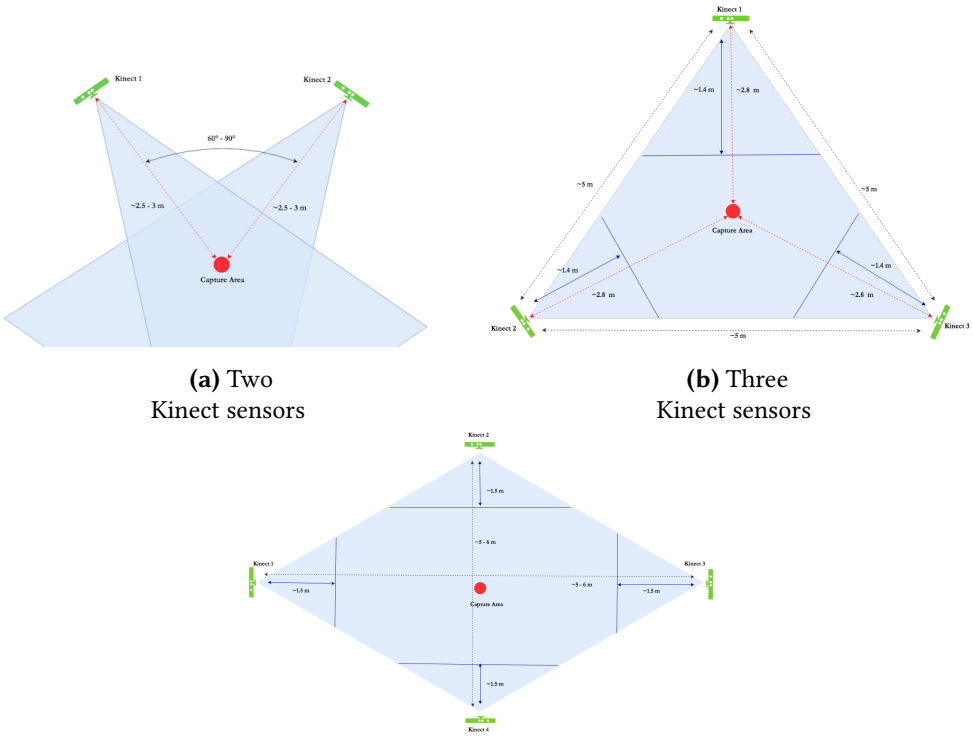
6.5.1 General Description

As a multidisciplinary research field, Serious Games have evolved substantially in recent years, solving problems in many different areas: military, education, rehabilitation and healthcare. Although there is no single accepted definition of the term, it was a consensus among the scientific community that Serious Games refer to games with a specific purpose beyond pure entertainment. In particular, in the field of cognitive rehabilitation, Serious Games are used as computer games whose main purpose is to achieve a specific goal (rehabilitate) other than entertainment, and which use the entertainment component and the game skills ability to hold the patient's attention and interest during the game.

Several studies on rehabilitation show that the effect is greater when patients follow intensive training programs aimed at achieving a goal and divided into specific tasks that need to be performed repeatedly [195, 196]. This type of game has become popular in recent years thanks to the appearance of consoles like Nintendo Wii, Playstation or Xbox, which use gestural interaction interfaces. Likewise, these technologies have become extremely useful rehabilitation tools and are expected to reduce costs in social and health settings.

Accurate motion capture plays an important role in sports analysis in the medical field and virtual reality. Current methods of acquisition of movement often suffer from occlusions, limiting the accuracy of their pose estimation.

Considering a cognitive rehabilitation scenario, to address this limitation, we



(a) Two Kinect sensors

(b) Three Kinect sensors

(c) Four Kinect sensors

Figure 6.1: Kinect arrangements.

propose a virtual reality system that leverages two-Kinect sensor connected each one to a Raspberry Pi Model B+ (peer node) and configured according to Figure 6.1 (a); the proposed system is illustrated in Figure 6.2. According to the OC principles, there are several distributed microelements (DMELs), microservices (MS) or microdata (MD) that can be off-loaded from the IoT to the edge or back to the cloud. To measure the human body's pose parameters accurately, different from previous monocular depth camera systems, the two-Kinect sensors allow us to acquire more information about human movements, ensuring that we can still get an accurate estimation even when significant occlusion occurs.

There could be different types of Serious Games through which the patient can train or rehabilitate several aspects such as strength, aerobic or cognitive capacities. The system has been modelled so that the physical presence of a therapist is not required during the session of the session and there is no need to wear any marker or sensor. Whenever a patient begins his session, a particular game that was previously configured by the therapist who defined certain aspects of the game such as speed, the angle that a particular limb must perform, or the game's difficulty level are based on the patient's condition.

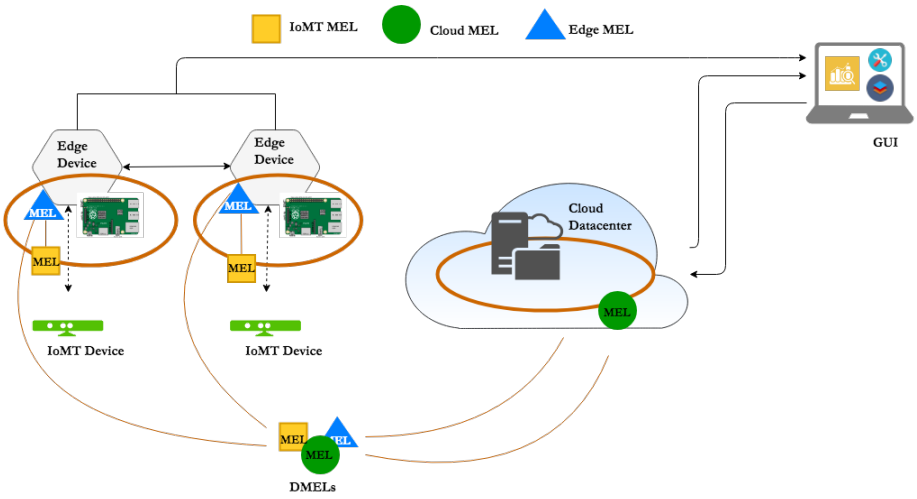


Figure 6.2: Virtual reality two-Kinect-based system for cognitive rehabilitation.

6.5.2 Closed-Loop Osmotic Flow Implementation

The proposed closed-loop flow model use case follows two guidelines, as shown in Figure 6.3. The first one relates to the development guideline, whereas the second one concerns the evaluation guideline.

The development flow includes selecting three-dimensional (3D) computer graphics technologies and tools, the modelling of physical aspects, the design of rehabilitation scenarios, and the implementation of the proposed scenario. This flow aims to design fun but useful game scenarios to motivate end users to perform functional rehabilitation tasks.

The evaluation guideline is implemented with a Distributed Node-RED (DNR); on each peer node is deployed a MS with a Node-RED instance. As illustrated in Figure 6.4, the evaluation guideline is managed with four Node-RED flows. Therefore:

F1: Configuration and Metrics Definition Through a Graphical User Interface (GUI), therapists register patients, configure the serious games' parameters according to the patient requirements and define the evaluation metrics. All the specified information are saved in a MongoDB database instance MS running in the cloud. The first patient' evaluation requires the therapist interaction to register it and configure the parameters. Moreover, in the next sessions of the same patient, all the different exercises' parameters can be configured without the physical presence of a therapist. This flow concludes with the deployment of the necessary MELs on each peer node. On each peer node is deployed a KinectRTC [197] MS which allows communicating with each other to control the streaming of different data types and manages the connections (e.g. activation or deactivation of data streams, adjusting bit rates, etc.). KinectRTC takes

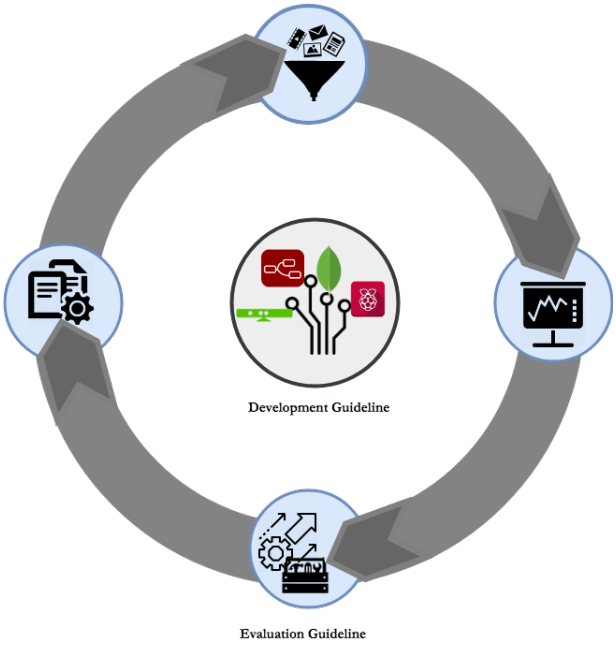


Figure 6.3: Closed-loop osmotic flow model.

advantage of the configuration functionality that WebRTC offers to manage media streams and adapt the quality of the RGB image and the audio to the available bandwidth. This means that if necessary the video resolution and the audio bit rate are automatically reduced to improve data transfer. Moreover, we deploy the Serious Game MS and then the necessary components to store all the patient's data.

F2: Patient Data Stream Collection It manages patient's data collection. All the data (e.g., files, video) are stored in different MDs containers. According to the osmosis principles, after a certain time, if the peer's node resources are not sufficient to host the Serious Game MS or MD container, they are migrated in the cloud, and vice-versa, if the Serious Game MS or MD do not need the Cloud resources and can be hosted by the peer they are migrated on it.

F3: Output Results Stream It manages the specified evaluation metrics processing. The collected data with the F2 flow, at this point, can be processed locally on the peer to evaluate the metrics. The metrics evaluation is done by different MSs accordingly to each specified metrics. Also, in this case, according to the OC principles, if the peer's resources are not necessary to make the computation, the MS is migrated and executed in the Cloud and vice-versa as previously explained.

F4: Feedback Analysis and System Improvement The improved game is reevaluated in a closed-loop technique and the corresponding parameters are

updated in the MongoDB database collection to enhance successive patient's experience and evaluation.

The reports of each session can also be read offline, therefore, the therapist will always know if a user has performed the session in a good way and act accordingly, modifying whatever he deems necessary in the patient's therapy and finally to assign an evaluation score. This user-centred game design approach allows different users (e.g., patients and medical experts) to participate actively in the design and evaluation stages. Embracing the DevOps philosophy, any change in the F1 flow produces the automatic redeployment of all dependent MELs according to the Continous Integration/Continuous Deployment (CI/CD) pipelines.

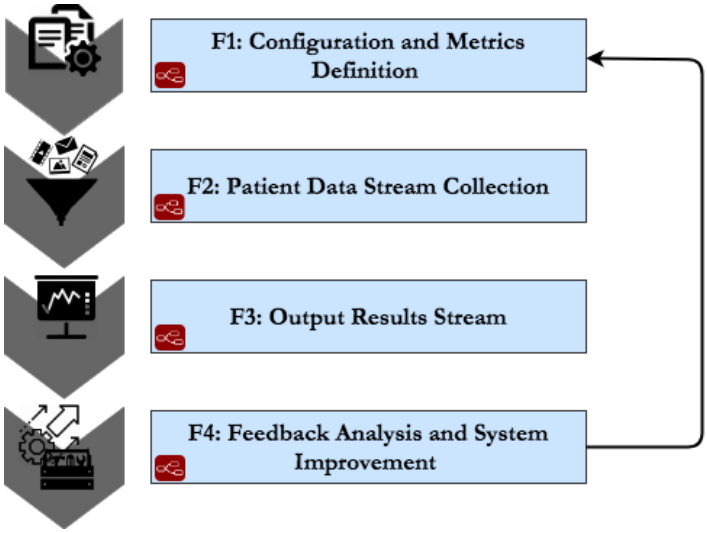


Figure 6.4: Closed-loop osmotic flow: evaluation guideline

6.6 Conclusions and Future Work

To summarize, driven by the IoMT proliferation and the associated data streams in the healthcare sector that introduces limitations for the current cloud-IoT centric infrastructures, which are not designed to handle huge volumes and velocity of data, we investigated the OC basic principles to propose an OC flow applied to a real gamified cognitive rehabilitation use case. Moreover, this cognitive rehabilitation use case introduces a customized virtual reality system based on a serious game that allows the patient to carry out physical and cognitive rehabilitation therapies using a natural user interface based on Microsoft© Kinect.

Given the OC flow and the proposed virtual reality cognitive rehabilitation system described in this contribution, in our on-going work, we aim to evaluate

the system's performances and involve Artificial Intelligence (AI) techniques to evaluate the rehabilitation session of each patient. This also implies the development of an own evaluation system.

7 Function-as-a-Service (FaaS)-based Osmotic Flow

Nowadays, the rapid development of emerging technologies such as cloud, fog, edge and Internet of Things (IoT) technologies has accelerated the advancement trends forcing applications and information systems (IS) to evolve. In this hybrid and distributed ecosystem, the management of service heterogeneity is complex, as well as the service provisioning according to classification and allocation of appropriate computational resources remains a challenge. A potential solution to these issues is Osmotic Computing (OC) - a recently introduced paradigm that allows the service migrations ensuring better resource utilization within Cloud, Fog and Edge Computing environments. Driven by the needs of complex management mitigation, greater agility, flexibility and scalability, this paper aims to propose an innovative OC ecosystem leveraging Function-as-a-Service (FaaS); there is also introduced the concept of hybrid architectural style combining both microservices and Serverless architecture. Furthermore, to support the FaaS-based OC ecosystem, an osmotic flow model for video surveillance in smart cities is presented. To validate the functionality and assess the performance and further improve the understanding of the OC flow's usability in real-world applications, several experiments have been carried out.

7.1 Introduction

In the era of globalization and information technology, the rapid development of Cloud Computing (CC), Fog Computing (FC), Edge Computing (EC) and Internet of Things (IoT) technologies has accelerated the advancement trends forcing applications and information systems (IS) for smart environments to evolve. In this hybrid ecosystem, the management of service heterogeneity is complex, as well as services provisioning according to the classification and allocation of appropriate computational resources remains a challenge. Osmotic Computing (OC) [14] is a new computing paradigm that has been introduced to address these issues. It can manage heterogeneous computing infrastructures and processing devices transparently, orchestrating the deployment of MicroElements (MELs) among cloud, edge, and IoT nodes according to IoT applications' specific requirements and physical/virtual resources availability.

Usually, OC is implemented by using microservice-oriented architectures, which structure each application as a collection of loosely coupled fine-grained MicroELements. This approach is not always suitable for deploying applications

in a complex osmotic ecosystem that must extend between cloud, fog, edge and IoT because it requires an enormous effort for careful planning and management. Recently, a serverless and lightweight technology, also known as Function-as-a-Service (FaaS), appeared as a disruptive alternative that organizes applications as a set of stateless functions and delegates the management of the execution environment to the infrastructure provider thus simplifying the development, management and execution of such functions.

IoT applications and related services can be complex systems that need statefully interconnected and cooperating MELs. Thus, the FaaS approach can not always be applied due to its stateless nature, and microservice-based approaches are necessary. FaaS can also introduce significant overhead management and complexity reduction when several features can be provided as functions. Thus, according to different applications and developers requirements, a hybrid architectural approach that combines microservices and FaaS technologies must be adopted in an OC ecosystem.

This chapter proposes the deployment of OC flows over heterogeneous systems using FaaS. In particular, we modelled a video processing application as an osmotic flow of MELs. This kind of applications is usually adopted in video surveillance scenarios.

The flow management benefits from mainstream edge, private and public CC infrastructures, and MELs rely on a set of FaaS implemented leveraging OpenFaaS. OpenFaaS is a framework for building serverless functions executed within Docker containers, and their orchestration is performed through Docker Swarm from the edge up to private and public clouds. The proposed approach is capable of efficiently distributing and allocating MELs by following the principle of OC. In our evaluation of the implemented application, we analyze the execution time for different implemented FaaS according to different deployment strategies across heterogeneous edge and cloud systems.

The remainder of the chapter is organized as follows. Section 7.2 analyze the current state of the art. In Section 7.3 we discuss our motivations behind the integration of FaaS in OC. The osmotic flow model design is presented in Section 7.4. Experimental analysis with reference to a video surveillance use case is discussed in Section 7.5. Finally, our conclusions and highlight for future works are provided in Section 7.6.

7.2 Related Works

In literature, several research works make use of microservice architecture for OC.

In [198], the authors investigate OC and proposes an osmotic flow model to manage microservices-based MELs across Cloud and Edge datacenters. In [199], a Message Oriented Middleware (MOM) to facilitate device-to-device commu-

nication in IoT environments and also to integrate complex edge computing applications that rely on message brokers, such as distributed real-time data analytics applications, is proposed. [200] proposes a pervasive trust management framework for POSNs which is capable of generating high trust value between the users with a lower cost of monitoring using Flexible Mixture Model (FMM) to develop the system and leveraging the concept of OC to perform computational offloading using 3 different solutions: i) fitness-based movement, ii) probabilistic movement, and iii) threshold-based movement. In [201], the authors investigated the feasibility to apply OC in healthcare by designing a closed-loop flow model for a gamified cognitive rehabilitation use case. Moreover, the use case introduces a customized virtual reality system based on a serious game executed by a microservices-based MEL to allow patients to carry out physical and cognitive rehabilitation. The osmotic flow follows two guidelines. The first one relates to the development guideline, whereas the second one concerns the evaluation guideline. In the same context, in [197], the authors discussed the implementation of OC as a distributed multi-agent system where microservices-based MELs and their management workflow use an Executor - to run independent microservices and a Monitor component to manages their dynamic migration.

We differentiate our work from the ones in the literature because we investigate an event-driven OC where osmotic flows leverage FaaS capabilities. This contribution also introduces for the first time an OC ecosystem that uses a hybrid architectural approach combining microservices and serverless FaaS architectures to deploy and manage MELs.

7.3 Motivations

OC is derived from the term *"osmosis"* which refers to the equalization of a solution concentration by allowing the solvent to move through a semipermeable membrane [202]. A similar analogy can be applied to modern-day computing infrastructure by separating the services to acquire processors to balance the load and resource utilization without any redundancy. OC is extensively explained in Chapter 5.

In OC, a microservice oriented solution allows deploying MELs over different computation nodes, thus optimizing available resources. Virtualization technologies, such as containerization, increase the flexibility in using heterogeneous devices and infrastructures even characterized by different constraints, such as in the case of IoT, edge and cloud nodes. Nevertheless, splitting an application into independent microservices is a complex task that requires careful planning, massive programming effort and skills.

Due to this distributed approach in managing MELs, one of the major issues in microservice-based OC is represented by the network configuration among MELs cooperating for the same application. To address the complexity of net-

work configuation [203], network softwarization approaches can be used to enable the creation of virtual overlay networks that meet the requirements of applications. However, this introduces much more complexity and overhead for developers. Another limitation of microservice architecture is that often, some microservices overload edge nodes and therefore degrade the performances.

Recently, Serverless Architectures, also known as Functions-as-a-Service (FaaS), appeared as a disruptive alternative typically referring to a software architecture where the application is decomposed into *"triggers"* and *"actions"* (or functions), that delegates the management of the execution environment of an application (in the form of stateless functions) to the infrastructure provider [204]. Functionalities offered by FaaS are more general-purpose in nature because they aim to implement separate serverless functions on demand. Consequently, a FaaS platform provides seamless hosting and execution of functions using provider-managed containers that execute functions (MS) without pre-allocating computing capability or dealing with scalability and load-balancing burden. Functions are more granular than microservice architecture. However, serverless functions can also act as scheduled jobs, event handlers, and not just as application-oriented services.

An OC platform must orchestrate and deploy into IoT, Cloud and Edge infrastructures different MELs; some are strictly related to the specific application purpose, others are necessary for the management of the OC ecosystem (e.g., network monitoring and management, security management). Since such MELs can be strongly interdependent, a FaaS approach to implement MELs is not always feasible. For this reason, we aim to investigate a hybrid architectural approach for OC that combines microservices and FaaS. Where boot time counts, where many users access even indirectly, where applications are complex, where flexibility is needed, and where cost is mounting, microservices can replace those spots to increase performance and decrease costs. As well, where event-driven is required, where need the speed of development, automatic scaling and significantly lowered runtime costs the Serverless architecture should be adopted [205].

To summarize, the integration of FaaS in OC brings the following advantages:

- *Elasticity and scalability:* rather than scaling the entire service, MEL functions can be automatically and independently scaled with usage; there is the absence of scaling management according to the requests, it is managed entirely by the service provider and potentially optimized, reducing the execution time, and then the costs;
- *Built in availability and fault tolerance*; the system must manage failures to avoid the services downtimes;
- *Productivity and ease of deployment:* the code is "packaged" and loaded, ready to be executed;

- *Provisioning:* decrease in the application complexity; fewer infrastructure components (e.g., servers, load balancers) to manage;
- *Reusability:* possibility to reuse components already built for different applications;
- *Costs:* the user will pay only for the resources he has used; as well as reduced operational costs;
- *Decreased time to market*: Create new apps in hours and days instead of weeks and months.

Even if OC's primary goal is to bridge the gap between the edge cloud and the public cloud, it can also be demonstrated by adopting the private cloud. The latter is mainly grounded to provide a "cloudlet" closed to the users that requires an efficient way to decide the procedures for the service's execution. Moreover, private cloud/cloudlet enhance resource-constrained edge nodes (e.g., computation offloading and data staging) or provide an execution environment for cloud-centric IoT applications.

7.4 Osmotic Flow Model Design

Let us consider a contemporary smart IoT application scenario composed of different private and public cloud, edge nodes, where several IoT devices are disseminated all over the urban environment as shown in Figure 7.1. To manage such IoT devices, several public and private cloud, edge and IoT MELs have been distributed across each node.

In such a heterogeneous and highly distributed scenario, each IoT device must be managed independently; this is due to the multitude and diversity of such devices (e.g., sensors and gateways), requiring different hardware and software capabilities. According to the OC principles, the management must be done following different QoS policies, as explained in Section 7.3.

Therefore, according to different QoS, MELs can be migrated from the public cloud to the private cloud and up to the edge nodes and vice versa, where each policy is managed through a pipeline. Given the complexity of the OC ecosystem, a very challenging task is, therefore, managing the pipeline.

To this aim, we present an innovative osmotic flow model able to holistically manage deployment, execution and migration of MELs according to different pipelines (e.g., violet, grey, orange). Thus, each osmotic flow will be independent by each other, scalable and flexible. To efficiently implement the osmotic flow, we leverage FaaS capabilities. In detail, the osmotic flow is implemented using functions acting as triggers, while pipelines are implemented using functions acting as event handlers. MELs are just functions acting as generic IoT services.

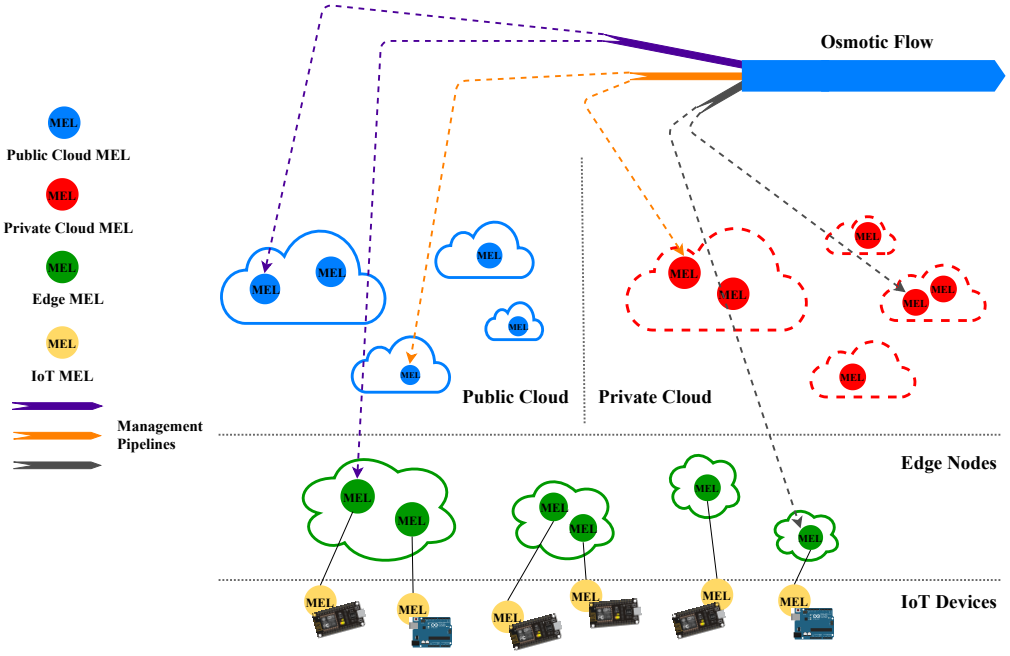


Figure 7.1: Reference Osmotic Computing scenario.

7.5 Experimentation and Evaluation

To validate the functionality and assess our OC flow's performance based on FaaS, several experiments have been carried out. These experiments furthermore improve the understanding of the usability of the OC flow in real-world applications.

7.5.1 Use Case Definition: Face Recognition in a Video Surveillance Application

Over the past decades, digital video surveillance (DVS) in smart city environments has attracted more and more researchers due to its enormous application prospects. Even though the development of complex DVS schemes constitutes a great challenge, the importance along with the necessity of preserving the safeness of society have played a decisive role as one of the main incentives for researchers and developers to work on the integration of some technologies, such as data management and computer vision, to produce systems that are reliable and effective enough to serve as a solution for tasks like cities surveillance, video analytics and efficient video management in order to support city officials and/or security employees in their duty.

Contemporary smart cities are in prompt need of means for intelligent decision-making, where a crucial role belongs to smart DVS systems necessary to monitor and detect early events in indoor and outdoor scenes of airports, train stations,

highways, parking lots, stores, and even shopping malls to avoid disasters such as fire, terrorist attacks, traffic congestion. Therefore, there is a need for the development of smart DVS systems that will automatically detect potentially dangerous situations. Furthermore, DVS might be performed using different VIoT characterized by heterogeneous constrained visual sensor nodes, lower cost, smaller size, limited processing power, average quality cameras, and smartphone cameras.

Let us consider a contemporary smart city scenario, where multi VIoT (Video Internet of Things) Connected Surveillance Cameras (CCTV) are disseminated all over the urban environment. Moreover, let us there are CCTV can be HD and non as well as the video frame acquisition is variable, e.g., 30 FPS, 60 FPS. A challenging task is, therefore, realizing such complex video surveillance architecture satisfying different requirements. To this aim, an osmotic flow must be adopted. For example, by considering an HD CCTV and a frame rate of 60 FPS, an edge node could not be able to manage the acquisition from an HD CCTVs due to hardware constraints; so, the acquisition will be managed by a private cloud node more powerful and able also to minimize latency - crucial for video surveillance. Hence, the FaaS that manages the video frames acquisition will be deployed on the private cloud node, which will also be connected to the CCTV camera.

7.5.2 Environment

Experiments were carried out in three different scenarios: i) Public Cloud-to-Edge, ii) Private Cloud-to-Edge and iii) Hybrid Cloud (Private and Public), respectively.

For each scenario, the cloud node is deployed on the public cloud platform provided by GARR (an Italian no-profit organization founded by several research organizations), having the following hardware and software characteristics: 4 VCPUs @3.2 GHz, RAM 18GB, OS Ubuntu 18.04 LTS. The private cloud node is deployed on a locally hosted server having 2 VCPUs @2.4 GHz, RAM 4 GB, OS Ubuntu 18.04 LTS. The edge node is hosted by a Raspberry PI 3 Model B+ having a quad-core CPU @1.4 GHz and RAM 1GB, OS Raspbian Jessie. Each node has installed Docker version 18.09.1 with Docker Swarm and OpenFaaS version 0.8.9.

We connected a 5 MP video security camera on the private cloud/edge node according to the configuration scenario to capture frames in each scenario. Since the OpenFaaS functions follow the stateless principle, they do not provide data saving during execution. They are also distributed on the cluster, so they do not see exactly the locally hosted data. To overcome this limitation, we used MongoDB Database, which is accessible by the functions. So, the video frames will be firstly saved on MongoDB and then accessed by the OpenFaaS functions.

Hence, our osmotic flow is composed by three different functions: $FaaS_1$, $FaaS_2$ and $FaaS_3$ respectively.

Workflow: As previously explained, the experimentation objective is to validate the functionality and assess the performance of our OC flow based on FaaS in a smart city video surveillance scenario, detecting and recognizing objects/-faces in a video stream coming from a video camera.

To achieve this, we created the $FaaS_1$ - a python function that acquires the frames from the video camera and sends them, one by one, to a $FaaS_2$ - a Python function that uses OpenCV, an Open Source Computer Vision library, to processes them and returns the name of the recognized object/face.

In this workflow, we also deployed a third $FaaS_3$, which saves frames into a MongoDB database. We have chosen to use the MongoDB because, for example, in scenario i) the video camera is connected to the edge node on which we deployed $FaaS_1$, $FaaS_3$ and on the cloud one we deployed the $FaaS_2$, the latter, given the stateless nature of OpenFaaS functions, the latter will not be able to access the frames produced by the $FaaS_1$ and hosted by the edge node. As well as, passing directly the image acquired from the video camera to the OpenFaaS function is not simple because it is necessary to convert it into the required format, which is limited from this point of view; in fact, the OpenFaaS functions read in input from the STDIN channel coming from terminals. To overcome this limitation, we introduced the $FaaS_3$, which saves the frames acquired from the video camera into MongoDB, and then $FaaS_2$ will do the processing reading directly them from the database. We also have evaluated the $FaaS_2$'s container creation time on the edge, private and public cloud nodes.

In order to verify the feasibility to apply this workflow in real-world applications, we quantified the execution time of each proposed FaaS. To have truthful results, for each proposed scenario, we collected 30 subsequent experiments and calculated the average times and confidence interval of 95% for all proposed FaaS.

7.5.3 Results

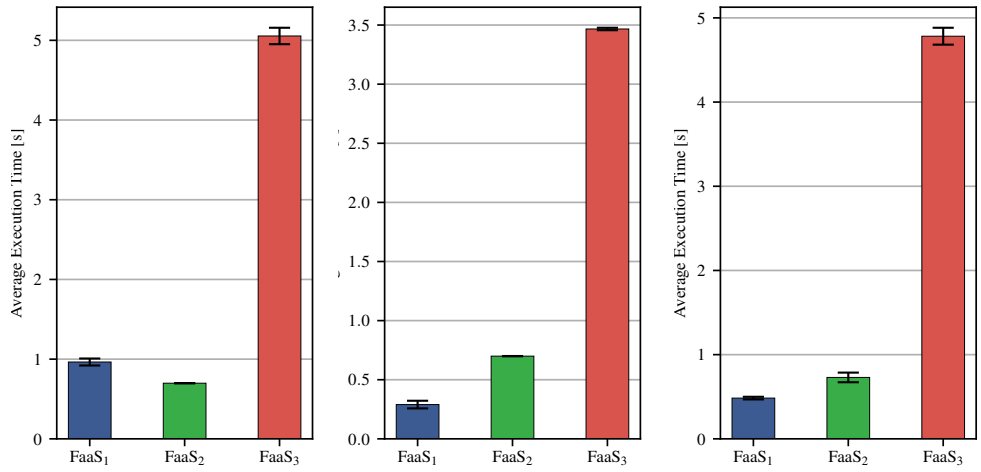
The following examples explain the expected results of the proposed scenarios.

Scenario #1: Public Cloud-to-Edge In this scenario, the $FaaS_2$ and $FaaS_3$ execution is forwarded to the public cloud, while $FaaS_1$ is deployed on edge. Hence, we have deployed the $FaaS_3$ and $FaaS_2$ on the public cloud. Instead, we capture frames and invoke the $FaaS_2$ from the edge node. This scenario exemplifies a situation where the requested $FaaS_2$ is hardware intensive, therefore taking much time to execute on edge. As well as, the video camera is connected to the latter one. Due to the cloud node's high processing power, it is beneficial to forward the execution request to the public cloud.

Figure 7.2 (a) shows the collected execution times for all three FaaS in this scenario. We notice that $FaaS_1$ requires in average 0.69 s, $FaaS_2$ 5.05 s and $FaaS_3$ 0.96 s.

Scenario #2: Private Cloud-to-Edge This scenario is similar to the previous one, but with the difference that the $FaaS_2$ and $FaaS_3$ are deployed on the private cloud; also, in this case, FaaS is invoked from the edge node. This scenario describes a situation where the requested $FaaS_2$ is latency-sensitive and requires computational capabilities that cannot be satisfied by the edge node.

Figure 7.2 (b) shows the collected executions times for all three FaaS in this scenario. Comparing the results with those obtained in Scenario #1 we notice that $FaaS_2$ and $FaaS_3$ respectively have less execution times; in particular, $FaaS_2$ - 3.46 s and $FaaS_3$ - 0.22 s respectively. This trend is due to the private cloud node closed to the edge node with respect to the public cloud one.



(a) Public Cloud-to-Edge **(b)** Private Cloud-to-Edge **(c)** Hybrid Cloud

Figure 7.2: Average execution time [s].

Scenario #3: Hybrid Cloud In this scenario, the $FaaS_2$ and $FaaS_3$ are deployed on the public cloud, while $FaaS_1$ is deployed on the private cloud. This scenario exemplifies a situation where $FaaS_1$ captures frames from a high-resolution video camera. This FaaS should be executed on a private cloud node instead of an edge because the latter does not satisfy the hardware requirements. Moreover, the processing done through $FaaS_2$ is hardware intensive and is executed on the cloud. As well, the MongoDB is hosted on the cloud, given the high dimension of the captured frames.

Figure 7.2 (c) shows the collected executions times for all three FaaS in this scenario. As we expected, also in this scenario, $FaaS_1$ requires, on average, the

same execution time as that collected in Scenario #1 and Scenario #2. Moreover, $FaaS_2$ takes 4.78 s while $FaaS_3$ 0.48 s respectively. We notice that the execution time required by the $FaaS_2$ is greater than that obtained in Scenario #1; this is due to latency communication with public cloud introduces. The $FaaS_2$ execution time is less than that obtained in Scenario #2; this is because the public cloud node has more computational resources with respect to the private cloud one and requires less time to process. Finally, a similar trend is also registered for $FaaS_3$.

We have also evaluated the $FaaS_2$'s container creation time on edge, private and public cloud. Figure 7.3 illustrates the trend obtained. As we expected, the $FaaS_2$'s container creation time on the edge is slower with respect to those obtained on the private cloud node and edge; as well as the $FaaS_2$ creation time on the private cloud is slower than that obtained on the public cloud. This behaviour is due to the edge node's hardware constraints with respect to the private and public cloud. To conclude, we notice that in all three scenarios, the obtained

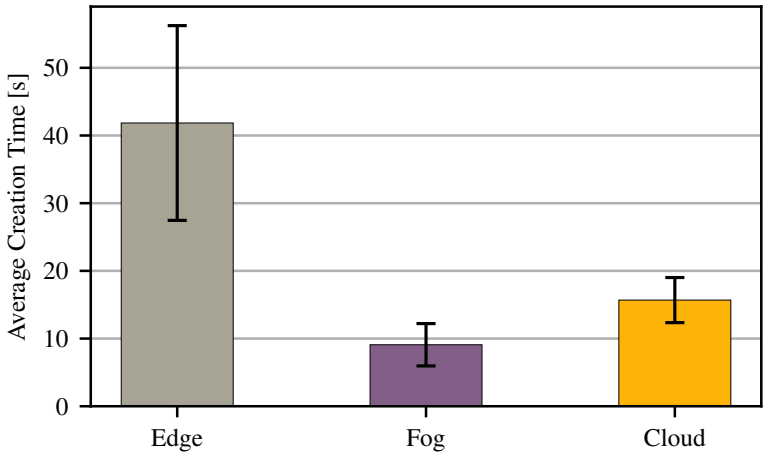


Figure 7.3: $FaaS_2$'s container creation time [s]

execution times are acceptable for video surveillance real-world applications; almost in average 0.7 s for $FaaS_1$, 4.4 s for $FaaS_2$ and 0.55 s for $FaaS_3$ respectively. Moreover, each scenario's choice depends on different parameters, as explained in Section 7.4.

7.6 Conclusions and Future Work

This chapter's focus was on providing an innovative OC ecosystem leveraging FaaS; there is also introduced the concept of hybrid architecture combining both microservices and Serverless architectures. Furthermore, in order to support the FaaS-based OC ecosystem, by using the event-driven nature of FaaS as well as

given that Serverless functions can also act as scheduled jobs, event handlers etc., and not just as services, an osmotic flow model for video surveillance in smart cities is presented. This allows simplifying the management complexity due to the highly distributed and heterogeneous nature of the OC ecosystem and ensuring a high level of scalability, elasticity, and flexibility.

To validate the functionality and assess our OC flow's performance based on FaaS, a number of experiments have been made. These experiments furthermore improve the understanding of the usability of the OC flow in real-world applications.

In our on-going work, we plan to investigate the osmotic flow's feasibility in more real use cases; we also plan to implement and test the OC ecosystem using hybrid architectures based on microservices and FaaS.

Part III

How to Model and Emulate Osmotic Computing Ecosystems

8 OsmoticToolkit

Digital services are increasingly becoming cyber-physical and osmotic, combining cloud resources with fog, edge, and IoT devices. This trend can be observed in the eHealth domain or in smart city applications where the location of software deployments and data processing matters. Before such applications go live, careful planning with real system emulation is necessary. We claim that the OsmoticToolkit, although in the early stages, is the first emulation environment designed to address this challenge. In this chapter, we introduce the emulator's functionalities and validate experimentally with an e-health scenario, using a reference deployment of a microservice-based hospital application. The experimental results carried out show its effectiveness providing valuable support for understanding the impact on resources, workloads, and QoS requirements within cloud-ddge/fog-IoT scenarios while preserving the users' SLAs.

8.1 Introduction

Internet of Things (IoT) is a profound technology evolution incorporating billions of devices (sensors, RFIDs, smartphones, and wearables) owned by different organisations and people who are deploying and using them for pervasive digital services. IoT-Business-News forecasts 24 billion connected things generating $1.5 trillion in 2030 [206]. Their number, capabilities, scope of use and data volume keep growing & changing rapidly. This leads to higher complexity in IoT applications. Thus, new distributed computing paradigms, such as Edge Computing (EC) or IoT-Cloud Computing, have been investigated to extend IoT resources into centralised data centres (e.g., clouds) or at the edge of IoT systems (e.g., edge micro datacenters). Among the most promising ones is Osmotic Computing (OC), motivated by the lack of a scalable, interoperable, configurable solution for delivering IoT applications in complex, heterogeneous and dynamic computing environments. As previously anticipated in Chapter 5, the OC paradigm looks at the opportunistic management of MicroELements (MELs) to improve the Quality of Services (QoS) and networking management, interoperability, and efficiency of next-generation IoT applications.

The main issue arising in using such combined computing models to support IoT applications is the management of different physical/virtual infrastructures (e.g., data centres, edge devices, IoT devices & gateways) according to specific application/service requirements (e.g., latency, data volume, responsivity, pro-

cessing delay, privacy). In particular, it is hard to determine a priori how to deploy the MELs composing IoT applications into different infrastructures – since resource availability, system load, and connectivity features can unpredictably vary over time. OC provides convergence and holistic planning for IoT, edge, fog, and cloud technologies of the C2T continuum in this scenario.

This chapter presents an OC emulation tool called *OsmoticToolkit* that executes workflows based on MELs in particular conditions where the edge has limited computation and networking capabilities. We evaluate the work with the case of a rural medical lab with limited processing power and rare ability to use a cloud service. The toolkit provides valuable support for understanding the impact of processing power, workloads, and QoS requirements while preserving the users' SLAs.

The chapter is organized as follows. After the background and related work in Section 8.2, we provide motivations (Section 8.3) and explain the tool design in Section 8.4, followed by the implementation in Section 8.5 and performance evaluation in Section 8.6. Finally, Section 8.7 highlights the advantages of OsmoticToolkit and reasons about future directions.

8.2 Background and Related Work

Planning and testing applications in distributed computing environments that involve a high level of heterogeneity and complexity is costly since the provisioning and management of needed hardware are very expensive. In recent years, simulation techniques have been proven to be a partial solution for investigating different aspects of complex osmotic systems, e.g., service configuration and deployment, resource placement, or management strategies. Cost-effective emulation tools based on controlled service execution further align results with reality. To evaluate the current state of the art, we define four key criteria: i) hybrid cloud-edge/fog-IoT Architecture, ii) dynamic infrastructure topology (modelling of physical networks and virtual topologies), iii) resource provisioning approach, and iv) real application execution.

8.2.1 Simulation Tools

While simulators cannot execute real applications, their designs are of interest to ensure that our emulator meets functional expectations on expressible scenarios. They simulate hybrid Cloud Computing (CC) and Edge Computing (EC) based on simplified models. Many EC simulators extend CloudSim [207], such as iFogSim [208]. It provides an evaluation platform for resource allocation policies. A limitation of iFogSim is that the location of end devices is static and cannot be updated; further, it is limited to the Discrete Event Simulators (DES) and has poor scalability because of the CloudSim characteristics.

Similarly, EdgeCloudSim [209] extends CloudSim. In contrast to iFogSim,

it is focused on a more dynamic and realistic investigation of service usage and implements mobility models for mobile devices. MyiFogSim [210] extends iFogSim to support mobility through the migration of VMs between cloudlets. IoTSim [211] also extends CloudSim. It emphasises the processing performance of large IoT applications that process huge amounts of data. As a result, it adds storage and big data processing layers with map-reduce cloudlets to CloudSim. EdgeCloudSim and IoTSim both inherit the same scalability and DES limitations as iFogSim.

Yet Another Fog Simulator (YAFS) [212] is a DES for cloud & fog networks. Its primary focus is the performance evaluation of placement, scheduling, and routing strategies. Applications are modelled as a set of modules that run services, following the concept defined by iFogSim. Sphere [213] extends SCORE [214] and allows creating a cloudlet network based on graphs, generating dynamic and parallel workloads, and specifying the geographic location, resource density, and deployment requirements. However, it does not support the nodes' mobility and lacks the migration model of a workload.

Simulators such as iFogSim, CloudSim, and YAFS support the dynamism and on-demand requirements of Fog services/applications via VM elasticity and migration, federation policies, and computational clustering nodes. EdgeCloudSim only supports federation and scalability between nodes of the same tier (only cloud or only fog), which means that it is impossible to achieve a proper orchestration along with the cloud to fog continuum.

While the simulation tools simplify the evaluation process, the differences be-

Table 8.1: Summary of simulation and emulation tools.

Tool	Hybrid	Dynamic	Resources	Execution
CloudSim	|	|	allocation	|
iFogSim	IoT/fog	|	alloc+monit	|
EdgeCloudSim	edge	mobility	alloc+monit	|
myiFogSim	fog	migration	alloc+monit	|
IoTSim	IoT/-cloud	|	allocation	|
YAFS	fog	app	allocation+fail	|
Sphere	edge	workloads	orchestration	|
EmuFog	fog	|	placement	containers
Fogbed	fog	volatility	|	containers
MockFog	fog	|	allocation	VM-based

tween the simplified and real production scenarios are remarkable. Experimental results can be unfaithful, especially for a highly dynamic osmotic scenario. Network emulation can support repeatable and controllable experiments with real

applications that solve some simulation problems and improve its experimental results.

8.2.2 Emulation Tools

EmuFog [215] is an emulation framework for Fog Computing built on top of MaxiNet [216]. It emulates Fog nodes and makes use of Docker to run applications. EmuFog is more realistic than simulation tools, implementing a Fog node placement algorithm based on arbitrary latency costs to the connections between hosts and switches. It does not support the mobility of clients and fog nodes. Fogbed [217] extends the Mininet network emulator. In contrast to EmuFog, it uses Docker containers to run virtual nodes and allows developers to dynamically add, connect, and remove nodes from the topology. This feature allows investigating real-world Fog infrastructures, where Cloud services are provided closer to the network edge. It does not support mobility, security, fault tolerance, scalability, and reliability and does not implement any resource providing model for the application services. The application used to evaluate the emulator is not faithful to real-case scenarios and lacks configurability and extensibility. It is strongly-coupled with the virtual node code. EmuFog and Fogbed have scalability support regarding the communication and topology infrastructure but lack strategies to deal with applications' on-demand requirements inside computational nodes.

Finally, MockFog [218] allows the emulation of a Fog Computing infrastructure in arbitrary cloud environments. It creates a VM for every node lacking scalability and being expensive when implementing an infrastructure model with many nodes. It also has problems when smaller devices are involved, as they cannot be accurately emulated.

Several optimisation solutions for service deployment have been investigated concerning the service execution plan and resource provisioning in hybrid cloud-fog-IoT environments, each of them focusing on different target variables. In [219], the authors investigate a solution for maximising the number of services deployed on fog devices by applying heuristics to solve their service placement problem based on collected response times. The approach proposed by the authors is not realistic as they assume that each service of an application can be executed independently from workflow structures with chained output-input links. In [220], the authors address service provisioning as a Delay and Payment optimisation problem, which is the trade-off among energy consumption, delay performance, and payment cost when deploying services. In [221], the authors propose a distributed alternating direction method of multipliers to approach the allocation as a trade-off between the users' Quality of Experience (QoE) and the fog nodes' power efficiency.

Table 8.1 summarises the capabilities of related simulation and emulation approaches regarding the four key criteria mentioned before.

8.3 Motivations and Requirements

Osmotic capabilities are of increasing importance when considering the growing digitalisation of life. To counter the pandemic in 2020, several national governments have released software applications with workflows encompassing mobile phones, telecom carriers and cloud services. The degree to which processing logic has been placed on one side depends on political strategies. Due to their volatility, engineering effort can be saved from a technical perspective when mechanism and policy are properly separated and the placement gains flexibility. The mechanism then entails a decomposition of the application into either resource-bound or portable parts, the MELs, that can be implemented as cloud functions, containers, other MicroService technologies (MS) and associated MicroData (MD) representations. The assignment of MELs to computing resources can become dynamic at deployment time. It requires osmotic management, where MELs can move across different infrastructures based on several potential triggers (e.g., performance, networking, security/privacy, or cost-oriented). The Software-Defined Membrane (SDMem) in OC enforces these concepts filtering the MELs flows in the system.

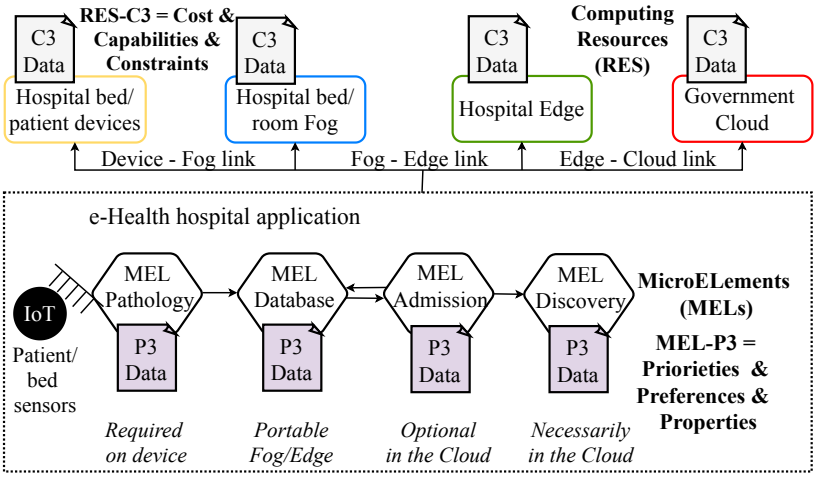


Figure 8.1: Scenario of osmotic e-health application: four MELs connected horizontally in a workflow linked to computing resources across devices, fog, edge and cloud.

Let us consider the case of a rural medical lab with limited processing power and network access. The application deployment needs to prioritise which MELs can be deployed locally if there are resource contentions and otherwise needs to optimise within the given degrees of freedom among all the portable MELs. Hence, the application consists of a mandatory on-site part on health testing, a portable part on image detection that can run either on-site or in the Cloud, and an optional part on recommending further tests that run in a particular Cloud environment. Figure 8.1 explains this scenario. It involves

descriptors of application requirements, including deployment priorities (P3), and corresponding hosting capabilities (C3), facilitating deployment dynamics.

In contrast to the streamlined onboarding of software applications in clouds, the decomposition and description of applications for such dynamic scenarios is currently a challenging engineering task. To give application engineers the ability to prepare, using an emulator will save precious time and effort and facilitate resource planning. To overcome the limitations of existing emulation tools, we require OsmoticToolkit to provide the following technological advances:

1. Combined inherent support for all of the four key criteria outlined in Table 8.1 by design;
2. Well-defined usage procedure with explicit infrastructure and application modelling, infrastructure instantiation and application pipeline deployment;
3. Service-oriented integration with APIs/CLIs to fit into automated osmotic and cloud-native systems.

8.4 OsmoticToolkit Workflow Design

This section highlights OsmoticToolkit's features and overviews a high-level design workflow outlining the emulation abstractions, toolkit core components, and interactions.

8.4.1 Design Principles

OsmoticToolkit should support four main features:

Hybrid topology OC ecosystems consist of hybrid complex IoT-oriented computing systems where both resource-constrained edge/fog nodes and cloud-hosted services in public/private, hybrid or multi-cloud are involved. The generation of such topologies should be realistic with a high degree of confidence, allowing to assign capacities (e.g., CPU, memory) and capabilities (e.g., hosted services, applications) for each infrastructure component trivially.

Dynamicity In OC, computation is dynamically distributed across nodes based on QoS requirements and available infrastructure resources. Particularly, services with short lifecycles are frequently instantiated and offloaded. It also occurs at the lowest level of the infrastructure, where edge and IoT nodes may join and permanently leave the network according to service usage, failures, policies, and maintenance operations. OsmoticToolkit should provide a holistic approach for managing the network infrastructure and application.

Resource provisioning and orchestration Applications range from simple IoT-based sensing to complex data processing inherent to e-health or smart city systems with different QoS and SLA (e.g., location/latency awareness, security levels, heterogeneity, interoperability), processing (e.g., batch, real-time), mobility. OsmoticToolkit should consider these aspects during the orchestration allowing dynamic and flexible resource provisioning and monitoring mechanisms.

Execution OsmoticToolkit should allow the execution of realistic applications on top of the infrastructure topology. This feature should minimise the effort in preparing applications, avoiding costly changes in stack and tools.

8.4.2 OsmoticToolkit Infrastructure Model

Figure 8.2 illustrates the high-level design workflow of OsmoticToolkit. In this scenario, the DevOps Engineer (DOE) is involved in several phases, as explained in the following.

Phase #1: Infrastructure Modeling An OC ecosystem comprises Infrastructure Elements (IEs) such as compute nodes, Network Elements (NEs) such as switches and routers, and Application Elements (AEs) deployed on top of the infrastructure at different levels. The infrastructure topology is modelled as directed graph T = (V, E) where V is a set whose elements are called vertices (e.g., IE), and E is a set of paired vertices, whose elements are called links. Each IE is characterised by different computing properties (e.g., CPU, memory), while different network parameters characterise each link and thus NEs (e.g., latency, bandwidth, packet loss). As exemplified in Figure 8.2, if the link between N1 and S1 (N1-S1) has a delay of 3 ms, S1-S2 has 5 ms, and S2-N2 has 2 ms, the overall delay is 10 ms. During this phase, the DOE starts with such a graph specification before assigning properties to vertices and links using a template primitive. OsmoticToolkit relies on pre-configured container images, e.g., cloud, fog, edge, and IoT, retrieved as IEs the toolkit's registry. Thus, each IE in the emulated network executes independently, increasing the emulation's realism and affording behaviour similar to that in production infrastructures.

Phase #2: Application Modeling Similar to Infrastructure Modeling, applications deployed in an osmotic ecosystem are structured as graph P = (V, E), where V = MELs and E = interconnections. OsmoticToolkit associates the concept of the pipeline to an application. Namely, the pipeline's anatomy describes MELs properties and how they are interconnected. The DOE defines independent pipelines inside the toolkit and interacts with each separately. Each pipeline is described within a template primitive. For each MEL inside the pipeline, the DOE can specify different resource requirements, constraints, and scheduling policies.

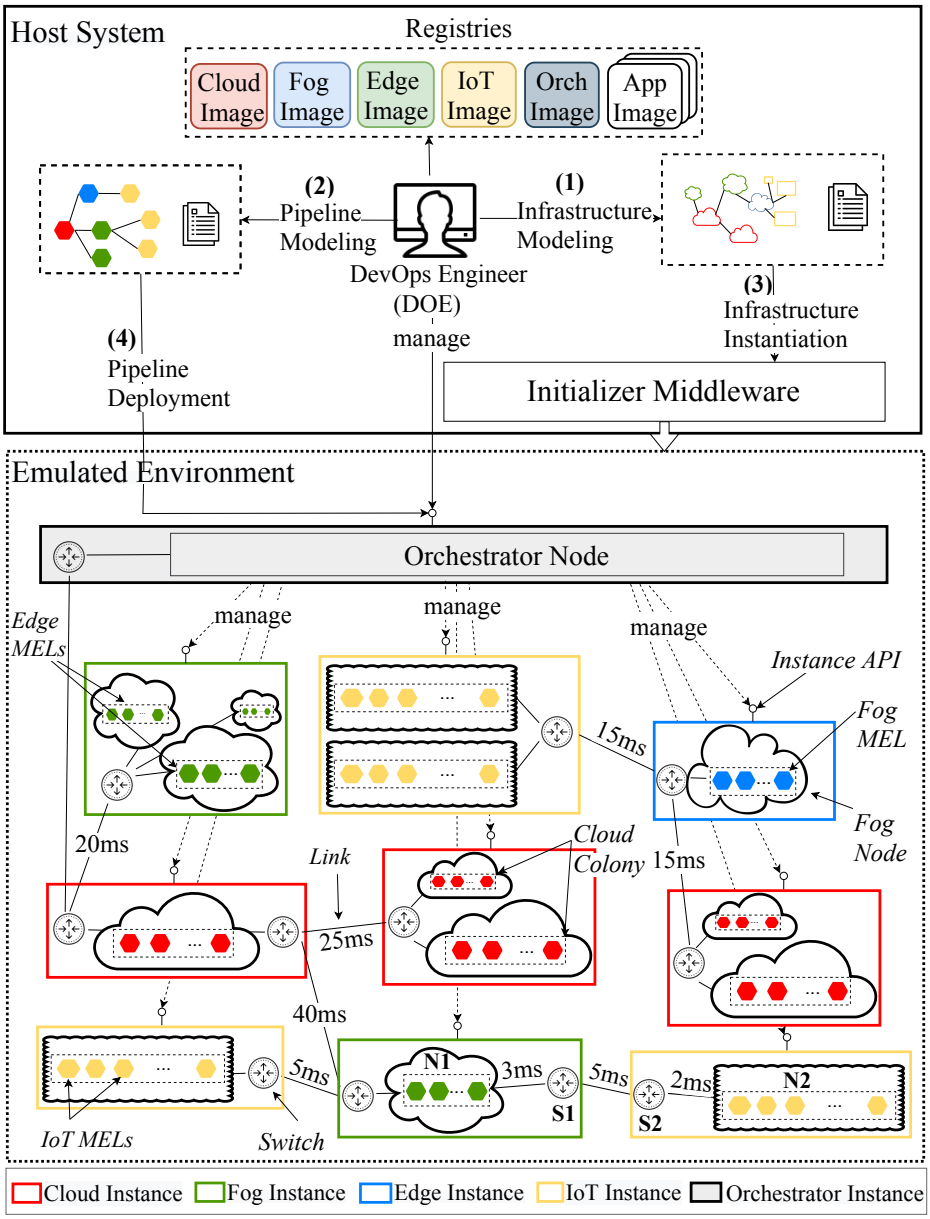


Figure 8.2: OsmoticToolkit general workflow: two boxes, one representing the emulated environment, and one running OsmoticToolkit to create the emulated environment.

Phase #3: Infrastructure Instantiation During this phase, the Initializer Middleware loads the template primitive containing the infrastructure description and instantiates it by deploying instances, e.g., for the cloud (CI), fog (FI), edge (EI) and IoT (IoTI), as well as Compute or Managed Nodes (MNs), switches, and links into an emulated environment. An example of a running environment with 9 instances is shown in Figure 8.2. The instance abstraction allows the management of MNs and switches as a single entity. Generally, each instance

involves one or more switches and MNs. An instance containing more of those is called a colony, e.g., Cloud Colony (see Figure 8.2). This colony composes a new and isolated network slice. It is relevant in mobile networks, where the limited radio resources are shared among multiple users that experience variable radio quality conditions over time. To properly control and manage the QoE in the network slice, the NEs are adapted to the different service requirements, and the applications adjust the configuration to the network capabilities over time dynamically.

Each instance has associated a resource model defining the amount of computing resources distributed among its MNs. Each MN in the infrastructure model is mapped to a running container. The resource model allows the DOE to apply limitations that impose each instance's available resources according to a specific scenario.

To efficiently control and manage this complex osmotic ecosystem, an effective control system becomes essential. The Orchestrator Node (ON) handles this system. During this phase, the ON is instantiated within an Orchestrator Instance (OI), and it is the point of entry for DOE via API endpoints and automation handlers. Thus, it manages the entire infrastructure, deploys pipelines or offloads MELs, spawns new nodes, and forwards configuration details. It also offers its complete functionality via API endpoints so that OsmoticToolkit can easily be integrated into existing automated testing workflows.

Phase #4: Pipeline Deployment The application is deployed on the emulated infrastructure through ON's Instance API endpoint for this phase. The ON firstly evaluates the MEL's predefined constraints and scheduling policies specified by the DOE within the template. This filtering step allows selecting a set of nodes obeying the specified restrictions. Next, it evaluates the computation requirements for each MEL. It generates an optimal execution plan for the pipeline for describing the MELs contextualisation across the cloud, cloud, fog, edge and IoT MNs through an optimisation algorithm. Namely, it assigns each MN satisfying the computation requirements of one or more MELs by minimising a specific cost function. It is assumed that the ON has full control over which MELs are executed on each instance. Finally, the MELs are instantiated and run in Docker containers on top of the MNs (i.e., Docker-in-Docker) according to the previously generated optimal scheduling plan. The DOE interacts with the pipeline using the orchestrator APIs by controlling the MELs status, performing updates, tearing down the pipeline, or deploying a new one.

8.5 Emulator Implementation

This section discusses the architectural components, along with their interactions (illustrated in Figure 8.3), and motivates the set of technologies used to imple-

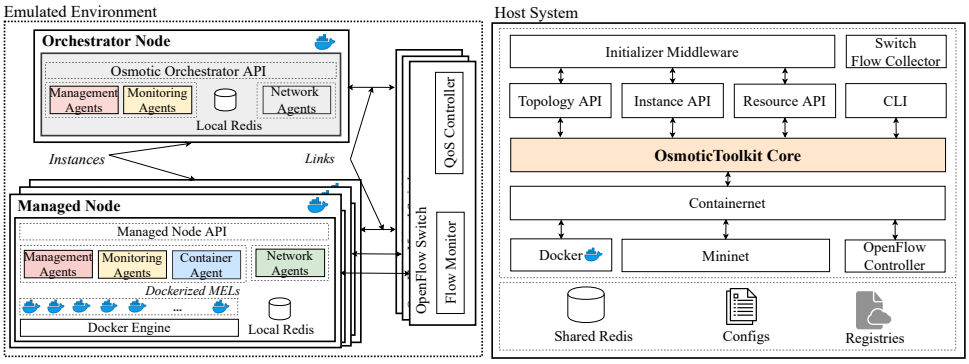


Figure 8.3: OsmoticToolkit architecture: Two boxes, emulated environment with virtual nodes including the respective agents, and host system including the OsmoticToolkit architecture around OsmoticToolset Core

ment OsmoticToolkit[2]. The technological choices are constrained by several non-functional requirements such as flexibility, ease of use, cost-effectiveness, scalability, extensibility.

8.5.1 Core Components and APIs

OsmoticToolkit core The toolkit is based on Containernet [222] that extends the Mininet emulation framework by adding Docker containers at runtime as compute instances within the emulated topology. We chose Containernet/Mininet because they are highly prevalent in the distributed computing community, open-source, scalable, easily extensible, and flexible. The core offers three convenient APIs.

Topology API This API is based on the Mininet Python API. It interacts with the core to allow the DOE to generate different topologies straightforwardly. Virtual switches are implemented leveraging Open vSwitch.

For example, in an osmotic ecosystem, using the concept of instance, a node with constrained resources attached to a virtual switch can represent a virtual gateway for IoT. Similarly, a virtual switch that connects a node with sufficient resources to run applications can be considered a fog or edge node. Each node comprises at least two interfaces, where one is used to access the control and management of the nodes and one or more other ports are used for forwarding the data on emulated topology. These interfaces are virtual Ethernet pair ports (veth devices) and tunnel ports (TUN/TAP) resulting from Generic Routing Encapsulation (GRE) or Virtual Extensible Local Area Network (VXLAN) tunnels.

According to the topology a user has defined, a virtual link is responsible for creating interconnections among two or more nodes. There are two types of virtual links: i) *point-to-point*, which connects nodes, and ii) *point-to-multipoint*,

2 OsmoticToolkit: http://github.com/alinabuzachis/OsmoticToolkit

which creates a control and management bus among nodes. These virtual links can be either Ethernet virtual links or GRE or VXLAN tunnels.

Standard SDN controllers configure the switches as part of the Mininet emulation environment, e.g., OpenFlow. Instances can communicate with external devices on the Internet or act as proxies for cloud applications located in remote data centres using Network Address Translation (NAT). The DOE can implement other advanced network protocols and forwarding setups.

Instance API It provides an Infrastructure-as-a-Service (IaaS) endpoint allowing to manage MNs within instances in an adaptable way. The core interacts with this API to control the instance semantics. The default approach adds one specific Instance API to each type of instance. With this kind of abstraction, an instance can be managed in different ways, e.g., as a colony with different resource allocation or placement policies for each MN. The Instance API can be easily extended, and DOEs can implement their management interfaces on top.

Resource API Osmotic ecosystems are highly heterogeneous, being characterized by the existence of nodes with different computation capabilities. For instance, while cloud services provide virtually infinite computation resources to their clients, fog/edge and IoT offer limited computation, memory, and storage resources to applications. The Orchestrator Node must consider this heterogeneity when offloading, resource allocation, and scheduling decisions are taken. This API lets the DOE apply resource limits for each instance, such as constrained CPU and memory, and specify additional parameters such as a pricing model. OsmoticToolkit allow associating with each kind of instance, a pricing model that will be explained better in the following. The resource models are instantiated when containers are allocated or released. By specifying resource models, we can limit the overall available for each instance so that one instance does not influence others.

OsmoticToolkit supports two kinds of resource models:

1. Predefined Resource Model: It assigns predefined resources to each instance; in particular, there are 7 fixed models, e.g., m1.small (CPU: 1 and memory: 512 MB), m1.medium (CPU: 1 and memory: 1024 MB) and so on (see Table 8.2 for a complete list of all available predefined resource models). If no resource model is specified, the m1.medium is applied for all instances by default, while the OI uses m2.medium.

2. Customized Resource Model: Conversely, this model allows defining custom resource limitations for each kind of instance. For cloud-based instances, some real resource limitations and pricing models can be configured through a Python-based script that scrapes current container pricing data from different service providers, e.g., AWS, Azure.

Table 8.2: Available predefined resource models.

Model	CPU (cu)	Memory (mu) MB
m1.tiny	1	128
m1.small	1	512
m1.medium	1	1024
m2.medium	2	1024
m1.large	4	2048

8.5.2 Command Line Interface

The Command Line Interface (CLI) allows developers to interact with the emulated components to modify configurations or run commands.

Initializer Middleware It is implemented as a Python-based script and runs either on the DOE host system or within the build pipeline. Its primary purpose is to load the infrastructure topology definition template (see Listing 8.1) and start the emulation. As shown in Listing 8.1, each node is identified by several properties. In particular, for the CLOUD node c1, is a public cloud; its information is filled with the current container pricing data from the AWS provider. For each link connecting two nodes, there are associated with different parameters such as latency (unit: ms), bandwidth (unit: GHz), and packet loss (unit: %).

Listing 8.1: Infrastructure Topology definition JSON file.

```
{
  "nodes" : {
    "c1" : {
      "type" : "CLOUD",
      "model" : "public",
      "provider" : "AWS",
      "size" : "a1.medium",
      "memory" : 4,
      "vCPU" : 1,
      "price-per-hour" : 0.0255
    },
    "e1" : {
      "type" : "EDGE",
      "model" : "Raspberry",
      "RAM" : 1,
      "CPU" : 1.4,
      "cost" : 35
    }
  },
  "node_links" : [
  {
    "source" : "c1",
    "target" : "e1",
    "BW" : 10,
    "latency" : 0,
    "packet_loss" : 2
  }
```

```
    ]
}
```

Listing 8.2: Pipeline definition YAML file.

```
---
ship_provider: dynamic # static
name: p1
# ships:
  # ship1:
    # ip: x.x.x.x
services:
  foo:
    image: ubuntu
    security_opt: [affinity:service != test]
    requires: [test]
    labels:
      constraint: [node.type == CLOUD, node.model == public, node.provider
          == AWS]
      priority: high
    limits:
      memory: 50m
      cpu: 1
    instances:
      foo-1:
        # ship: ship1
  test:
    image: mongo
...
---
```

8.5.3 Emulated Environment

The ON is the central component serving RESTful API as an MN with additional responsibilities. The ON extends MaestroNG [223] by implementing dynamic scheduling, monitoring, service offloading, and policy management. Specifically, MaestroNG is a simple and easily extensible orchestrator for Docker-based, multi-host environments that offer service-level and container-level controls that rely on declared service dependencies static placement.

As shown in Figure 8.3, the ON in the topmost level exposes an API through which the DOE interacts with the emulated ecosystem. In the lower levels, there are several agents, each one dealing with specific tasks. We use Celery, a simple, flexible, and reliable distributed task queue system, to roll up these agents. Celery is configured to use the local Redis database as a message broker. The main agents instantiated on every MN are described below.

Monitoring Agents One Monitoring Agent (MonAs) is the Resource Monitoring Agent (RMA). It is responsible for collecting utilisation metrics from the MNs, allowing the ON at every level to be aware of the capabilities of the MNs present within the topology. RMA has been implemented through a non-intrusive Python library, e.g., psutil, and resides on each MN. Another MonA is Healthcheck Monitoring Agent (HMA). It is responsible for maintaining the

infrastructure's topology by regularly monitoring the health of the MNs. HMA resides only on the orchestrator node.

Management Agents Management Agents (MAs) interface with deployment and control interaction within the infrastructure components. The Node Management Agent (NMA) is the most crucial component residing on the ON. It is responsible for handling every interaction with DOE through ON's APIs. It also deals with two primary operations, MEL scheduling and offloading. According to the pipeline's deployment requirements set by DOE, the NMA generates a resource provisioning plan via an optimisation algorithm for describing the MELs contextualisation across the MNs. Its implementation is extensible to allow the DOE to plug other resource provisioning strategies. OsmoticToolkit supports two provisioning approaches natively: a) static - inherited from MaestroNG and b) dynamic.

The first step necessary to deploy a pipeline composed of several MELs is its definition. We extended the MaestroNG version 2 YAML schema by adding support for selecting the resource provisioning strategy (e.g., static or dynamic) and defining several constraints, anti-/affinity scheduling policies and priority-based mechanisms to enforce the Software-Defined Membrane concept. Namely, the affinity policy specifies whether the MELs should be collocated on the same node (affinity) or if they should be spread onto as many nodes as possible (anti-affinity). A generic deployment pipeline with two MELs (foo and test) is illustrated in **??**. Going through **??**, the *ship_provider* field allows us to select between static and dynamic resource provisioning strategy. Using a static provisioning strategy, the DOE has to know a priori the network topology, e.g., the nodes' IP addresses. The IP addresses are necessary to define the shipment *ship1* in *ship* field, namely, the node on which to schedule the MEL. The static approach does not use any optimization algorithm for resource provisioning; consequently, the nodes can result in over/underprovisioned.

The *name* field allows us to specify the name of the pipeline. Thus, a unique name identifies each pipeline, allowing the deployment of different, even independent pipelines within the system. The *services* field allows us to specify the MELs composing the pipeline. For each MEL, we can specify a *name*, the *image* used for the Docker container and the complete list of parameters that can be found in the MaestroNG documentation [223].

The *security_opt* field allows us to set a deployment restriction defined through an anti-/affinity policy. Namely, the affinity policy specifies whether the MELs should be collocated on the same node (affinity) or if they should be spread onto as many nodes as possible (anti-affinity). For instance, in **??**, there is implemented an anti-affinity policy for the *foo* MEL. Namely, *foo* MEL cannot be deployed on the same node where *test* MEL has been deployed. There is also supported a dependency-based mechanism between MELs implemented using the *requires* field. With this restriction, the *test* MEL has to be deployed before the *foo* MEL.

By specifying constraints, the DOE implements a filtering mechanism selecting only the nodes able to obey to the following restrictions, such as *node.type*, *node.model*, *node.provider*.

There is also a priority-based mechanism allowing to define for each MEL a priority level. In particular, there are implemented three priority levels, e.g., high, medium, and low. The high priority is used for MELs that are computation-intensive and require a significant amount of resources. A MEL with a high priority is scheduled on a cloud node. The medium priority is assigned for latency-sensitive MELs; the MEL is scheduled on a fog or edge node according to the amount of computation resources it requires. Finally, the low priority allows us to schedule the MELs using the best-effort approach. The high priority MELs are scheduled primarily, then the ones with medium priority and finally the latest with low priority on the remaining available nodes. This can ensure specific MELs only run on nodes with certain isolation, security, or regulatory properties.

Using a static provisioning strategy, the DOE has to know a priori the network topology, e.g., the IPs of the MNs on which to schedule the MELs. The static approach does not use any optimisation algorithm for resource provisioning; consequently, the nodes can result over/underprovisioned. The static provisioning strategy implemented in MaestroNG does not provide any constraint-, priority- or anti-/affinity-based scheduling policy.

Each MEL is also characterized by different requirements for computation resources specified by the *limits* label. In particular, for each MEL, the DOE can specify the amount of CPU and memory it requires.

In OsmoticToolkit, dynamic scheduling is treated as an assignment problem. Several algorithms are used to solve assignment problems, e.g., Kuhn's Hungarian algorithm, heuristic algorithms (e.g., simulated annealing algorithm, ant colony algorithm, particle swarm algorithm, and genetic algorithms). Heuristic algorithms are generally used to solve highly complex assignment problems. However, as it starts the search randomly, it cannot guarantee to reach the optimization result. Kuhn's Hungarian algorithm is an analytic algorithm. Because of its simplicity and ability to find the optimal solution without requiring validation, Kuhn's Hungarian algorithm is widely used to solve assignment problems. Because of its simplicity and ability to find the optimal solution without requiring validation, we choose Kuhn's Hungarian algorithm [224] to solve the assignment problem for the MELs' optimal scheduling.

Each assignment problem is associated with a cost matrix. The rows contain the workers or MNs, and the columns comprise jobs or MELs. The cost function c_{ij} used to compute the cost matrix used is given in Equation (8.1) [225] and is defined as the weighted sum of the following five parameters: i) number of containers running on each MN (n_c), ii) percentage of memory used (n_{mem}), iii) average CPU utilization (n_{cpu}), iv) amount of CPU the MEL requires and finally

(a_{cpu}) and v) amount of memory the MEL requires (a_{mem}). Where i varies from 1 to the number of MN N, j varies from 1 to the number of parameters 5. w_{ij} is a weight between 0 and 1, $component_i$ is an array containing the values of the parameters as mentioned above, and $f_j(t)$ represents that these parameters vary in time. The weights used are $w_1 = 0.4$, $w_2 = 0.04$, and $w_3 = 0.01$ (see [225] for further details).

$$C = \sum_{i=1}^{N} \sum_{j=1}^{5} w_{ij} \times component_{ij} \times f_j(t) \tag{8.1}$$

Kuhn's Hungarian algorithm treats the Optimal Assignment Problem (OAP) as a combinatorial problem to efficiently solve an $n \times n$ task assignment problem in $O(n^3)$ time. It starts with a complete bipartite graph, G = {V, U, E}, where V and U are the sets of nodes in each partition of the graph, and E is the set of edges. The cost estimations become edge weights and each node and MEL becomes a vertex. Starting with an empty matching, Kuhn's Hungarian algorithm's basic strategy is to search for augmenting paths in the equality subgraph repeatedly (see Algorithm 1).

If an augmenting path is found, the current set of matches is augmented by flipping the matched and unmatched edges along this path. Because there is one more unmatched than a matched edge, this flipping increases the cardinality of the matching by one, completing a single stage of the algorithm. If an augmenting path is not found, the $l(x) + l(y) = w(x, y)$ variables are adjusted to bring additional edges are added into the equality subgraph by making them admissible, and the search continues. n such stages of the algorithm are performed to determine n matches, at which point the algorithm terminates.

It should be noted that the orchestrator has to be also able to recognise situations when a MEL on edge must be offloaded on a cloud MN or vice-versa. The Osmotic Orchestrator Agent (OOA) is a MA that periodically performs an orchestration by running the Hungarian algorithm to check whether the actual scheduling plan is optimal. If it is not, a new optimal scheduling plan is generated. The OOA contacts an NMA's API to update the new scheduling plan and perform the necessary MEL's offloading.

The offloading is implemented as live migration to limit the service downtime. This technique used is based on lazy or post-copy memory migration using Checkpoint-Restore in Userspace (CRIU). CRIU is a tool that allows such a live migration of a hierarchy of processes (container) between hosts. As implemented in CRIU, live migration techniques are used for moving a container instance from one physical host to another while preserving the running state of the containerized applications and maintaining open network connections.

Algorithm 1: Hungarian algorithm.

Input:
Set nodes, Set requests
Cost Function:
C = $w_1 \times n_c + w_2 \times (n_{cpu} + a_{cpu}) + w_3 \times (n_{mem} + a_{mem})$
Cost Matrix:
A valid n×n assignment matrix represented as the equivalent complete
weighted bipartite graph G = (X, Y, E), where |X| = |Y| = n.
Output:
A perfect matching, M.

1: Generate an initial labelling l and matching M in G_e.
2: If M perfect, terminate algorithm. Otherwise, randomly pick an exposed
vertex u ∈ X. Set S = {u}, T = Ø.
3: If N(S) = T, update labels:
$\delta = min_{x \in S, y \in Y-T}\{l(x) + l(y) - w(x,y)\}$

$$l'(v) = \begin{cases} l(v) - \delta & \text{if } v \in S \\ l(v) + \delta & \text{if } v \in T \\ l(v) & otherwise \end{cases}$$

4: If N(S) ≠, pick y ∈ N(S) - T
(a) If y exposed, u → y is augmenting path. Augment M and go to step 2.
(b) If y matched, say to z, extend Hungarian tree: S = S ⋃ z, T = T ⋃ y,
and go to step 3.
* Definitions:

- Equality graph G_e = {e(x, y) : l(x) + l(y) = w(x, y)}
- Neighbor N(u) of vertex u ∈ X: N(u) = {v : e(u,v) ∈ G_e}.

During migration, several resources are transferred over the network, e.g., CPU state, memory state, network state, and disk state. Post-copy minimises the application downtime during live migration.

Contrary to other migration approaches, such as pre-copy, post-copy transfers all memory pages until after the CPU state has already been moved and resumed on the destination node. The current post-copy implementation in CRIU starts with a checkpoint (*criu dump*) with provided $--lazy-pages$ option. Using this option, the process memory is collected into pipes, and non-lazy pages are stored into image files or transferred over to the destination host via page-server instantiated on-demand on the destination VN via the Migration or Offloading

Agent. After the checkpoint is completed (at the dump finish stage), a TCP server is started to handle page requests from the restore host. Another NMA is the Osmotic Node Agent (ONA) as the main component of the MNs, exposing the necessary APIs to communicate with it, e.g., health-check endpoint, resource monitoring, migration, as previously explained. It is implemented as a Python Flask service.

Container Agent On every cloud, fog, edge, and IoT MN, the Docker engine is installed. The Container Agent (CA) is represented by the persistent process that exposes the Docker API. It is used for communication with the Docker daemon, allowing it to control the status of containers. The ON contacts remotely the MNs' Docker daemon via this API every time it needs to deploy a new MEL.

Network Agents The Network Agents (NAs) collect flow statistics on each MN by running flow monitors for each network interface on nodes and switches. This data is stored in the Redis local instance for future analysis.

Database Instances Our system involves two Redis DB instances. The local instances are used to store the configuration parameters and MN statistics, such as resource utilisation. The shared Redis instance is for the versioning of the scheduling plan. It is also used to store the infrastructure topology description. There are also store the resource utilisation metrics gathered from the RMA.

8.6 Experiments and Evaluation

In this section, we present experiments and evaluations that we undertook to quantify the efficiency of OsmoticToolkit in modelling and simulating OC environments.

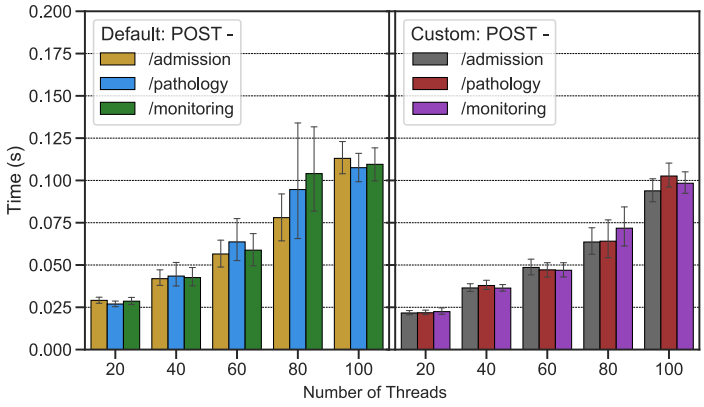
8.6.1 Methodology

The evaluation criteria leverage a set of metrics that can be used to evaluate the proposed emulator's effectiveness in terms of a) responsiveness, b) reactiveness and c) agility.

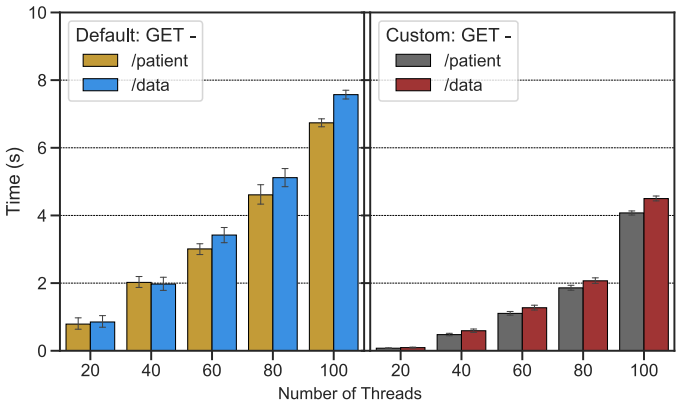
Responsiveness Assures that the system continues to have adequate response times even when the load rises. Generally, a system that strives to handle many requests with acceptable latency requires more computation resources. Hence, the system can be over-provisioned to keep system responsiveness. Such resources are expensive and a system should always optimise the use to be cost-effective. One of the responsiveness properties is SLA preservation. The guarantees provided by the SLA concern the fact that response times to user requests should never exceed a certain threshold.

Reactiveness Indicates the reaction time of an environment composed of multiple individual applications blending into one unit while staying aware of each other to produce a workflow execution.

Agility Indicates the ease of applying changes in the environment. One helpful metric in this context is the system downtime when there is a dynamic reconfiguration. It represents the amount of time that the system or a portion of the system is not working correctly.



(a) POST



(b) GET

Figure 8.4: Hospital application MEL's average response times (s).

8.6.2 Application Use Case Scenario and Infrastructure Setup

To validate the toolkit's effectiveness, we implemented the microservice-based rural hospital application illustrated in Figure 8.1. Its workflow involves each patient in several steps. The patient is initially examined, with preliminary clinical trials conducted to identify possible pathologies. The "MEL Pathology" manages these trials. It updates the patient's Electronic Medical Record (EMR) with the collected health data. Then, he passes through admission for recovery. "MEL

admission" handles this by updating the EMR with the recovery information. Health data are collected and stored within the EMR for local processing during the recovery phase through continuous monitoring. In the case of abnormal values, the EMR is submitted to offloaded processing.

All MELs composing the eHealth application are implemented with Java 12 and SpringBoot 5. To discover MELs, we use the REST-based Eureka server. The patient's actions are simulated using JMeter and with periodic requests. We configure a test plan with 100 threads (equivalent to 100 patients), a duration of 8 minutes (480 s), and ramping-up over 2 minutes with 3 minutes of hold time. Hold time confirms that the system handles the load and its performance stays stable and does not deteriorate. There is a tunable interarrival time between one API call and the next one. It is managed by setting up a Gaussian random timer with a deviation of 500 ms, and a constant delay offset of 1000 ms.

The infrastructure topology used to run the MELs consists of 1 CI, 1 FI (connected to CI), 1 EI (connected to FI), and an IoTI (connected to EI). We consider two resource models: (i) default: all the instances have minimal hardware resources (e.g., 1 CPU and 1024 GB of memory), (ii) custom: different user-defined hardware resources characterize each instance, such as cloud (CPU: 1, memory: 4096 MB), fog (CPU: 1, memory: 2048 MB), edge (CPU: 1.4, memory: 1024 MB) and IoT (CPU: 1, memory: 512 MB). The experiments have been conducted on an OpenStack cluster instance by CloudLab[3]. The instance has 16 GB RAM, 8 VCPU and 240 GB of storage and runs Ubuntu 18.04 LTS.

8.6.3 Results and Findings

Network Parameters Selection We modelled the described scenario with suitable latency and packet loss values for the links to reflect as much as possible a real site for the emulated infrastructure.

To do so, we conducted a series of experiments to measure the latency and packet loss in three different setups: (i) edge to cloud and (ii) fog to edge, and (iii) IoT to fog by performing 3000 ping requests.

As a cloud node, we used the OpenStack instance previously listed. As an edge node, we used a Raspberry Pi 3 with 4 cores @ 1.4 GHz and 1 GB RAM with Raspberry Pi OS Lite connected to a router via Ethernet and, as an IoT device, an iPhone 7+ with iOS 14.1 connected via WiFi with the same router. As a fog node, we used a MacBook Pro 3.3 GHz Dual-Core Intel Core i7 with 16 GB RAM and MacOS Catalina 10.15.7 connected via WiFi with the same router. The router, edge, fog, and IoT devices are physically located in the same room.

First, we ran a ping on the cloud to measure the Round Trip Time (RTT) from the edge node. We found the average RTT is 67.56 ms and 4.0% of packet loss. Then, we ran a ping on the edge to measure the RTT from the Fog. We found

3 CloudLab: info.cloudlab.zhaw.ch

the average RTT is 42.189 ms and 0.2% of packet loss. Finally, we ran a ping on the IoT to measure the RTT from the edge. We found an average RTT of 7.57 ms and 0.0% of packet loss. We used the measured average RTTs from the real experiment and provided that as an input parameter for our model's emulated links (see Figure 8.1).

Responsiveness To ensure the SLA preservation, it is helpful to see the MEL's average response times when the load is low and use them as a reference for checking the degraded performance when the load increases. As shown in Figure 8.4 (a), we note the average response times increase with patients' number. The three endpoints have different requests over time; JMeter periodically performs a set of requests to different API endpoints over time. Figure 8.4 (a) shows that the response time per each API endpoint. In particular, the results obtained by applying a custom resource model revels lower average response times and show a more stable trend when the number of patients increases. The average response times obtained with both configurations are acceptable (around 0.125 s using a default resource model and 0.1 s using a custom resource model when the maximum number of patients is reached), preserving the SLA. The trend is even more evident in Figure 8.4 (b).

In conclusion, the slowdowns are not particularly severe, and the responsiveness results when both resource models are applied to show that the application remained responsive throughout all executions.

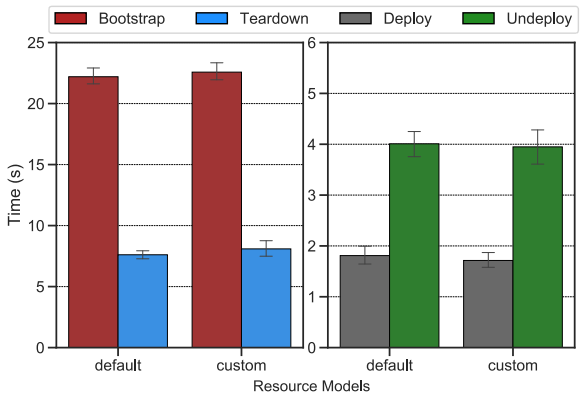


Figure 8.5: Infrastructure and pipeline instantiation: barcharts comparing bootstrap, teardown, deploy and undeploy actions across default and custom resource models.

Reactiveness To assess the proposed system's reactiveness, we evaluated the workflow from the infrastructure bootstrap/tear down and pipeline deploy/undeploy sides. We collected these results using both resource models default and custom and dynamic and static resource provisioning approaches. According to Figure 8.5, we notice the infrastructure bootstrap with a custom resource model

requires, on average, 22.5 s, while the default one takes on average 24 s. A similar trend is obtained when the infrastructure is torn down. This is because the nodes initialised with the custom resource model have a higher amount of resources assigned and require more time to instantiate them. The same is also applied when the infrastructure is torn down and all allocated resources are released.

For the pipeline deployment, we used both static and dynamic provisioning approaches. Figure 8.5 shows the obtained response times when a static resource provisioning approach has been used. In the case of a dynamic approach, the response times must be summed up the time the Kuhn's Hungarian algorithm takes to provide the scheduling plan. That is, on average, 4.29 ms. As we can see, Kuhn's Hungarian algorithm's execution time is almost negligible and does not impact response times. We notice the opposite trend with respect to the bootstrap/tear down response times in terms of deploy and undeploy response times. We notice that deploy/undeploy operations performed on top of the nodes initialised with the custom resource model require less time respect when the default resource model is used.

This is because the nodes initialised with the custom resource model are more powerful in terms of hardware resources than those instantiated with the default resource model. In this way, all the processes run more fluently are quicker to initialise components and respond. Similar behaviour is obtained when the status of the pipeline is checked by calling the corresponding API. The response times collected for deploying the pipeline do not consider the time necessary to download the MEL's images. To download all the images, there are required almost 50 s on average more. Therefore, the overall response times are acceptable and in line with what we were expecting.

Agility Evaluation of the system's agility is done by dynamically performing a MEL offloading from a cloud node to an edge node. MEL's offloading allows minimising cost and energy consumption of services to end-users by improving QoE leveraging the awareness of their location, network, mobility, and context information. The main objective is to show that the offloading causes no system downtime minimising as much as possible the application's downtime in case of reconfiguration. We performed the MEL's offloading when both resource models were used to initialised the nodes. We therefore used for this evaluation two MELs of different sizes to understand how the image size impacts the overall times. In particular, one of 75.3 MB and another of 182,41 MB.

As explained, offloading is implemented as live migration consisting mainly of two phases: (i) checkpoint and (ii) restore. We performed 30 subsequent executions and gathered the average response, checkpoint/restore and downtime times. The response time measures the elapsed time between the first POST call on the /offload endpoint and the MEL restore phase's start on the destination node.

Figure 8.6 show that the average response time is greater when a default

resource model is applied. The same behaviour is obtained for the checkpoint/restore times. The mean MEL downtime during the offloading is 0.5 s when the default model is applied and 0.35 ms with the custom one. This fact highlights that the service has been offloaded with minimum downtime. This is because, with the custom model, the nodes are more powerful and perform faster operations. However, for an image of 75.3 MB, the timings are higher than with an image size of 182.41 MB. The image size impacts heavily on the checkpoint/restore performances. Thus, the offloading operations can be performed dynamically and asynchronously while the system is running, introducing zero downtime for the system and improving the overall agility.

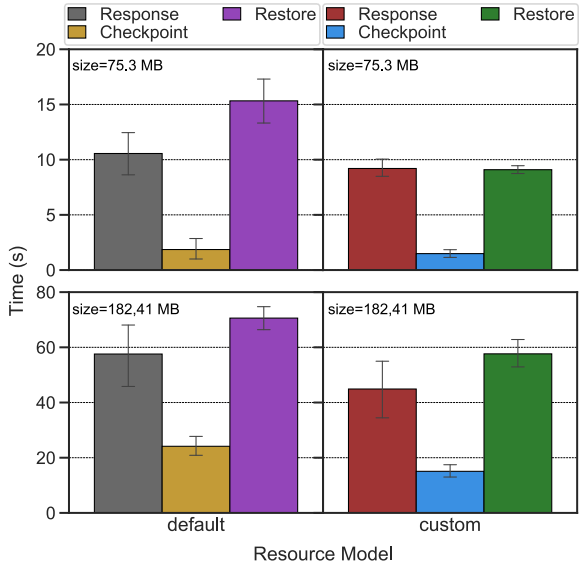


Figure 8.6: MEL offloading: performance barchart comparing responses, checkpoint and restore times across default and custom resource models; checkpointing is much faster than the other operations.

8.7 Conclusions and Future Directions

With this work, we contributed and evaluated OsmoticToolkit, emulating resources and network connections for distributed applications deployment to IoT devices and fog, edge and cloud resources within the C2T continuum. As the emulation approximates real deployments and eases their planning, it exceeds simulation approaches. Moreover, it is the first emulator to combine four key characteristics: hybrid topologies, dynamicity with service offloading, resource provisioning/orchestration, and realistic container execution.

Based on the achieved toolkit, we intend to conduct broader studies on integrating commercial cloud providers and minimising the runtime behaviour

differences between real deployments and emulation by supporting further cloud-native MELs and SDMems.

Part IV

Advanced Resource Scheduling for Composite Applications accross Continuums

9 Rule-based Resource Matchmaker (RBMM)

Where shall my new shiny application run? Hundreds of such questions are asked by software engineers who have many cloud services at their disposition, but increasingly also many other hosting options around managed edge devices and fog spectrums, including for functions and container hosting (FaaS/CaaS). Especially for composite applications prevalent in this field, the combinatorial deployment space is exploding. We claim that a systematic and automated approach is unavoidable to scale functional decomposition applications further to fully exploit each hosting facility. To support engineers while transitioning from cloud-native to continuum-native, we provide a Rule-based Matchmaker called RBMM that combines several decision factors typically present in software description formats and applies rules to them. Using the MaestroNG orchestrator and OsmoticToolkit, we also contribute to integrating the matchmaker into an actual deployment environment.

9.1 Introduction

Continuum applications are widely considered as the next evolutionary step of cloud applications [226], by departing from the notion of single or even multiple clouds and instead incorporating other computing facilities such as data-generating nodes (mobile devices, IoT sensor nodes) and intermediaries (edges, fogs) [227].

In such continuums, various deployment patterns can be identified. Figure 9.1 shows four such patterns, out of a lot more that are already used in industrial applications: pipelining follows the flow of data mostly for analytics purposes; multiplexing considers the parallel use of multi-cloud services for increasing availability or security; offloading adaptively switches on a cloud on-demand, and switching optimizes the connection latency by choosing between the intermediary and direct cloud access depending on the situation of each link.

Instead of hardcoding where each part of a composite application runs, it is possible and indeed desired to automate that process and specify requirements along with desired patterns programmatically. This requires transferring the knowledge from humans, who would do the hardcoding, to knowledge bases and tools that understand the characteristics of application parts, resources along the continuum, and other decision factors. Based on this knowledge, matchmaking can be performed to solve the assignment problem and yield suitable deployment instructions that fulfil all hard constraints and consider soft preferences.

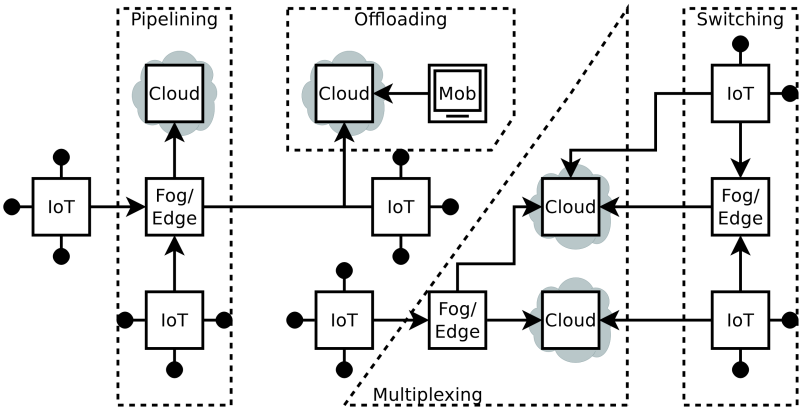


Figure 9.1: Application deployments across continuums.

This chapter defines application and resource models, then categorizes the associated decision factors in the Rule-based Matchmaker (RBMM) model. To increase automation, we present suitable acquisition techniques for each, complemented by propagation, skipping, deployment and accumulation rules. Subsequently, we present simple matchmaking algorithms to yield suitable deployment topologies (Section 9.2.1 and Section 9.3). To increase the applicability of the work, we implement the matchmaking along with a composite application deployment scenario (Section 9.4), where the OsmoticToolkit is taken into consideration as a practical demonstrator before diving into related works and concluding our work (Section 9.5 and Section 9.6).

9.2 Models, Decision Factors and Rules

9.2.1 Definitions and Models

We define a composite application A to consist of several software artifacts a_i which are loosely coupled and instantiated as application execution units, or parts, with certain scaling factors, i.e. A = $s_0 \times a_0, s_1 \times a_1, ...$

A continuum resource collection R consists of independent resources r_i whose owners or operators can differ, leading to further differences in location and technical characteristics, including the resource level (infrastructure, platform, middleware). Both artefacts and resources have certain properties, although not all of them are guaranteed to be explicitly expressed in machine-readable descriptions. Sometimes, they are also loosely expressed, for example, a runtime environment java without the corresponding version number. Therefore, we assume those deployment factors to contain a measure of uncertainty, i.e., F = $u_0 \times f_0, u_1 \times f_1, ...$ In component notation, resource factors are offered, whereas artefact factors are required. Some factors only exist for either artefacts or re-

sources, while others exist for both; this is expressed by the factor scope (A, R or both).

A deployment plan (assignment) is the projection $A \leftarrow_{C+P} R_{used} \subseteq R$ under a set of conditions C and a set of preferences P. Conditions must be fulfilled (e.g., sufficient memory to run the application, monthly cost not more than a certain limit) whereas preferences are used to determine the winning resource combination out of several that fulfil the conditions (e.g., smallest possible latency). Multiple preferences can be combined by weights, i.e. $P = w_0 \times p_0, w_1 \times p_1, ...$ All conditions and preferences are expressed with application-specific rules (Ψ) referencing arbitrary deployment factors F and applying to any pair (a_i, r_i).

Our model is limited by not taking data dependencies or workflows into account. Specifically, we assume that any application part can generate data and transmit it freely to any other part. We acknowledge this limitation while claiming that even the simplified model advances systematic deployment methodologies beyond current deployment tool designs for clouds and continuums. On the other hand, the model is flexible by allowing to skip certain factors so that a subsequent matchmaker or deployment tool can perform further micro-optimization. Together with the deployment rules and propagation and accumulation rules, these skipping rules lead to a highly flexible approach that fits multiple deployment patterns and topologies.

9.2.2 Decision Factors

Deployment processes are constrained and influenced by decision factors inherent to any application part or resource, or even both, as well as to the composite application as a whole. The precise definition of these factors depends on the operational environment and primarily on automating the collection of factors. We propose the following common categories of hosting-related factors to control application deployments in continuums:

1. Infrastructure. Application parts are only deployed if their requirements in terms of low-level computing hardware if fulfilled.
2. Platform. For higher-level deployments, operating systems and language runtimes and their version and managed back-end services are important.
3. Connectivity. For latency-sensitive or bandwidth-intensive services, the connectivity to end-users or other resources in the continuum is significant.
4. Security. Vulnerabilities and protection levels further influence deployments.
5. Economics. Pricing models, plans and tariffs must match before deployment is considered valid.

Furthermore, we foresee external context information, such as date, time and system load, but omit their handling from our current approach. Table 9.1 contains an exemplary set of suitable decision factors. All values are restricted by a (typed) value space and complemented with physical units for numerical values. The selection of factors may be extended and customized depending on the application domain or specific infrastructure topologies.

Table 9.1: Subset of Deployment Decision Factors

Scope	Name	Values (examples)
A, R	memory	128 MiB, 2 GiB
A, R	runtime	python:3, java
A, R	latency	5 ms
A, R	duration	900 s
A, R	zone	intranet, dmz, internet
A	vulnerability	backdoor, CVE-477
A	consistency	true, false
A	complexity	high, medium, low
A	port	9233
R	country	gb, cn
R	trust	high, low
R	billing	monthly, pay-per-use, free
R	gpu	true, false

9.2.3 Acquisition Techniques

Acquisition of accurate and up-to-date values of decision factors is defined as the automated process of building up the knowledge on both software applications, ranging from the source code level to packaged artefacts ready for deployment and computing resources.

Software Artefacts

To acquire metrics from composite application units, we rely on the Microservice Artefact Observatory (MAO) [228] that can perform a static and dynamic assessment on several artefacts, including Docker containers. MAO is a federated data management system that schedules and orchestrates individual tools that acquire different software artefacts metrics, creating a more detailed overview of the artefact's quality and properties. The result is an aggregated knowledge base of software artefact information. Data can be contributed or validated by any member node of the federation.

Computing Resources

Researchers have proposed automated approaches to scrape key characteristics of cloud services from provider websites as a means to automate the acquisition process [229]. Similarly, CloudPick acknowledges the variety in cross-cloud service selection and contributes a translator component for automatic semantic enrichment [230]. We point out that certain subjective factors, such as trust in a resource provider, needs to be modelled manually.

9.2.4 Rules

Rules (Ψ) applying to factors are composed of propagation rules (Ψ_π), skipping rules (Ψ_σ) deployment rules (Ψ_δ) and accumulation rules (Ψ_α). They are applied in this order: First, propagation rules use invariants to complement missing factors or change existing factors in application compositions and resource sets. On this basis, skipping rules temporarily hide factors – marking them to be skipped during processing – so that they remain intact in the output mapping and serve as input for further post-processing. Then, matchmaking is performed with deployment rules, and all successful assignments imply the use of accumulation rules to adjust post-deployment resource characteristics. In case an assignment is reverted, for instance, through backtracking, the accumulation rules are executed in inverse order to roll back forecasted resource modifications.

Propagation Rules

By considering a hierarchical application and resource model, it is possible to specify which factors at lower levels invariably influence those at higher levels, and vice-versa, as well as lower-level siblings (up-, down- and side-propagation). Although many composition formats adopted in the industry can have multiple levels, the hierarchy can only support two levels on the application side. However, resources have sub-resources, for instance, a VM instance offering both CPU and GPU computing access. Through propagation, further efficiency gains can be achieved when only modelling some of the factors that have to be modelled manually or whose automated acquisition consumes much time. For instance, a software composition affected by a security vulnerability in one constituent unit is considered itself tainted, representing an up-propagation, whereas another component in the same composition remains unaffected by itself. Specifically, the following propagation rules, including two trivial ones, are useful in continuums and need to be expressible as a preprocessing step in matchmaking.

1. Replication. All factors in α trivially apply to all other instances of the same artefact. This applies to all static factors as well as, assuming they share the same resource, to dynamic runtime-related ones such as maximum

task processing rate. However, in our work, we apply these rules before deployment and do not consider dynamic factors.

2. Subsumption. The resource needs are trivially defined as the conjunction of all constituent resource needs, including scaling factors (up).

3. Bounding. The upper bound of latency in α is mirrored in A (up).

4. Tainting. Any quality deficiency or security vulnerability in α is mirrored in A (up), and any trust level in r is mirrored in the subset of R that shares the same operator (side).

Skipping Rules

A potential use case of matchmaking is generating a subset of valid deployment mappings and further post-processing it with another matchmaking or optimization tool that might use different algorithms for achieving increased output precision. Skipping rules mark attributes such as CPU or memory needs by the application. The corresponding offered capacity by the resources effectively leads to them being skipped during the match-making. Afterwards, they get reinstated on the resulting mapping. Possible skipping rules are:

1. Context. Skip CPU and memory factors, deferring these technical details to later, and instead perform matchmaking primarily on contextual factors.

2. Feasibility. Pre-check whether a deployment is technically feasible by skipping non-technical factors such as trust, country or geolocation.

Deployment Rules

These rules set constraints on where each application part α can be deployed. We require the deployment rules to express the following scenarios:

1. An application processing sensitive personal information shall not be deployed in hosting locations whose jurisdiction does not support certain minimum guarantees on privacy.

2. An application subject to a vulnerability shall only be deployed into the demilitarised zone (DMZ), not in the internal network behind the firewall.

3. Any application part α needs to be deployed into a resource with sufficient memory. For latency-sensitive applications, the entire application A needs to fit within one resource.

Accumulation Rules

Due to resource sharing and resource utilization in general, each deployment leads to some changes in factors. We differentiate between constant factors unimpeded by any deployment (e.g., location of a DC) and those changing their values according to accumulation rules, for instance, available memory being reduced by any running application. More accurately, we assume that resource access is either unlimited, shared, or exclusive. A concrete set of rules might look as follows:

1. The amount of free memory in r is reduced by the memory claimed by α (shared).
2. The range of free port numbers in r is reduced by any allocated port in α (shared).
3. A GPU available as sub-resource in r is occupied by any α claiming to perform GPU computing (exclusive).

The distinction into resource access models influences the permissible algorithms. Under the assumption that resources are infinite ($|R| = \infty$) or largely available beyond what can be consumed by A, as in most clouds, simple combinatorial matchmaking can be performed. Otherwise, a complex assignment and satisfiability problem needs to be solved. We propose a depth-first recursive tree search for constrained devices where after each candidate assignment, its validity is determined by successful matchmaking of the remaining subtree, otherwise rolled back.

9.3 Rule-Based and Weighted Matchmaking Concept

The matchmaking process guarantees that if A's deployment, including its constituent parts, e.g., microservices, to R, is possible, a valid deployment plan is returned.

9.3.1 Design and Architecture

RBMM's matchmaking component operates as a service to be queried by deployment tools after scanning the composition description and artefacts to be deployed. Figure 9.2 outlines the process of acquiring the factors through automated scanning (acquirer tools) and manual curation, creating an instance of A and R models, and submitting them to the matchmaker yield a resource-aware deployment plan.

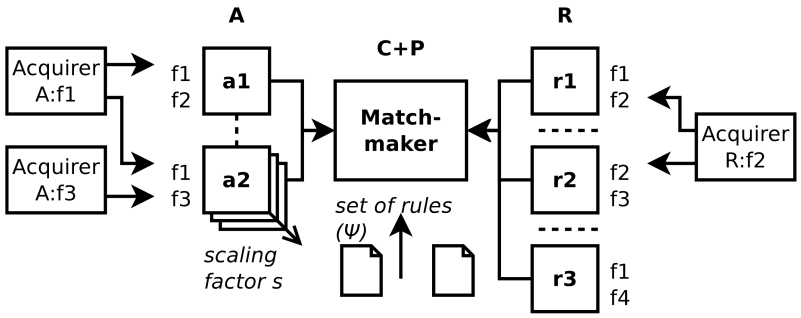


Figure 9.2: Matchmaking based on collected factors, rules and constraints.

9.3.2 Matchmaking Algorithms

The goal of the matchmaking process is to achieve an optimal deployment by creating, rating and ranking all possible assignment combinations of A ← R. We briefly describe an exemplary *fast combinatorial algorithm* assuming infinite resource availability and a more thorough *recursive tree search* assuming finite resources. These algorithms are suitable for simple scenarios but would be replaced with more capable ones in actual deployment systems.

Combinatorial Algorithm

The iterative combinatorial algorithm attempts to perform a mapping of all resources on all application parts. For any part, a mapping has been found and validated according to deployment and accumulation rules. As further resources are then skipped, the algorithm complexity is approximately $O(n \times \frac{n}{2})$ for |A|=|R|=n.

Tree search algorithm

In the recursive tree search, again for each application part, a mapping is attempted. Any successful mapping of a_i applies the accumulation rules, followed by a recursive invocation of A / a_i, i.e., the set of application parts without the one already mapped. In case the invocation returns a valid result, it is proven that all application parts have been mapped successfully. Otherwise, the accumulation is reversed and the next resource is mapped for a_i. The complexity is approximately $O(n \times \frac{(n-1)^2}{2})$.

9.4 Implementation

9.4.1 Acquisition Tools

We have implemented and integrated MAO to automatically produce non-functional property metrics for Docker containers, Docker compositions, serverless applications packaged as Serverless Application Model (SAM), and various other formats typically used in cloud and fog software. In contrast, we have not implemented the automatic acquisition of metrics for resources but acknowledge the approaches mentioned above' existence.

For the example of Docker images, there are currently two separate factor acquirer tools in production and others under development. One collects public image metadata from Docker Hub, including supported system architectures and artefact size. The other uses the Clair scanner to scan images, producing a report on the number and severity of security issues according to Common Vulnerability and Exposures (CVE) present in the image. These reports are retrieved from the knowledge base and compiled into a merged tree of image factors. Listing 9.1 shows a typical MAO output for a single Docker container retrieved from Docker Hub, determining which resources this image can be deployed to and what security constraints apply to it. The combined tree format allows for direct feeding into the matchmaker.

Listing 9.1: Source of application factor acquisition.

```
{
  "image": "docker.io/library/mongo:4.2", "architecture": "amd64",
  "features": "",
  "variant": null,
  "digest": "sha256:93f3dc8491f23d507...",
  "os": "linux",
  "os_features": "",
  "os_version": null,
  "size": 164677487,
  "CVEs": {
    "Medium": 12,
    "Low": 25,
    "Negligible": 11
  }
}
```

9.4.2 Matchmaker Library

First, we implemented the matchmaking algorithms as a Python library and coupled it with a test tool to synthetically generate applications, resources and rules. In an experiment with 10'000 application parts and the same number of resources, i.e. 100 million possible combinations, around 488 million factor comparisons were generated. On a single-core Intel i7 processor with 2.60 GHz, using only deployment and accumulation rules, the iterative combinatorial matchmaking took 3.3 s.

For a more modest scenario with 200 application parts and resources, i.e., 40'000 possible combinations and 175'275 factor comparisons, the combinatorial matchmaking finished in less than 0.05 s. For this scenario, the recursive algorithm implementation became feasible and finished in 80.1 s

9.4.3 Emulator Integration

To demonstrate the practical usefulness of RBMM, we integrated the resource-aware deployment of applications to continuums with OsmoticToolkit. OsmoticToolkit is extensively described in Chapter 8. As OsmoticToolkit has its matchmaking logic based on cost functions and the Hungarian algorithm [231] covering a specific set of metrics, including CPU and memory, we activate the skipping rule *Context* in RBMM to restrict the matchmaking to contextual factors. OsmoticToolkit allows modelling from-scratch infrastructure topologies and applications using graph theory. On an abstract level, the infrastructure topology is modelled as a directed graph T = (V, E), where vertices V is a set of resources, and E is a set of two-sets (sets with two distinct elements) of vertices, whose elements are network links between them. Each graph vertex is annotated with appropriate metadata, including computing properties, while different network parameters characterize each link (e.g., latency, bandwidth, packet loss). Each computing resource is emulated leveraging Docker. Similarly, applications deployed in an osmotic ecosystem are structured as a graph P = (V, E), where vertices are represented by MELs and links (E) by their interconnections for inter-service communication. Thus, RBMM's R/A model is directly mapped to OsmoticToolkit's T/P model.

OsmoticToolkit extends MaestroNG's YAML-based schema on a technical level by adding support for the decisional factors specified in Table 9.1. OsmoticToolkit associates the concept of a pipeline to an application. Namely, the pipeline's anatomy describes MELs properties and how they are interconnected. Listing 9.2 illustrates with a code fragment how MEL's constraints can be expressed. The logic is implemented in the osmotic orchestrator that uses a two-phase optimization approach to find the most appropriate deployment plan resources for an application. An application's constraints are classified into hard constraints and soft constraints. Hard constraints refer to must-have requirements that persist and are invariant during execution, such as CPU, memory, or latency; soft constraints refer to desired requirements that can change or be re-prioritized the cost of consuming resources.

Listing 9.2: MaestroNG YAML format representation of application factors.

```
ship_provider : dynamic # static
name : p1
# ships :
#  ship1 :
#     ip : x.x.x.x
services :
```

```
foo :
  image : ubuntu
  security_opt : [zone==intranet, vulnerability==backdoor, consistency==
      true] requires: [test]
  labels :
  constraint :
    runtime : python3
    complexity : high
    latency : 5ms
    duration : 900s
  limits :
    memory : 50m
    cpu : 1
  instances :
    foo-1 :
      # ship : ship1
```

The classification of requirements into hard constraints or soft constraints depends on the user's need. For example, network latency can be classified as soft constraints if an application is not latency-sensitive; however, one could classify latency as hard constraints if the application's response time must not exceed a specific threshold limit.

The first phase selects a set of resources by ensuring that all application context-based constraints are satisfied. This is accomplished by executing the RBMM. The second phase balances the complexities of cost and resource-based constraints. This phase involves the Hungarian algorithm. In OsmoticToolkit, optimal deployment is treated as an assignment problem solved using the Hungarian algorithm (see Section 8.5.3 for further details)

9.4.4 Limitations

Although RBMM is among the most user-controllable matchmakers, we summarise the limitations to define future research paths. These are:

- Lack of support for dynamic external context factors and redeployment calculation when runtime factors change;
- No specific consideration of data flows, network links and microservice interconnects as first-class citizens, which would also help OsmoticToolkit achieving more accurate emulation;
- Non-optimized recursive algorithm that may limit large deployments with more than a few hundreds of microservices;
- No integration of the automatic acquisition of resource characteristics.

9.5 Related Works

Matchmaking and mediation have been a traditional research topic in service-oriented systems design, particularly around semantic web services communities

with two decades of history. Early works used heavy semantic modelling, for instance, using the Web Service Modelling Ontology, with the advantage of being able to express al- most arbitrary details while at the same time requiring much effort to maintain and extend the descriptions [232]. Early systems like ConQo enabled matchmaking between service providers and clients [233], and expressive ontologies like WSMO4IoS modelled cloud providers and their service characteristics, although failed to account for the application-side properties [234] and most only worked on single services, not on sets of services. Ontologies were also proposed in alternative approaches for cloud service composition matchmaking [235], but specifically tailored for virtual machines rather than today's variety beyond cloud platforms and assuming existing knowledge repositories. The composition objectives were defined as compatibility, total cost, total deployment time or total reliability.

Recent approaches distinguish text-based matchmaking equivalent to full-text searching [236] and attribute-based matchmaking [237]. However, no matchmaker with factor acquisition specifically designed for heterogeneous multi-target deployment of composite software is known, which is a necessity when automating the management of future computing continuums.

9.6 Conclusions

RBMM performs versatile rule-based matchmaking between composite applications and distributed resources. It advances the state of automation for computing continuums and osmotic processes around IoT, fog, edge and cloud deployments. The RBMM implementation is publicly available as an open-source library [4]. More challenges have to be resolved to make programmable continuums as straightforward and as widely accepted as centralized and programmable infrastructure offered by commercial cloud providers. We point out the open research questions related to the four current system limitations. Additionally, the applications themselves will need to gain more awareness of how they are deployed to adapt and offer higher resilience by replicating data if the deployment topology permits, based on an initial matchmaking goal of prioritizing resilience over other holistic application characteristics.

4 RBMM code: https://github.com/serviceprototypinglab/rbmm

10 Conclusions

This chapter recalls the thesis's context, summarises the contributions and research results and outlines potential future outlining perspectives and future research directions.

10.1 Contributions Summary

The emergence of the Internet of Things (IoT) has led to infrastructure development that extends beyond centralized data centres from the cloud to the edge, the so-called Cloud-to-Thing (C2T) continuum. This infrastructure is characterized by extreme heterogeneity, geographic distribution, and complexity, where the Key Performance Indicators (KPIs) for the traditional model of Cloud Computing (CC) may no longer apply in the same way.

To understand the C2T requirements, we face one of the major challenges of the C2T continuum, which is resource management. In the quest to assist this, we faced resource management by decoupling user data and applications management (i) from networking (ii) and security (iii) management. Hence, we focused on each of those aspects and formulated the RQs covering the whole IoT application's lifecycle, i.e., development, deployment, execution, management, and orchestration.

(i) is addressed by proposing a new type of service model for IoT applications into the C2T continuum (see Chapter 2). We presented three motivating use-cases; an IoTaaS for Public Safety and Disaster Response, a smart home scenario that proposes a smart meter IoTaaS consisting of a Fast Fourier Transform (FFT)-based microservice, and an IoTaaS Kinect-based gait assessment for Ataxia) as an application example and show the potential benefits of the continuums.

Network management (iii) leveraging SDN and NFV is addressed in Chapter 3, while security management (iv) leveraging blockchain technologies is faced in Chapter 4.

We determined a key concern with using computing models to support IoT applications is managing different physical and virtual infrastructures (e.g., edge devices, and IoT devices) according to specific application and service requirements (e.g., latency, data volume, responsivity, and processing delays). However, these models address specific application issues and often coexist or need to co-operate. The coexistence of these computing paradigms in the same application scenario can be hard to manage, and it requires additional services to support interoperability and service management.

In this context, Osmotic Computing (OC) motivated by the lack of a scalable, interoperable, configurable solution for delivering IoT applications in the complex, heterogeneous, and dynamic C2T continuum, addresses issues related to deployment, networking, and security of microservices, called MicroELements (MELs), that are composed and interconnected over both cloud/edge and IoT infrastructures with specified levels of QoS and security constraints.

Nevertheless, the usage of an OC poses new challenges for IoT workflow application developers and operations managers as they need the awareness of resource/device (cloud vs IoT gateway) heterogeneity, virtualization software heterogeneity (e.g., hypervisor vs container), data analytic programming model heterogeneity (stream processing vs batch processing), geographic distribution, and network performance uncertainties. The seamless orchestration and operation of such an infrastructure are challenging due to the involved devices' heterogeneity and network connectivity. Accordingly, there is no established solution available yet. In the quest to assist this, we derived the RQs and brought the following contributions.

Part II investigates how IoT applications into the C2T continuum can benefit from OC. Basic concepts, methodologies, key technologies behind OC ecosystems are introduced in Chapter 5. There is also presented an efficient orchestration approach to redeploy containerized microservices eliminating the application outage and promptly reacting to failures. A careful analysis highlights the advantages introduced by our solution. Finally, Chapter 7 and Chapter 6 proposed two use case scenarios illustrating the benefits of applying OC in the continuums.

Part III contributes with a novel, cost-effective and flexible toolkit, called OsmoticToolkit, for the from-scratch design of OC ecosystems and real-world emulation applications. The toolkit offers support for hybrid cloud-edge/fog-IoT architecture, dynamic infrastructure topology (modelling of physical networks and virtual topologies), resource provisioning approach, real application execution, well-defined usage procedure with explicit infrastructure and application modelling, infrastructure instantiation and application pipeline deployment and service-oriented integration with APIs and CLIs to fit into automated osmotic and cloud-native environments. In contrast to the streamlined onboarding of software applications in clouds, the decomposition and description of applications for such dynamic scenarios is currently a challenging engineering task. To give application engineers the ability to prepare, using an emulator will save precious engineering time and effort and facilitate resource planning.

Part IV contributes to implementing a rule-based matchmaker called RBMM that combines several decision factors for supporting applications deployment in the C2T continuum. This chapter defines application and resource models, then categorizes the associated decision factors in the RBMM model. To increase automation, we present suitable acquisition techniques for each, complemented by propagation, skipping, deployment and accumulation rules. Subsequently, we

present simple matchmaking algorithms to yield suitable deployment topologies. To increase the work's applicability, we implement the matchmaking and a composite application deployment scenario where the OsmoticToolkit is considered a practical demonstrator.

10.2 Future Directions

This thesis can be extended in multiple directions. Each chapter outlines its future directions, of which some are related to or overlapped with others. The field of OC holds many promises for future research, especially for microservice deployment and orchestration in the C2T continuum. Below we describe the detailed potentialities of this field for future extensions.

The OsmoticToolkit proposed in Chapter 8 might be extended in the following directions:

- *Efficient scheduling algorithms*: the scheduling algorithm implemented for the MELs placement inside the Osmotic Computing ecosystem is based on Kuhn's Hungarian algorithm that scheduling as an Optimal Assignment Problem (OAP) to efficiently solve an $n \times n$ task assignment problem in $O(n^3)$ time. This complexity renders the procedure unsuitable for complex applications. To overcome this limitation, a more efficient algorithm for application scheduling can be investigated. Also, advanced scheduling approaches based on genetic algorithms would also be beneficial to be investigated.

- *Geographical edge/fog nodes placement*: the nodes' placememnt solution provided in placements does not consider the nodes' geographical coordinates. When considering the Edge Computing (EC) distributed nature and the high number of nodes deployed, capital and operational expenses (CAPEX and OPEX) become critical concerns. Thus, the strategy for the placement of the edge/fog nodes becomes crucial. By optimizing the edge/-fog nodes site selection process, the deployment/operation cost savings can be substantially increased. However, the placement scheme comes to be challenging due to the numerous trade-offs involved. At first glance, minimizing the number of edge nodes would reduce global expenditures but at the cost of a decreased performance or unmet demands (e.g., latency, reliability). Another challenge is the edge/fog nodes capacity allocation and site rentals trade-off. On a site with low location-dependent costs, it is more efficient to put as much capacity as possible, but this strategy would not comply with the service demand distribution. Also, in some cases, it is preferred to reuse existent Information Technology (IT)-capable locations to reduce initial deployment expenses, but this could be disadvantageous in the mid to long-term. If the forecasted demand surge occurs away from

such sites, choosing locations closer to the user demands, even if they do not have IT capabilities, will imply a higher initial CAPEX but lower overall costs once deployed.

- *Cluster federation*: with the research presented in this thesis, it is possible to provide a cluster of compute nodes managed by an organization. However, there may be a case that one workload requires to be run on the compute resources shared between two organizations. In such scenarios, a federation must be implemented between the two organizations and a customized service model could be provided. It would be valuable to investigate how such a federation can be created to utilize multiple compute nodes.

- *Dymanic network segmentation management*: the proposed tool does not provide granular control on the network slices management. Through network segmentation mechanisms, it is possible to minimize an intruder's movements across the network and controls user access to specific applications and data. Further investigation in this field would allow implementing different Software Defined Membranes (SDMems), each with fine-grained control over security, with policies based upon application and workload requirements rather than rigid network addressing schemes.

Although OC has gained significant attention over the last few years, it is still in its infancy. There will be more opportunities for OC research in the following years before OC becomes mainstream.

To conclude, this thesis investigated how OC can be leveraged to achieve secure and dependable microservices orchestration in the C2T continuum, where deployment and orchestration strategies depend on IoT applications' specific requirements and physical/virtual resources availability. This thesis also demonstrates that computing as a utility can be made available outside large data centres, which is a promising implication for future clouds to use resources in the C2T continuum.

Bibliography

[1] Cisco. *White Paper - Internet of things at a glance*. Last accessed on 28 December 2020. URL: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/iot-aag.pdf (see page 1).

[2] Ericsson. **Mobility Report - q4, 2018**. Tech. rep. Last accessed on 28 December 2020. URL: https://www.ericsson.com/4932c2/assets/local/mobility-report/documents/2019/emr-q4-update-2018.pdf (see page 1).

[3] Philbert Shih Jim Davis and Alex Marcham. **An Edge Computing Ecosystem Report**. Tech. rep. Last accessed on 28 December 2020. URL: https://www.stateoftheedge.com/ (see page 1).

[4] M. Satyanarayanan. **The Emergence of Edge Computing**. *Computer* 50 (2017), 30–39 (see page 1).

[5] John Fredette, Revital Marom, K Steiner, and Louis Witters. **The promise and peril of hyperconnectivity for organizations and societies**. *The global information technology report* 2012 (2012), 113–119 (see page 1).

[6] Beniamino Di Martino, Kuan-Ching Li, Laurence T Yang, and Antonio Esposito. **Internet of everything: Algorithms, methodologies, technologies and perspectives**. Springer, 2017 (see page 1).

[7] H. F. Atlam, A. Alenezi, A. Alharthi, R. J. Walters, and G. B. Wills. **Integration of Cloud Computing with Internet of Things: Challenges and Open Issues**. In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2017, 670–675. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.105 (see page 1).

[8] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos. **Security and Privacy for Cloud-Based IoT: Challenges**. *IEEE Communications Magazine* 55:1 (2017), 26–33. DOI: 10.1109/MCOM.2017.1600363CM (see page 1).

[9] Mingchen Zhao, Paarijaat Aditya, Ang Chen, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, Bill Wishon, and Miroslav Ponec. **Peer-Assisted Content Distribution in Akamai Netsession**. In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain: Association for Computing Machinery, 2013, 31–42. ISBN: 9781450319539. DOI: 10.1145/2504730.2504752. URL: https://doi.org/10.1145/2504730.2504752 (see page 2).

[10] Long Vu, Indranil Gupta, Klara Nahrstedt, and Jin Liang. **Understanding Overlay Characteristics of a Large-Scale Peer-to-Peer IPTV System**. *ACM Trans. Multimedia Comput. Commun. Appl.* 6:4 (Nov. 2010). ISSN: 1551-6857. DOI: 10.1145/1865106.1865115. URL: https://doi.org/10.1145/1865106.1865115 (see page 2).

[11] 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*. MCC '12. Helsinki, Finland: Association for Computing Machinery, 2012, 13–16. ISBN: 9781450315197. DOI: 10.1145/2342509.2342513. URL: https://doi.org/10.1145/2342509.2342513 (see page 2).

[12] OpenFog Consortium. **Open fog architecture overview**. Tech. rep. Last accessed on 28 December 2020 (see page 2).

[13] Raka Mahesa. *How Cloud, Fog and Mist Computing can work together*. Last accessed on 28 December 2020. URL: https://developer.ibm.com/dwblog/2018/cloud-fog-mist-edge-computing-iot/ (see page 2).

[14] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, and Rajiv Ranjan. **Osmotic Computing: A New Paradigm for Edge/Cloud Integration**. *IEEE Cloud Computing* 3:6 (2016), 76–83. DOI: 10.1109/mcc.2016.124. URL: https://doi.org/10.1109/mcc.2016.124 (see pages 3, 157).

[15] Malika Bendechache, Sergej Svorobej, P. Endo, and Theo Lynn. **Simulating Resource Management across the Cloud-to-Thing Continuum: A Survey and Future Directions**. *Future Internet* 12 (2020), 95 (see page 4).

[16] W. Yu, H. Xu, J. Nguyen, E. Blasch, A. Hematian, and W. Gao. **Survey of Public Safety Communications: User-Side and Network-Side Solutions and Future Directions**. *IEEE Access* 6 (2018), 70397–70425 (see page 13).

[17] Marius Portmann. **Wireless Mesh Networks for Public Safety and Disaster Recovery Applications** (Dec. 2006). DOI: 10.1201/9781420013542.ch16 (see page 14).

[18] M. Portmann and A. A. Pirzada. **Wireless Mesh Networks for Public Safety and Crisis Management Applications**. *IEEE Internet Computing* 12:1 (2008), 18–25. ISSN: 1089-7801. DOI: 10.1109/MIC.2008.25 (see page 14).

[19] Q. T. Minh, K. Nguyen, and S. Yamada. **DRANs: Resilient Disaster Recovery Access Networks**. In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. 2013, 754–759. DOI: 10.1109/COMPSACW.2013.88 (see page 14).

[20] Minh Quang Tran, Yoshitaka Shibata, Cristian Borcea, and Shigeki Yamada. **On-site configuration of disaster recovery access networks made easy**. *Ad Hoc Networks* 40 (Jan. 2016). DOI: 10.1016/j.adhoc.2015.12.008 (see page 14).

[21] A. Yarali, B. Ahsant, and S. Rahman. **Wireless Mesh Networking: A Key Solution for Emergency Rural Applications**. In: *2009 Second International Conference on Advances in Mesh Networks*. 2009, 143–149. DOI: 10.1109/MESH.2009.33 (see pages 14, 15).

[22] J. Burchard, D. Chemodanov, J. Gillis, and P. Calyam. **Wireless Mesh networking Protocol for sustained throughput in edge computing**. In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. 2017, 958–962. DOI: 10.1109/ICCNC.2017.7876263 (see page 14).

[23] Daniel Gutiérrez, S.L. Toral, Federico Barrero, Nik Bessis, and Eleana Asimakopoulou. **Evaluation of Ad Hoc Networks in Disaster Scenarios**. In: Nov. 2011, 759–764. DOI: 10.1109/INCoS.2011.86 (see page 14).

[24] A. Khan, F. Aftab, and Z. Zhang. **UAPM: An urgency-aware packet management for disaster management using flying ad-hoc networks**. *China Communications* 16:11 (2019), 167–182 (see page 14).

[25] W. Zhao, W. Xin, X. Zheng, and T. Hara. **Comparison Study On UAV Movement for Adapting to Multimedia Burst in Post-Disaster Networks**. In: *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2018, 339–343 (see page 15).

[26] J. R. E. Leite, P. S. Martins, and E. L. Ursini. **Planning of AdHoc and IoT Networks Under Emergency Mode of Operation**. In: *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2019, 1071–1080 (see page 15).

[27] A. I. Husain and H. Bhardwaj. **A Cluster based WSN for Earthquake and Tsunami: Detection and Mitigation**. In: *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. 2019, 847–849 (see page 15).

[28] Noman Islam, Ghazala Sheikh, and Zeeshan Islam. **A cognitive radio ad hoc network's based disaster management scheme with efficient spectrum management, collaboration and interoperability**. *ITB Journal of Information and Communication Technology* 10 (Jan. 2017) (see page 15).

[29] P. K. Dalela, S. Sachdev, and V. Tyagi. **LoRaWAN Network Capacity for Practical Network Planning in India**. In: *2019 URSI Asia-Pacific Radio Science Conference (AP-RASC)*. 2019, 1–4 (see page 15).

[30] R. Murugeswari and S. Radhakrishnan. **Reliable data delivery for emergency and disaster recovery in wireless mesh network**. In: *International Conference on Information Communication and Embedded Systems (ICICES2014)*. 2014, 1–6. DOI: 10.1109/ICICES.2014.7033950 (see page 15).

[31] Z. Lu, G. Cao, and T. L. Porta. **Networking smartphones for disaster recovery**. In: *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2016, 1–9 (see page 15).

[32] M. Portmann and A. A. Pirzada. **Wireless Mesh Networks for Public Safety and Crisis Management Applications**. *IEEE Internet Computing* 12:1 (2008), 18–25. ISSN: 1089-7801. DOI: 10.1109/MIC.2008.25 (see page 15).

[33] T. Ngo, H. Nishiyama, N. Kato, Y. Shimizu, K. Mizuno, and T. Kumagai. **On the throughput evaluation of wireless mesh network deployed in disaster areas**. In: *2013 International Conference on Computing, Networking and Communications (ICNC)*. 2013, 413–417. DOI: 10.1109/ICCNC.2013.6504119 (see page 15).

[34] G. Iapichino, C. Bonnet, O. del Rio Herrero, C. Baudoin, and I. Buret. **A mobile ad-hoc satellite and wireless Mesh networking approach for Public Safety communications**. In: *2008 10th International Workshop on Signal Processing for Space Communications*. 2008, 1–6. DOI: 10.1109/SPSC.2008.4686695 (see page 15).

[35] Q. T. Minh, K. Nguyen, C. Borcea, and S. Yamada. **On-the-fly establishment of multihop wireless access networks for disaster recovery**. *IEEE Communications Magazine* 52:10 (2014), 60–66. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6917403 (see page 15).

[36] *Golden hour (medicine)*. [Online];Accessed: 2018-03-28. URL: http://en.wikipedia.org/wiki/Golden_hour_(medicine) (see page 16).

[37] M. Villari, G. Tricomi, A. Celesti, and M. Fazio. **Orchestration for the deployment of distributed applications with geographical constraints in cloud federation**. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST* 189 (2018), 177–187 (see page 16).

[38] E. M. Royer and. **A review of current routing protocols for ad hoc mobile wireless networks**. *IEEE Personal Communications* 6:2 (1999), 46–55. ISSN: 1070-9916. DOI: 10.1109/98.760423 (see page 18).

[39] L. Carnevale, A. Galletta, M. Fazio, A. Celesti, and M. Villari. **Designing a FIWARE Cloud Solution for Making Your Travel Smoother: The FLIWARE Experience**. In: *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. 2018, 392–398. DOI: 10.1109/CIC.2018.00059 (see page 19).

[40] Zhenyu Yang, Chandrakanth Chereddi, and Haiyun Luo. **Bandwidth measurement in wireless mesh networks**. *Course Project Report, URL: http://www.crhc. uiuc. edu/ cchered2/pubs. html (checked 2005-05-23)* (2007) (see page 24).

[41] K. Roy, V.K. Banninthaya, S.M. Prabhu, A. Koomar, D. Karnataki, and G. Shankar. **Smart IoT based Energy Metering System for Microgrids with Load Management Algorithm**. In: 2018, 252–256 (see page 32).

[42] A. Berouine, F. Lachhab, Y. Nait Malek, M. Bakhouya, and R. Ouladsine. **A smart metering platform using big data and IoT technologies**. In: vol. 2018-January. 2018, 1–6 (see page 32).

[43] L. Vangelista. **Frequency Shift Chirp Modulation: The LoRa Modulation**. *IEEE Signal Processing Letters* 24:12 (2017), 1818–1821. ISSN: 1070-9908. DOI: 10.1109/LSP.2017.2762960 (see page 32).

[44] M. Pennacchioni, M.-G. Di Benedette, T. Pecorella, C. Carlini, and P. Obino. **NB-IoT system deployment for smart metering: Evaluation of coverage and capacity performances**. In: vol. 2017-January. 2017, 1–6 (see page 32).

[45] R. Ramakrishnan and L. Gaur. **Smart electricity distribution in residential areas: Internet of Things (IoT) based advanced metering infrastructure and cloud analytics**. In: 2016, 46–51 (see page 32).

[46] R. Bhilare and S. Mali. **IoT based smart home with real time E-metering using E-controller**. In: 2016 (see page 32).

[47] Qi-Lin Mao and Ming-Yue Zhai. **A new grid frequency estimation algorithm based on the fractional FFT for IoT nodes time stamps**. *Cluster Computing* (2018). ISSN: 1573-7543. DOI: 10.1007/s10586-017-1653-2. URL: https://doi.org/10.1007/s10586-017-1653-2 (see page 32).

[48] Y. Lu and T. J. Kazmierski. **An ultra-low-power variable-accuracy bit-serial FFT butterfly processing element for IoT sensors**. In: *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 2016, 13–16. DOI: 10.1109/APCCAS.2016.7803883 (see page 32).

[49] D.-H. Lee and I.-Y. Lee. **Dynamic group authentication and key exchange scheme based on threshold secret sharing for IoT smart metering environments**. *Sensors (Switzerland)* 18:10 (2018) (see page 32).

[50] M. Cebe and K. Akkaya. **Efficient certificate revocation management schemes for IoT-based advanced metering infrastructures in smart cities**. *Ad Hoc Networks* (2018) (see page 32).

[51] S. Tonyali, K. Akkaya, N. Saputro, A.S. Uluagac, and M. Nojoumian. **Privacy-preserving protocols for secure and reliable data aggregation in IoT-enabled Smart Metering systems**. *Future Generation Computer Systems* 78 (2018), 547–557 (see page 32).

[52] *Students are Mining Bitcoin in University Dorms, Campuses are Alert*. https://www.newsbtc.com/2018/08/12/students-are-mining-bitcoin-in-university-dorms-campuses-are-alert/. [Online]; Accessed: 2019-03-28 (see page 39).

[53] Rick Brandsma, Tjitske F Lawerman, Marieke J Kuiper, Roelineke J Lunsing, Huibert Burger, and Deborah A Sival. **Reliability and discriminant validity of ataxia rating scales in early onset ataxia**. *Developmental Medicine and Child Neurology* 59:4 (2017), 427–432. DOI: https://doi.org/10.1111/dmcn.13291. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/dmcn.13291. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/dmcn.13291 (see page 45).

[54] David R. Lynch, Ashley McCormick, Kimberly Schadt, and Elizabeth Kichula. **Pediatric Ataxia: Focus on Chronic Disorders**. *Seminars in Pediatric Neurology* 25 (2018). Pediatric Movement Disorders Movements, 54 –64. ISSN: 1071-9091. DOI: https://doi.org/10.1016/j.spen.2018.01.001. URL: http://www.sciencedirect.com/science/article/pii/S1071909118300019 (see page 45).

[55] Enrico Bertini, Ginevra Zanni, and Eugen Boltshauser. "Chapter 6 - Non-progressive congenital ataxias." In: *The Cerebellum: Disorders and Treatment*. Ed. by Mario Manto and Thierry A.G.M. Huisman. Vol. 155. Handbook of Clinical Neurology. Elsevier, 2018, 91 –103. DOI: https://doi.org/10.1016/B978-0-444-64189-2.00006-8. URL: http://www.sciencedirect.com/science/article/pii/B9780444641892000068 (see page 45).

[56] Matthis Synofzik and Andrea H. Németh. "Chapter 5 - Recessive ataxias." In: *The Cerebellum: Disorders and Treatment*. Ed. by Mario Manto and Thierry A.G.M. Huisman. Vol. 155. Handbook of Clinical Neurology. Elsevier, 2018, 73 –89. DOI: https://doi.org/10.1016/B978-0-444-64189-2.00005-6. URL: http://www.sciencedirect.com/science/article/pii/B9780444641892000056 (see page 45).

[57] Piero Pavone, Andrea Pratico, Vito Pavone, Riccardo Lubrano, Raffaele Falsaperla, Renata Rizzo, and Martino Ruggieri. **Ataxia in children: Early recognition and clinical evaluation**. *Italian Journal of Pediatrics* 43 (Jan. 2017). DOI: 10.1186/s13052-016-0325-9 (see page 45).

[58] Tommaso Schirinzi, Andrea Sancesario, Enrico Bertini, Enrico Castelli, and Gessica Vasco. **Speech and Language Disorders in Friedreich Ataxia: Highlights on Phenomenology, Assessment, and Therapy**. *The Cerebellum* 19 (Feb. 2020). DOI: 10.1007/s12311-019-01084-8 (see page 45).

[59] Tommaso Schirinzi, Martina Favetta, Alberto Romano, Andrea Sancesario, Susanna Summa, Silvia Minosse, Zanni Ginevra, Enrico Castelli, Enrico Bertini, Maurizio Petrarca, and Gessica Vasco. **One-year outcome of coenzyme Q10 supplementation in ADCK3 ataxia (ARCA2)**. *Cerebellum and Ataxias* 6 (Dec. 2019). DOI: 10.1186/s40673-019-0109-2 (see page 45).

[60] Martin B. Delatycki and Sanjay I. Bidichandani. **Friedreich ataxia- pathogenesis and implications for therapies**. *Neurobiology of Disease* 132 (2019), 104606. ISSN: 0969-9961. DOI: https://doi.org/10.1016/j.nbd.2019.104606. URL: http://www.sciencedirect.com/science/article/pii/S0969996119302815 (see page 45).

[61] Alberto Benussi, Alvaro Pascual-Leone, and Barbara Borroni. **Non-Invasive Cerebellar Stimulation in Neurodegenerative Ataxia: A Literature Review**. *International Journal of Molecular Sciences* 21 (Mar. 2020), 1948. DOI: 10.3390/ijms21061948 (see page 45).

[62] T. Schmitz-Hübsch, S. Tezenas du Montcel, L. Baliko, J. Berciano, S. Boesch, C. Depondt, P. Giunti, C. Globas, J. Infante, J. S. Kang, B. Kremer, C. Mariotti, B. Melegh, M. Pandolfo, M. Rakowicz, P. Ribai, R. Rola, L. Schöls, S. Szymanski, B. P. van de Warrenburg, A. Dürr, and T. Klockgether. **Scale for the assessment and rating of ataxia**. *Neurology* 66:11 (2006), 1717–1720. ISSN: 0028-3878. DOI: 10.1212/01.wnl.0000219042.60538.92. eprint: https://n.neurology.org/content/66/11/1717.full.pdf. URL: https://n.neurology.org/content/66/11/1717 (see page 45).

[63] Marco Germanotta, Gessica Vasco, Maurizio Petrarca, Stefano Rossi, Sacha Carniel, Enrico Bertini, Paolo Cappa, and Enrico Castelli. **Robotic and clinical evaluation of upper limb motor performance in patients with Friedreich's Ataxia: An observational study**. *Journal of neuroengineering and rehabilitation* 12 (Apr. 2015), 41. DOI: 10.1186/s12984-015-0032-6 (see page 45).

[64] Tjitske F Lawerman, Rick Brandsma, Huibert Burger, Johannes G M Burgerhof, Deborah A Sival, and the Childhood Ataxia and Cerebellar Group of the European Pediatric Neurology Society. **Age-related reference values for the pediatric Scale for Assessment and Rating of Ataxia: a multicentre study**. *Developmental Medicine & Child Neurology* 59:10 (2017), 1077–1082. DOI: https://doi.org/10.1111/dmcn.13507. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/dmcn.13507. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/dmcn.13507 (see page 45).

[65] Rick Brandsma, Anne H Spits, Marieke J Kuiper, Roelinka J Lunsing, Huibert Burger, Hubertus P Kremer, Deborah A Sival, and The Childhood Ataxia and Cerebellar Group. **Ataxia rating scales are age-dependent in healthy children**. *Developmental Medicine & Child Neurology* 56:6 (2014), 556–563. DOI: https://doi.org/10.1111/dmcn.12369. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/dmcn.12369. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/dmcn.12369 (see page 45).

[66] Karen Otte, Bastian Kayser, Sebastian Mansow-Model, Julius Verrel, Friedemann Paul, Alexander U. Brandt, and Tanja Schmitz-Hübsch. **Accuracy and Reliability of the Kinect Version 2 for Clinical Measurement of Motor Function**. *PLOS ONE* 11:11 (Nov. 2016), 1–17. DOI: 10.1371/journal.pone.0166532. URL: https://doi.org/10.1371/journal.pone.0166532 (see page 45).

[67] Lazzaro Di Biase, Susanna Summa, Jacopo Tosi, Fabrizio Taffoni, Massimo Marano, Angelo Cascio Rizzo, Fabrizio Vecchio, Domenico Formica, Vincenzo Di Lazzaro, Giovanni Di Pino, and Mario Tombini. **Quantitative Analysis of Bradykinesia and Rigidity in Parkinson's Disease**. *Frontiers in Neurology* 9 (Mar. 2018), 121. DOI: 10.3389/fneur.2018.00121 (see page 45).

[68] Susanna Summa, Angelo Basteris, Enrico Betti, and Vittorio Sanguineti. **Adaptive training with full-body movements to reduce bradykinesia in persons with Parkinson's disease: A pilot study**. *Journal of NeuroEngineering and Rehabilitation* 12 (Dec. 2015), 16. DOI: 10.1186/s12984-015-0009-5 (see page 45).

[69] Lazzaro di Biase, Susanna Summa, Jacopo Tosi, Fabrizio Taffoni, Massimo Marano, Angelo Cascio Rizzo, Fabrizio Vecchio, Domenico Formica, Vincenzo Di Lazzaro, Giovanni Di Pino, and Mario Tombini. **Quantitative Analysis of Bradykinesia and Rigidity in Parkinson's Disease**. *Frontiers in Neurology* 9 (2018), 121. ISSN: 1664-2295. DOI: 10.3389/fneur.2018.00121. URL: https://www.frontiersin.org/article/10.3389/fneur.2018.00121 (see page 45).

[70] J. Tosi, S. Summa, F. Taffoni, L. d. Biase, M. Marano, A. C. Rizzo, M. Tombini, E. Schena, D. Formica, and G. D. Pino. **Feature Extraction in Sit-to-Stand Task Using M-IMU Sensors and Evaluatiton in Parkinson's Disease**. In: *2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. 2018, 1–6. DOI: 10.1109/MeMeA.2018.8438737 (see page 45).

[71] Bruno Bonnechère, Bart Jansen, Inès Haack, Lubos Omelina, Veronique Feipel, Serge Van Sint Jan, and Massimo Pandolfo. **Automated functional upper limb evaluation of patients with Friedreich ataxia using serious**

games rehabilitation exercises. *Journal of NeuroEngineering and Rehabilitation* 15 (Oct. 2018). DOI: 10.1186/s12984-018-0430-7 (see page 45).

[72] Bekkers EMJ Van den Bergh V Ginis P Rochester L Hausdorff JM Mirelman A Dockx K and A Nieuwboer. **Virtual reality for rehabilitation in Parkinson's disease**. *Cochrane Database of Systematic Reviews*: 12 (2016). ISSN: 1465-1858. DOI: 10.1002/14651858.CD010760.pub2. URL: https://doi.org//10.1002/14651858.CD010760.pub2 (see pages 45, 58).

[73] Jamie Taylor and Kevin Curran, 183–192. In: Jan. 2015. DOI: 10.4018/978-1-4666-9522-1.ch009 (see page 45).

[74] H. Tran, P. N. Pathirana, M. Horne, L. Power, and D. Szmulewicz. **Automated Finger Chase (ballistic tracking) in the Assessment of Cerebellar Ataxia**. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018, 3521–3524. DOI: 10.1109/EMBC.2018.8513036 (see page 45).

[75] Anuschka Grobelny, Janina R. Behrens, Sebastian Mertens, Karen Otte, Sebastian Mansow-Model, Theresa Krüger, Elona Gusho, Judith Bellmann-Strobl, Friedemann Paul, Alexander U. Brandt, and Tanja Schmitz-Hübsch. **Maximum walking speed in multiple sclerosis assessed with visual perceptive computing**. *PLOS ONE* 12:12 (Dec. 2017), 1–13. DOI: 10.1371/journal.pone.0189281. URL: https://doi.org/10.1371/journal.pone.0189281 (see page 45).

[76] Susanna Summa, Tommaso Schirinzi, Giuseppe Massimo Bernava, Alberto Romano, Martina Favetta, Enza Maria Valente, Enrico Bertini, Enrico Castelli, Maurizio Petrarca, Giovanni Pioggia, and Gessica Vasco. **Development of SaraHome: A novel, well-accepted, technology-based assessment tool for patients with ataxia**. *Computer Methods and Programs in Biomedicine* 188 (2020), 105257. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2019.105257. URL: http://www.sciencedirect.com/science/article/pii/S0169260719315408 (see pages 45, 57).

[77] Ellen Buckley, Claudia Mazza, and Alisdair McNeill. **A systematic review of the gait characteristics associated with Cerebellar Ataxia**. *Gait and Posture* 60 (2018), 154 –163. ISSN: 0966-6362. DOI: https://doi.org/10.1016/j.gaitpost.2017.11.024. URL: http://www.sciencedirect.com/science/article/pii/S0966636217310238 (see page 46).

[78] Gessica Vasco, Simone Gazzellini, Maurizio Petrarca, Maria Luisa Lispi, Alessandra Pisano, Marco Zazza, Gessica Della Bella, Enrico Castelli, and Enrico Bertini. **Functional and Gait Assessment in Children and Adolescents Affected by Friedreich's Ataxia: A One-Year Longitudinal Study**. *PLOS ONE* 11:9 (Sept. 2016), 1–13. DOI: 10.1371/journal.pone.0162463. URL: https://doi.org/10.1371/journal.pone.0162463 (see pages 46, 57).

[79] Winfried Ilg, Zofia Fleszar, Cornelia Schatton, Holger Hengel, Florian Harmuth, Peter Bauer, Dagmar Timmann, Martin Giese, Ludger Schols, and Matthis Synofzik. **Individual changes in preclinical spinocerebellar ataxia identified via increased motor complexity**. *Movement Disorders* 31:12 (2016), 1891–1900. DOI: https://doi.org/10.1002/mds.26835. eprint: https:

//onlinelibrary.wiley.com/doi/pdf/10.1002/mds.26835. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mds.26835 (see page 46).

[80] Lynn Rochester, Brook Galna, Sue Lord, Dadirayi Mhiripiri, Gail Eglon, and Patrick F. Chinnery. **Gait impairment precedes clinical symptoms in spinocerebellar ataxia type 6**. *Movement Disorders* 29:2 (2014), 252–255. DOI: https://doi.org/10.1002/mds.25706. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mds.25706. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mds.25706 (see page 46).

[81] B. Munoz, Y. J. Castano-Pino, J. David Arango Paredes, and A. Navarro. **Automated Gait Analysis using a Kinect Camera and Wavelets**. In: *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*. 2018, 1–5. DOI: 10.1109/HealthCom.2018.8531161 (see pages 46, 56).

[82] A.Nandy and P. Chakraborty. **A new paradigm of human gait analysis with Kinect**. In: *2015 Eighth International Conference on Contemporary Computing (IC3)*. 2015, 443–448. DOI: 10.1109/IC3.2015.7346722 (see pages 46, 56).

[83] Ana Patricia Rocha, Hugo Miguel Pereira Choupina, Maria do Carmo Vilas-Boas, Jose Maria Fernandes, and Joao Paulo Silva Cunha. **System for automatic gait analysis based on a single RGB-D camera**. *PLOS ONE* 13:8 (Aug. 2018), 1–24. DOI: 10.1371/journal.pone.0201728. URL: https://doi.org/10.1371/journal.pone.0201728 (see pages 46, 56).

[84] Yunru Ma, Kumar Mithraratne, Nichola Wilson, Xiangbin Wang, Ye Ma, and Yanxin Zhang. **The Validity and Reliability of a Kinect v2-Based Gait Analysis System for Children with Cerebral Palsy**. *Sensors* 19 (Apr. 2019). DOI: 10.3390/s19071660 (see pages 46, 56).

[85] Ross A Clark, Stephanie Vernon, Benjamin F Mentiplay, Kimberly J Miller, Jennifer L McGinley, Yong Hao Pua, Kade Paterson, and Kelly J Bower. **Instrumenting gait assessment using the Kinect in people living with stroke: reliability and association with balance tests**. *Journal of neuroengineering and rehabilitation* 12 (2015), 15. ISSN: 1743-0003. DOI: 10.1186/s12984-015-0006-8. URL: https://europepmc.org/articles/PMC4333881 (see pages 46, 47, 56).

[86] J.A. Zeni, J.G. Richards, and J.S. Higginson. **Two simple methods for determining gait events during treadmill and overground walking using kinematic data**. *Gait and Posture* 27:4 (2008), 710 –714. ISSN: 0966-6362. DOI: https://doi.org/10.1016/j.gaitpost.2007.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0966636207001804 (see pages 46, 56).

[87] Björn Müller, Winfried Ilg, Martin Giese, and Nicolas Ludolph. **Validation of enhanced kinect sensor based motion capturing for gait assessment**. *PLOS ONE* 12 (Apr. 2017). DOI: 10.1371/journal.pone.0175813 (see pages 46, 56).

[88] Mohammad Ali Mansournia, Rachel Waters, Maryam Nazemipour, Martin Bland, and Douglas G. Altman. **Bland-Altman methods for comparing methods of measurement and response to criticisms**. *Global Epidemiology* 3 (2021), 100045. ISSN: 2590-1133. DOI: https://doi.org/10.1016/j.gloepi.2020.100045. URL: http://www.sciencedirect.com/science/article/pii/S2590113320300298 (see page 51).

[89] Jan S. Krouwer. **Why Bland–Altman plots should use X, not (Y+X)/2 when X is a reference method**. *Statistics in Medicine* 27:5 (2008), 778–780. DOI: https://doi.org/10.1002/sim.3086. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.3086. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.3086 (see page 51).

[90] J. Bland and D. Altman. **Measuring agreement in method comparison studies**. *Statistical Methods in Medical Research* 8 (1999), 135 –160 (see page 51).

[91] Arie Ben-David. **Comparison of classification accuracy using Cohen's Weighted Kappa**. *Expert Systems with Applications* 34:2 (2008), 825 –832. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2006.10.022. URL: http://www.sciencedirect.com/science/article/pii/S0957417406003435 (see page 51).

[92] Richard O Duda, Peter E Hart, et al. **Pattern classification**. John Wiley and Sons, 2006 (see page 51).

[93] **A Note on Distance-Weighted k-Nearest Neighbor Rules**. *IEEE Transactions on Systems, Man, and Cybernetics* 8:4 (1978), 311–313. DOI: 10.1109/TSMC.1978.4309958 (see page 51).

[94] Ron Kohavi. **A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection**. In: Morgan Kaufmann, 1995, 1137–1143 (see page 52).

[95] A. Salarian, H. Russmann, F. J. G. Vingerhoets, C. Dehollain, Y. Blanc, P. R. Burkhard, and K. Aminian. **Gait assessment in Parkinson's disease: toward an ambulatory system for long-term monitoring**. *IEEE Transactions on Biomedical Engineering* 51:8 (2004), 1434–1443. DOI: 10.1109/TBME.2004.827933 (see page 55).

[96] Fay Horak, Laurie King, and Martina Mancini. **Role of Body-Worn Movement Monitor Technology for Balance and Gait Rehabilitation**. *Physical Therapy* 95:3 (Mar. 2015), 461–470. ISSN: 0031-9023. DOI: 10.2522/ptj.20140253. eprint: https://academic.oup.com/ptj/article-pdf/95/3/461/31636529/ptj0461.pdf. URL: https://doi.org/10.2522/ptj.20140253 (see page 55).

[97] John Ludbrook. **SPECIAL ARTICLE COMPARING METHODS OF MEASUREMENT**. *Clinical and Experimental Pharmacology and Physiology* 24:2 (1997), 193–203. DOI: https://doi.org/10.1111/j.1440-1681.1997.tb01807.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1440-1681.1997.tb01807.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1440-1681.1997.tb01807.x (see page 56).

[98] Benjamin F. Mentiplay, Luke G. Perraton, Kelly J. Bower, Yong-Hao Pua, Rebekah McGaw, Sophie Heywood, and Ross A. Clark. **Gait assessment using the Microsoft Xbox One Kinect: Concurrent validity and inter-day reliability of spatiotemporal and kinematic variables**. *Journal of Biomechanics* 48:10 (2015), 2166 –2170. ISSN: 0021-9290. DOI: https://doi.org/10.1016/j.jbiomech.2015.05.021. URL: http://www.sciencedirect.com/science/article/pii/S0021929015002985 (see page 56).

[99] Masanobu Iwai, Soichiro Koyama, Shigeo Tanabe, Shohei Osawa, Kazuya Takeda, Ikuo Motoya, Hiroaki Sakurai, Yoshikiyo Kanada, and Nobutoshi Kawamura. **The validity of spatiotemporal gait analysis using dual laser range sensors: a cross-sectional study**. *Archives of physiotherapy* 9:1 (2019), 1–8 (see page 57).

[100] Matilde Bertoli, Andrea Cereatti, Diana Trojaniello, Laura Avanzino, Elisa Pelosin, Silvia Din, Lynn Rochester, Pieter Ginis, Esther Bekkers, Anat Mirelman, Jeffrey Hausdorff, and Ugo Della Croce. **Estimation of spatio-temporal parameters of gait from magneto-inertial measurement units: Multi-center validation among Parkinson, mildly cognitively impaired and healthy older adults**. *BioMedical Engineering OnLine* 17 (May 2018). DOI: 10.1186/s12938-018-0488-2 (see page 57).

[101] Felipe García-Pinillos, Pedro Á Latorre-Román, Víctor M Soto-Hermoso, Juan A Párraga-Montilla, Antonio Pantoja-Vallejo, Rodrigo Ramírez-Campillo, and Luis E Roche-Seruendo. **Agreement between the spatiotemporal gait parameters from two different wearable devices and high-speed video analysis**. *PloS one* 14:9 (2019), e0222872 (see page 57).

[102] M. Sojer, G. Ebersbach, J. Wissel, J. Müller, and W. Poewe. **Comparative analysis of gait in Parkinson's disease, cerebellar ataxia and subcortical arteriosclerotic encephalopathy**. *Gait and Posture* 10:1 (1999), 67 –68. ISSN: 0966-6362. DOI: https://doi.org/10.1016/S0966-6362(99)90419-8. URL: http://www.sciencedirect.com/science/article/pii/S0966636299904198 (see page 57).

[103] Maja Milošević, Mihovil Logar, and Biljana Djordjević. **Mineralogical analysis of a clay body from Zlakusa, Serbia, used in the manufacture of traditional pottery – CORRIGENDUM**. *Clay Minerals* (2020), 1–1. DOI: 10.1180/clm.2020.31 (see page 58).

[104] What Edge Computing Means for Infrastructure and Operations Leaders, 2017. https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders/ (see page 59).

[105] E. Ahmed, P. Chatzimisios, B.B. Gupta, Y. Jararweh, and H. Song. **Recent advances in fog and mobile edge computing**. *Transactions on Emerging Telecommunications Technologies* 29:4 (2018). DOI: 10.1002/ett.3307 (see page 60).

[106] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, and S. Schmid. **Adaptable and Data-Driven Softwarized Networks: Review, Opportunities, and Challenges**. *Proceedings of the IEEE* 107:4 (2019), 711–731. ISSN: 1558-2256. DOI: 10.1109/JPROC.2019.2895553 (see page 61).

[107] M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer. **Flexibility in Softwarized Networks: Classifications and Research Challenges**. *IEEE Communications Surveys Tutorials* 21:3 (2019), 2600–2636. ISSN: 2373-745X. DOI: 10.1109/COMST.2019.2892806 (see page 61).

[108] S.-M. Zhang and A.K. Sangaiah. **Reliable design for virtual network requests with location constraints in edge-of-things computing**. *Eurasip Journal on Wireless Communications and Networking* 2018:1 (2018). DOI: 10.1186/s13638-018-1075-8 (see page 61).

[109] S. Aditya, K. Subratie, and R. J. Figueiredo. **PerSoNet: Software-Defined Overlay Virtual Networks Spanning Personal Devices Across Social Network Users**. In: *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 2018, 171–180. DOI: 10.1109/CloudCom2018.2018.00043 (see page 61).

[110] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita. **Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources**. *Journal of Lightwave Technology* 36:7 (2018), 1420–1428. DOI: 10.1109/JLT.2018.2800660 (see page 61).

[111] R. Onet, R. Burian, C. Campeanu, I. Ivanciu, D. Zinca, and V. Dobrota. **Automatic Deployment of a Network Overlay in an Intelligent Transportation System: Docker and Open Baton Approach**. In: *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. 2019, 1–6. DOI: 10.1109/ROEDUNET.2019.8909588 (see page 61).

[112] C. Huang, M. Chiang, D. Dao, W. Su, S. Xu, and H. Zhou. **V2V Data Offloading for Cellular Network based on the Software Defined Network (SDN) inside Mobile Edge Computing (MEC) Architecture**. *IEEE Access* (2018). DOI: 10.1109/ACCESS.2018.2820679 (see page 61).

[113] K. Wang, H. Yin, W. Quan, and G. Min. **Enabling Collaborative Edge Computing for Software Defined Vehicular Networks**. *IEEE Network* (2018). DOI: 10.1109/MNET.2018.1700364 (see page 61).

[114] J. Nakazato, Y. Tao, G. K. Tran, and K. Sakaguchi. **Revenue Model with Multi-Access Edge Computing for Cellular Network Architecture**. In: *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. 2019, 21–26. DOI: 10.1109/ICUFN.2019.8805914 (see page 61).

[115] M. Chen and Y. Hao. **Task Offloading for Mobile Edge Computing in Software Defined Ultra-dense Network**. *IEEE Journal on Selected Areas in Communications* (2018). DOI: 10.1109/JSAC.2018.2815360 (see page 61).

[116] K. Kaur, S. Garg, G.S. Aujla, N. Kumar, J.J.P.C. Rodrigues, and M. Guizani. **Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay**. *IEEE Communications Magazine* 56:2 (2018), 44–51. DOI: 10.1109/MCOM.2018.1700622 (see page 61).

[117] H. Guo and J. Liu. **Collaborative Computation Offloading for Multi-Access Edge Computing over Fiber-Wireless Networks**. *IEEE Transactions on Vehicular Technology* (2018). DOI: 10.1109/TVT.2018.2790421 (see page 62).

[118] M. Liu, Y. Mao, S. Leng, and S. Mao. **Full-Duplex Aided User Virtualization for Mobile Edge Computing in 5G Networks**. *IEEE Access* 6 (2017), 2996–3007 (see page 62).

[119] Y. He, F. Richard Yu, N. Zhao, V.C.M. Leung, and H. Yin. **Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach**. *IEEE Communications Magazine* 55:12 (2017), 31–37. DOI: 10.1109/MCOM.2017.1700246 (see page 62).

[120] Y. Zhou, F.R. Yu, J. Chen, and Y. Kuo. **Virtual resource allocation for information-centric heterogeneous networks with mobile edge computing**. In: *2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017*. 2017, 235–240. DOI: 10.1109/INFCOMW.2017.8116382 (see page 62).

[121] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito. **Exploring Container Virtualization in IoT Clouds**. In: *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2016, 1–6. DOI: 10.1109/SMARTCOMP.2016.7501691 (see pages 63, 133).

[122] Kubernetes. *Kubernetes*. 2017. URL: https://kubernetes.io/ (visited on 2017) (see page 66).

[123] Kubernetes. *Viewing Pods and Nodes*. 2017. URL: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore-intro/ (see page 66).

[124] Kubernetes. *Cluster Networking*. 2017. URL: https://kubernetes.io/docs/concepts/cluster-administration/networking/ (see page 68).

[125] *Top 6 Digital Transformation Trends In The Automotive Industry*. [Online];Accessed: 2018-07-28. URL: http://www.forbes.com/sites/danielnewman/2017/07/25/top-6-digital-transformation-trends-in-automotive/#a2b64754e1e3 (see page 81).

[126] *220.76 Billion Smart Transportation Market by Solution Type, Service, and Region - Global Forecast to 2021 - Research and Markets*. https://markets.businessinsider.com/news/stocks/220-76-billion-smart-transportation-market-by-solution-type-service-and-region-global-forecast-to-2021-research-and-markets-1001991360 (see page 81).

[127] Ali Hassan Sodhro, Zongwei Luo, Arun Kumar Sangaiah, and Sung Wook Baik. **Mobile edge computing based QoS optimization in medical healthcare applications**. *International Journal of Information Management* 45 (2019), 308 –318. ISSN: 0268-4012 (see page 82).

[128] Antonio Celesti, Maria Fazio, Antonino Galletta, Lorenzo Carnevale, Jiafu Wan, and Massimo Villari. **An approach for the secure management of hybrid cloud–edge environments**. *Future Generation Computer Systems* 90 (2019), 1 –19. ISSN: 0167-739X (see page 82).

[129] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti. **PROUD #x2014;Public Road Urban Driverless-Car Test**. *IEEE Transactions on Intelligent Transportation Systems* 16:6 (2015), 3508–3519. ISSN: 1524-9050. DOI: 10.1109/TITS.2015.2477556 (see page 82).

[130] *Intersection Car Accidents: Statistics, Causes and Possible Solutions*. [Online];Accessed: 2018-07-28. URL: http://zaneslaw.com/intersection-car-accidents/ (see page 82).

[131] K. N. Bui and J. J. Jung. **ACO-Based Dynamic Decision Making for Connected Vehicles in IoT System**. *IEEE Transactions on Industrial Informatics* 15:10 (2019), 5648–5655. DOI: 10.1109/TII.2019.2906886 (see page 83).

[132] A. Celesti, A. Galletta, L. Carnevale, M. Fazio, A. Ĺay Ekuakille, and M. Villari. **An IoT Cloud System for Traffic Monitoring and Vehicular Accidents Prevention Based on Mobile Sensor Data Processing**. *IEEE Sensors Journal* 18:12 (2018), 4795–4802. ISSN: 1530-437X. DOI: 10.1109/JSEN.2017.2777786 (see page 84).

[133] S. Tsurumi and T. Fujii. **Reliable vehicle-to-vehicle communication using spectrum environment map**. In: *2018 International Conference on Information Networking (ICOIN)*. 2018, 310–315. DOI: 10.1109/ICOIN.2018.8343131 (see page 84).

[134] H. Rashid, M. J. F. Ashrafi, M. Azizi, and M. R. Heydarinezhad. **Intelligent traffic light control based on clustering using Vehicular Ad-hoc Networks**. In: *2015 7th Conference on Information and Knowledge Technology (IKT)*. 2015, 1–6. DOI: 10.1109/IKT.2015.7288801 (see page 84).

[135] M. M. Abdelhameed, M. Abdelaziz, S. Hammad, and O. M. Shehata. **A Hybrid Fuzzy-Genetic Controller for a multi-agent intersection control system**. In: *2014 International Conference on Engineering and Technology (ICET)*. 2014, 1–6. DOI: 10.1109/ICEngTechnol.2014.7016755 (see page 84).

[136] Khac-Hoai Nam Bui, Jason J. Jung, and David Camacho. **Consensual Negotiation-Based Decision Making for Connected Appliances in Smart Home Management Systems**. In: *Sensors*. 2018 (see page 84).

[137] M. M. Abdelhameed, M. Abdelaziz, S. Hammad, and O. M. Shehata. **Development and evaluation of a multi-agent autonomous vehicles intersection control system**. In: *2014 International Conference on Engineering and Technology (ICET)*. 2014, 1–6. DOI: 10.1109/ICEngTechnol.2014.7016754 (see page 84).

[138] A. Buzachis, A. Celesti, A. Galletta, M. Fazio, and M. Villari. **A Secure and Dependable Multi-Agent Autonomous Intersection Management (MA-AIM) System Leveraging Blockchain Facilities**. In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. 2018, 226–231. DOI: 10.1109/UCC-Companion.2018.00060 (see pages 84, 105).

[139] C. Wuthishuwong and A. Traechtler. **Vehicle to infrastructure based safe trajectory planning for Autonomous Intersection Management**. In: *2013 13th International Conference on ITS Telecommunications (ITST)*. 2013, 175–180. DOI: 10.1109/ITST.2013.6685541 (see page 84).

[140] K. N. Bui and J. J. Jung. **ACO-Based Dynamic Decision Making for Connected Vehicles in IoT System**. *IEEE Transactions on Industrial Informatics* 15:10 (2019), 5648–5655. ISSN: 1941-0050. DOI: 10.1109/TII.2019.2906886 (see page 84).

[141] J. Petit and S. E. Shladover. **Potential Cyberattacks on Automated Vehicles**. *IEEE Transactions on Intelligent Transportation Systems* 16:2 (2015), 546–556. ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2342271 (see page 85).

[142] M. Singh and S. Kim. **Crypto trust point (cTp) for secure data sharing among intelligent vehicles**. In: *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. 2018, 1–4. DOI: 10.23919/ELINFOCOM.2018.8330663 (see page 85).

[143] M. G. M. Mehedi Hasan, A. Datta, and M. A. Rahman. **Poster Abstract: Chained of Things: A Secure and Dependable Design of Autonomous Vehicle Services**. In: *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2018, 298–299. DOI: 10.1109/IoTDI.2018.00048 (see page 85).

[144] Y. Yuan and F. Wang. **Towards blockchain-based intelligent transportation systems**. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, 2663–2668. DOI: 10.1109/ITSC.2016.7795984 (see page 85).

[145] Satoshi Nakamoto. **Bitcoin: A Peer-to-Peer Electronic Cash System** (Mar. 2009) (see page 87).

[146] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. **Zerocash: Decentralized Anonymous Payments from Bitcoin**. In: *2014 IEEE Symposium on Security and Privacy*. 2014, 459–474. DOI: 10.1109/SP.2014.36 (see page 87).

[147] Daniel Davis Wood. **ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER**. In: 2014 (see page 87).

[148] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Q. Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. **Hyperledger fabric: a distributed operating system for permissioned blockchains**. *ArXiv* abs/1801.10228 (2018) (see page 88).

[149] Richard Brown, James Carlyle, Ian Grigg, and Mike Hearn. *Corda: An Introduction*. Sept. 2016. DOI: 10.13140/RG.2.2.30487.37284 (see page 88).

[150] Christian Cachin and Marko Vukolic. **Blockchain Consensus Protocols in the Wild**. *ArXiv* abs/1707.01873 (2017) (see page 89).

[151] Kurt Dresner and Peter Stone. **A Multiagent Approach to Autonomous Intersection Management**. *J. Artif. Intell. Res. (JAIR)* 31 (Jan. 2008), 591–656. DOI: 10.1613/jair.2502 (see page 91).

[152] Charles Neu, Regio Michelin, Avelino Zorzo, and Roben Lunardi. **Distributed Access Control on IoT Ledger-based Architecture**. In: Apr. 2018. DOI: 10.1109/NOMS.2018.8406154 (see page 96).

[153] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang. **CreditCoin: A Privacy-Preserving Blockchain-Based Incentive Announcement Network for Communications of Smart Vehicles**. *IEEE Transactions on Intelligent Transportation Systems* 19:7 (2018), 2204–2220. ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2777990 (see page 96).

[154] Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. **Block-VN: A Distributed Blockchain Based Vehicular Network Architecture in Smart City**. *JIPS* 13 (2017), 184–195 (see page 96).

[155] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. **BLOCKBENCH: A Framework for Analyzing Private Blockchains**. In: May 2017, 1085–1100. DOI: 10.1145/3035918.3064033 (see page 96).

[156] Carly Daley, Tammy Toscos, and Michael Mirro. **Data Integration and Interoperability for Patient-Centered Remote Monitoring of Cardiovascular Implantable Electronic Devices**. *Bioengineering* 6:1 (2019), 25 (see page 113).

[157] Julia Adler-Milstein. *Moving Past the EHR Interoperability Blame Game*. [accessed February 28, 2019]. 2017. URL: http://www.grandviewresearch.com/press-release/global-e-health-market (see page 113).

[158] A. Celesti, M. Fazio, A. Romano, and M. Villari. **A hospital cloud-based archival information system for the efficient management of HL7 big data**. In: 2016, 406–411 (see page 113).

[159] PR Newswire. *Survey: Patients See 18.7 Different Doctors on Average*. [accessed February 28, 2019]. 2010. URL: "http://www.prnewswire.com/news-releases/surveypatients-see-187-different-doctors-on-average-92171874.html" (see page 114).

[160] Paola Mosconi, Silvia Radrezza, Emanuele Lettieri, and Eugenio Santoro. **Use of Health Apps and Wearable Devices: Survey Among Italian Associations for Patient Advocacy**. *JMIR Mhealth Uhealth* 7:1 (2019) (see page 114).

[161] Ariel Ekblaw, Asaph Azaria, John D. Halamka, and Andrew Lippman. **A Case Study for Blockchain in Healthcare: " MedRec " prototype for electronic health records and medical research data**. In: 2016 (see page 114).

[162] Xiao Yue, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. **Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control**. In: vol. 40. Oct. 2016, 218. DOI: 10.1007/s10916-016-0574-6 (see page 114).

[163] Paul Beninger and Michael A. Ibara. **Pharmacovigilance and Biomedical Informatics: A Model for Future Development**. In: vol. 38. Nov. 2016. DOI: 10.1016/j.clinthera.2016.11.006 (see page 114).

[164] G. Zyskind, O. Nathan, and A. '. Pentland. **Decentralizing Privacy: Using Blockchain to Protect Personal Data**. In: *2015 IEEE Security and Privacy Workshops*. 2015, 180–184. DOI: 10.1109/SPW.2015.27 (see page 114).

[165] Tomas Mikula and Rune Hylsberg Jacobsen. **Identity and Access Management with Blockchain in Electronic Healthcare Records**. *2018 21st Euromicro Conference on Digital System Design (DSD)* (2018), 699–706 (see page 114).

[166] Xiaoshuai Zhang and Stefan Poslad. **Blockchain Support for Flexible Queries with Granular Access Control to Electronic Medical Records (EMR)**. In: Aug. 2018. DOI: 10.1109/ICC.2018.8422883 (see page 114).

[167] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. **MedRec: Using Blockchain for Medical Data Access and Permission Management**. In: Aug. 2016, 25–30. DOI: 10.1109/OBD.2016.11 (see pages 114, 119).

[168] Edward Y. Chang, Shih-Wei Liao, Chun-Ting Liu, Wei-Chen Lin, Pin-Wei Liao, Wei-Kang Fu, Chung-Huan Mei, and Emily J. Chang. **DeepLinQ: Distributed Multi-Layer Ledgers for Privacy-Preserving Data Sharing**. In: Dec. 2018, 173–178. DOI: 10.1109/AIVR.2018.00037 (see page 119).

[169] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, and Rajiv Ranjan. **Osmotic Computing: A New Paradigm for Edge/Cloud Integration**. *IEEE Cloud Computing* 3:6 (2016), 76–83. DOI: 10.1109/mcc.2016.124. URL: https://doi.org/10.1109/mcc.2016.124 (see pages 128, 129).

[170] Massimo Villari, Antonio Celesti, and Maria Fazio. "Towards Osmotic Computing: Looking at Basic Principles and Technologies." In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2017, 906–915. DOI: 10.1007/978-3-319-61566-0_86. URL: https://doi.org/10.1007/978-3-319-61566-0_86 (see page 128).

[171] S. K. Datta and C. Bonnet. **Next-Generation, Data Centric and End-to-End IoT Architecture Based on Microservices**. In: *2018 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*. 2018, 206–212. DOI: 10.1109/ICCE-ASIA.2018.8552135 (see page 128).

[172] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi. **Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing**. *IEEE Access* 7 (2019), 42735–42744. DOI: 10.1109/ACCESS.2019.2907615 (see page 130).

[173] Y. A. Younis, K. Kifayat, and M. Merabti. **A novel evaluation criteria to cloud based access control models**. In: *2015 11th International Conference on Innovations in Information Technology (IIT)*. 2015, 68–73. DOI: 10.1109/INNOVATIONS.2015.7381517 (see page 133).

[174] Z. B. Yahya, F. B. Ktata, and K. Ghedira. **Multi-organizational Access Control Model Based on Mobile Agents for Cloud Computing**. In: *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 2016, 656–659. DOI: 10.1109/WI.2016.0116 (see page 133).

[175] Y. Demchenko, C. Ngo, C. de Laat, and C. Lee. **Federated Access Control in Heterogeneous Intercloud Environment: Basic Models and Architecture Patterns**. In: *2014 IEEE International Conference on Cloud Engineering*. 2014, 439–445. DOI: 10.1109/IC2E.2014.84 (see page 134).

[176] *Agento*. URL: https://github.com/lcarnevale/agento (see page 138).

[177] M. Fazio, A. Celesti, F.G. Marquez, A. Glikson, and M. Villari. **Exploiting the FIWARE cloud platform to develop a remote patient monitoring system**. In: vol. 2016-February. cited By 16. 2016, 264–270. DOI: 10.1109/ISCC.2015.7405526. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84961967567&doi=10.1109%2fISCC.2015.7405526&partnerID=40&md5=90d1713b21a2c2e4e1fb8f49c022255e (see page 145).

[178] *Unlocking the potential of the Internet of Things*. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/. [Online];Accessed: 2018-09-17 (see page 146).

[179] *Internet of Things (IoT) Healthcare Market is Expected to Reach 136.8 Billion Worldwide, by 2021*. Available: http://www.marketwatch.com/press-release/. [Online];Accessed: 2018-09-17 (see page 146).

[180] E. Rachkidi, E. H. Cherkaoui, M. Ait-idir, N. Agoulmine, N. C. Taher, M. Santos, and S. Fernandes. **Towards Efficient Automatic Scaling and Adaptive Cost-Optimized eHealth Services in Cloud**. In: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, 1–6. DOI: 10.1109/GLOCOM.2015.7417751 (see page 146).

[181] A. Buzachis, A. Galletta, L. Carnevale, A. Celesti, M. Fazio, and M. Villari. **Towards Osmotic Computing: Analyzing Overlay Network Solutions to Optimize the Deployment of Container-Based Microservices in Fog, Edge and IoT Environments**. In: *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. 2018, 1–10. DOI: 10.1109/CFEC.2018.8358729 (see pages 146, 150).

[182] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan. **Osmotic Computing: A New Paradigm for Edge/Cloud Integration**. *IEEE Cloud Computing* 3:6 (2016), 76–83. ISSN: 2325-6095. DOI: 10.1109/MCC.2016.124 (see page 146).

[183] Alina Buzachis, Giuseppe Massimo Bernava, Mario Busà, Giovanni Pioggia, and Massimo Villari. **Towards Osmotic Computing: Future Prospect for the Health Information Technology (HIT) Systems of ISASI-CNR (ME)**. In: *IEEE ISCC 2018 Workshops - 3rd IEEE Workshop on ICT Solutions for Health (ISCC 2018 Workshops - ICTS4eHealth 2018)*. Natal, Brazil, June 2018 (see page 147).

[184] I. Ben Ida, A. Jemai, and A. Loukil. **A survey on security of IoT in the context of eHealth and clouds**. In: *2016 11th International Design Test Symposium (IDT)*. 2016, 25–30. DOI: 10.1109/IDT.2016.7843009 (see page 147).

[185] R. Boussada, M. E. Elhdhili, and L. A. Saidane. **Privacy Preserving Solution for Internet of Things with Application to eHealth**. In: *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. 2017, 384–391. DOI: 10.1109/AICCSA.2017.75 (see page 147).

[186] H. Ahmed, A. Alsadoon, P. W. C. Prasad, N. Costadopoulos, L. S. Hoe, and A. Elchoemi. **Next generation cyber security solution for an eHealth organization**. In: *2017 5th International Conference on Information and Communication Technology (ICoIC7)*. 2017, 1–5. DOI: 10.1109/ICoICT.2017.8074723 (see page 147).

[187] F. Firouzi, B. Farahani, M. Ibrahim, and K. Chakrabarty. **From EDA to IoT eHealth: Promise, Challenges, and Solutions**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018), 1–1. ISSN: 0278-0070. DOI: 10.1109/TCAD.2018.2801227 (see page 147).

[188] Pavlos Kosmides, Konstantinos Demestichas, Evgenia Adamopoulou, Nikos Koutsouris, Yannis Oikonomidis, and Vanessa De Luca. **InLife: Combining Real Life with Serious Games using IoT** (Aug. 2018) (see page 147).

[189] Tiago Martins, Vítor Carvalho, and Filomena Soares. **A serious game for rehabilitation of neurological disabilities: Premilinary study** (Apr. 2015) (see page 147).

[190] Alana Da Gama, Pascal Fallavollita, Veronica Teichrieb, and Nassir Navab. **Motor Rehabilitation Using Kinect: A Systematic Review**. 4 (Feb. 2015), 150206061432001 (see page 147).

[191] V. Fernandez-Cervantes, E. Stroulia, C. Castillo, L. Oliva, and F. Gonzalez. **Serious rehabilitation games with Kinect**. In: *2015 IEEE Games Entertainment Media Conference (GEM)*. 2015, 1–1. DOI: 10.1109/GEM.2015.7377254 (see page 147).

[192] A. Ballas, T. Santad, K. Sookhanaphibarn, and W. Choensawat. **Game-based system for learning labanotation using Microsoft Kinect**. In: *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*. 2017, 1–3. DOI: 10.1109/GCCE.2017.8229481 (see page 148).

[193] D. Ferreira, R. Oliveira, and O. Postolache. **Physical rehabilitation based on kinect serious games**. In: *2017 Eleventh International Conference on Sensing Technology (ICST)*. 2017, 1–6. DOI: 10.1109/ICSensT.2017.8304512 (see page 148).

[194] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. **How the Dataweb can support cloud federation: Service representation and secure data exchange**. In: cited By 16. 2012, 73–79. DOI: 10.1109/NCCA.2012.26. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84875618191&doi=10.1109%2fNCCA.2012.26&partnerID=40&md5=5e911776c5eb6dbc87a404a9f0a52caa (see page 148).

[195] James Burke, M D. J. McNeill, Darryl Charles, Philip Morrow, J H. Crosbie, and Suzanne Mcdonough. **Optimising engagement for stroke rehabilitation using serious games**. 25 (Dec. 2009), 1085–1099 (see page 150).

[196] J. W. Burke, M. D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough. **Augmented Reality Games for Upper-Limb Stroke Rehabilitation**. In: *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*. 2010, 75–78. DOI: 10.1109/VS-GAMES.2010.21 (see page 150).

[197] Lorenzo Carnevale, Antonio Celesti, Antonino Galletta, Schahram Dustdar, and Massimo Villari. **Osmotic Computing as a Distributed Multi-Agent system: the Body Area Network Scenario**. *Internet of Things* 5 (Mar. 2019). DOI: 10.1016/j.iot.2019.01.001 (see pages 152, 159).

[198] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, and R. Ranjan. **Osmotic Flow: Osmotic Computing + IoT Workflow**. *IEEE Cloud Computing* 4:2 (2017), 68–75. ISSN: 2325-6095. DOI: 10.1109/MCC.2017.22 (see page 158).

[199] Thomas Rausch, Schahram Dustdar, and R Ranjan. **Osmotic Message-Oriented Middleware for the Internet of Things**. *IEEE Cloud Computing* 5 (Mar. 2018), 17–25. DOI: 10.1109/MCC.2018.022171663 (see page 158).

[200] Vishal Sharma, Ilsun You, Ravinder Kumar, and Pankoo Kim. **Computational Offloading for Efficient Trust Management in Pervasive Online Social Networks Using Osmotic Computing**. *IEEE Access* PP (Mar. 2017), 1–1. DOI: 10.1109/ACCESS.2017.2683159 (see page 159).

[201] A. Buzachis, A. Galletta, L. Carnevale, A. Celesti, M. Fazio, and M. Villari. **Towards Osmotic Computing: Analyzing Overlay Network Solutions to Optimize the Deployment of Container-Based Microservices in Fog, Edge and IoT Environments**. In: *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. 2018, 1–10. DOI: 10.1109/CFEC.2018.8358729 (see page 159).

[202] M. Villari, M. Fazio, S. Dustdar, O. Rana, L. Chen, and R. Ranjan. **Software Defined Membrane: Policy-Driven Edge and Internet of Things Security**. *IEEE Cloud Computing* 4:4 (2017), 92–99. ISSN: 2325-6095. DOI: 10.1109/MCC.2017.3791014 (see page 159).

[203] V. Sharma, K. Srinivasan, D. N. K. Jayakody, O. Rana, and R. Kumar. **Managing Service-Heterogeneity using Osmotic Computing**. *ArXiv e-prints* (Apr. 2017). arXiv: 1704.04213 [cs.DC] (see page 160).

[204] Geoffrey Fox, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. **Status of Serverless Computing and Function-as-a-Service(FaaS) in Industry and Research**. In: Aug. 2017. DOI: 10.13140/RG.2.2.15007.87206 (see page 160).

[205] Serverless or microservices - which is better?, https://www.quora.com/Serverless-or-microservices-which-is-better (see page 160).

[206] *The IoT in 2030: 24 billion connected things generating $1.5 trillion.* Last accessed 27 July 2020. URL: https://iotbusinessnews.com/2020/05/20/03177-the-iot-in-2030-24-billion-connected-things-generating-1-5-trillion/ (see page 171).

[207] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. **CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms**. *Software: Practice and Exp.* 41:1 (2011), 23–50. DOI: 10.1002/spe.995. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.995. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.995 (see page 172).

[208] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. **iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments**. *Software: Practice and Experience* 47:9 (2017), 1275–1296. DOI: 10.1002/spe.2509. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2509. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509 (see page 172).

[209] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. **EdgeCloudSim: An environment for performance evaluation of edge computing systems**. *Trans. Emerging Telecommunications Tech.* 29:11 (2018), e3493. DOI: 10.1002/ett.3493. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3493. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3493 (see page 172).

[210] Márcio Lopes and Wilson et al. Higashino. **MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing**. In: Dec. 2017, 47–52. DOI: 10.1145/3147234.3148101 (see page 173).

[211] Xuezhi Zeng, Saurabh Kumar Garg, Peter Strazdins, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, and Rajiv Ranjan. **IOTSim: A simulator for analysing IoT applications**. *Journal of Systems Architecture* 72 (2017). Design Automation for Embedded Ubiquitous Computing Systems, 93 –107. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2016.06.008. URL: http://www.sciencedirect.com/science/article/pii/S1383762116300662 (see page 173).

[212] I. Lera, C. Guerrero, and C. Juiz. **YAFS: A Simulator for IoT Scenarios in Fog Computing**. *IEEE Access* 7 (2019), 91745–91758 (see page 173).

[213] Damián Fernández-Cerero, Alejandro Fernández-Montes, F. Javier Ortega, Agnieszka Jakóbik, and Adrian Widlak. **Sphere: Simulator of edge infrastructures for the optimization of performance and resources energy consumption**. *Simulation Modelling Practice and Theory* 101 (2020). Modeling and Simulation of Fog Computing, 101966. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2019.101966. URL: http://www.sciencedirect.com/science/article/pii/S1569190X19300991 (see page 173).

[214] Damián Fernández-Cerero, Alejandro Fernández-Montes, Agnieszka Jakóbik, Joanna Kołodziej, and Miguel Toro. **SCORE: Simulator for cloud optimization of resources and energy consumption**. *Simulation Modelling Practice and Theory* 82 (2018), 160 –173. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2018.01.004. URL: http://www.sciencedirect.com/science/article/pii/S1569190X18300030 (see page 173).

[215] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran. **EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures**. In: *2017 IEEE Fog World Congr.* 2017, 1–6. DOI: 10.1109/FWC.2017.8368525 (see page 174).

[216] Philip Wette, M Dräxler, Arne Schwabe, F Wallaschek, M Zahraee, and H Karl. **MaxiNet: Distributed emulation of software-defined networks**. In: June 2014, 1–9. DOI: 10.1109/IFIPNetworking.2014.6857078 (see page 174).

[217] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso. **Fogbed: A Rapid-Prototyping Emulation Environment for Fog Computing**. In: *2018 IEEE International Conference on Communications (ICC)*. 2018, 1–7. DOI: 10.1109/ICC.2018.8423003 (see page 174).

[218] Jonathan Hasenburg, Martin Grambow, and Elias et al. Grünewald. **MockFog: Emulating Fog Computing Infrastructure in the Cloud**. In: June 2019. DOI: 10.1109/ICFC.2019.00026 (see page 174).

[219] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. **Optimized IoT service placement in the fog**. *Service Oriented Computing and Applications* 11 (2017), 427–443 (see page 174).

[220] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi. **Multiobjective Optimization for Computation Offloading in Fog Computing**. *IEEE Internet of Things Journal* 5:1 (2018), 283–294. ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2780236 (see page 174).

[221] Y. Xiao and M. Krunz. **QoE and power efficiency tradeoff for fog computing networks with fog node cooperation**. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, 1–9 (see page 174).

[222] M. Peuster, H. Karl, and S. van Rossem. **MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments**. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 2016, 148–153. DOI: 10.1109/NFV-SDN.2016.7919490 (see page 180).

[223] *MaestroNG*. Last accessed 24 July 2020. URL: http://maestro-ng.readthedocs.io (see pages 183, 184).

[224] H. W. Kuhn. **The Hungarian method for the assignment problem**. *Naval Research Logistics* 2 (1955), 83–97 (see page 185).

[225] G. Rakshith, M. V. Rahul, G. S. Sanjay, B. V. Natesha, and G. Ram Mohana Reddy. **Resource Provisioning Framework for IoT Applications in Fog Computing Environment**. In: *2018 IEEE Intl. Conf. Advanced Networks and Telecommunications Systems (ANTS)*. 2018, 1–6 (see pages 185, 186).

[226] Daniel Balouek-Thomert, Eduard Gibert Renart, Ali Reza Zamani, Anthony Simonet, and Manish Parashar. **Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows**. *The International Journal of High Performance Computing Applications* 33 (Nov. 2019), 1159–1174. DOI: 10.1177/1094342019877383 (see page 197).

[227] Nicolas Seydoux, K. Drira, N. Hernandez, and T. Monteil. **EDR: A generic approach for the distribution of rule-based reasoning in a Cloud-Fog continuum**. *Semantic Web* 11 (2020), 623–654 (see page 197).

[228] P. Gkikopoulos. **Data Distribution and Exploitation in a Global Microservice Artefact Observatory**. In: *2019 IEEE World Congress on Services (SERVICES)*. Vol. 2642-939X. 2019, 319–322. DOI: 10.1109/SERVICES.2019.00089 (see page 200).

[229] Jesús García-Galán, Pablo Trinidad, Omer F. Rana, and Antonio Ruiz-Cortés. **Automated configuration support for infrastructure migration to the cloud**. *Future Generation Computer Systems* 55 (2016), 200 –212. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.03.006. URL: http://www.sciencedirect.com/science/article/pii/S0167739X15000618 (see page 201).

[230] Amir Dastjerdi, Saurabh Garg, Omer Rana, and Rajkumar Buyya. **CloudPick: A framework for QoS-aware and ontology-based service deployment across clouds**. *Software: Practice and Experience* 45 (Sept. 2014). DOI: 10.1002/spe.2288 (see page 201).

[231] H. W. Kuhn. **The Hungarian method for the assignment problem**. *Naval Research Logistics Quarterly* 2:1-2 (1955), 83–97. DOI: https://doi.org/10.1002/nav.3800020109. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109 (see page 206).

[232] Tomas Vitvar, Maciej Zaremba, Matthew Moran, and Adrian Mocan, 31–49. In: *Semantic Web Services Challenge: Results from the First Year*. Ed. by Charles Petrie, Tiziana Margaria, Holger Lausen, and Michal Zaremba. Boston, MA: Springer US, 2009. ISBN: 978-0-387-72496-6. DOI: 10.1007/978-0-387-72496-6_3. URL: https://doi.org/10.1007/978-0-387-72496-6_3 (see page 208).

[233] Iris Braun, A. Strunk, Gergana Stoyanova, and B. Buder. **ConQo – A Context-And QoS-Aware Service Discovery**. In: 2008 (see page 208).

[234] Josef Spillner and A. Schill. **A Versatile and Scalable Everything-as-a-Service Registry and Discovery**. In: *CLOSER*. 2013 (see page 208).

[235] A. V. Dastjerdi and R. Buyya. **Compatibility-Aware Cloud Service Composition under Fuzzy Preferences of Users**. *IEEE Transactions on Cloud Computing* 2:1 (2014), 1–13. DOI: 10.1109/TCC.2014.2300855 (see page 208).

[236] Yezheng Liu, Tingting Zhu, Yuanchun Jiang, and Xiao Liu. **Service matchmaking for Internet of Things based on probabilistic topic model**. *Future Generation Computer Systems* 94 (2019), 272 –281. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2018.11.040. URL: http://www.sciencedirect.com/science/article/pii/S0167739X18320788 (see page 208).

[237] Sukhpal Singh Gill and Inderveer Chana. **QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review**. *ACM Computing Surveys* 48 (Dec. 2015), 1–46. DOI: 10.1145/2843889 (see page 208).