UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
"GIOVANNI DEGLI ANTONI"

CORSO DI DOTTORATO IN INFORMATICA
XXXVI CICLO, INF/01

TESI DI DOTTORATO DI RICERCA

ASSURANCE-AWARE 5G EDGE-CLOUD ARCHITECTURES FOR INTENSIVE DATA ANALYTICS

Candidate: Dott. Filippo Berto

Supervisor:	Prof. Marco Anisetti
Co-Supervisor:	Prof. Claudio Agostino Ardagna
External Supervisor:	Dr. Massimo Valla
PhD Prog. Coordinator:	Prof. Roberto Sassi

A.A. 2022/2023

Abstract

Modern data-intensive applications are increasingly demanding in terms of non-functional properties such as performance, latency, security, and privacy. In order to achieve such non-functional properties, modern applications benefit from being developed as a composition of services and deployed in a heterogeneous continuum infrastructure that includes Edge and Cloud facilities. In this scenario, the infrastructures used to deploy services play a crucial role in providing or supporting non-functional properties of the applications. For instance, low latency can be achieved via deployment of services in the far edge nodes. Most of the current literature addresses the problem of deployment of single (stateless)services mainly with the aim of ensuring and verifying requirements in terms of resources and performance. Only few solutions exist to deploy applications made of services workflows, and in most of the cases they are focused on functional composition. In general, they fail to address composition deployment preserving advanced non-functional properties such as security and privacy. This thesis proposes novel assurance methodology for modern continuum infrastructures, enabling lightweight in-depth verification and assessment of non-functional properties constituting the key cornerstone for a fully non-functional-aware deployment of service based applications. The thesis proposes an advanced continuum infrastructure, where the 5G MEC is integrated as an Edge node. It also considers a continuum which is empowered with a big data ecosystem of services, where data-intensive analytic workflows can be executed to support critical applications. The assurance methodology defined in the thesis is collaborative and lightweight, and is based on i) transparent collection of evidence representing measurements of relevant continuum states (obtained via monitoring or testing of standard infrastructure-level hooks), ii) aggregation of measurements into metrics and iii) contracts linking metrics to specific non-functional properties. The assurance methodology decouples infrastructure assurance from data processing assurance and application-level assurance. It is the first at-

tempt to suggest that infrastructure and data processing assurance can effectively complement application-level assurance with a limited increase in computational effort while fully applicable in modern continuum infrastructures.

The contributions of this thesis are manifold: i) a generic assurance methodology for modern infrastructures ii) a set of specific verticalization of the generic assurance for 5G MEC, Big Data pipelines and CDN networks, iii) a novel notion of continuum empowered by 5G, iv) property aware deployment solution for the continuum integrating assurance controls, v) a complete realization of a continuum infrastructure with simulated 5G nodes and a real data-intensive application for robotic agronomy vi) full experimental evaluation of utility usability and performance.

The assurance approaches developed in the thesis have been applied to a real-world scenario through the construction of a complete 5G-enabled Edge-Cloud continuum infrastructure. This was achieved by integrating a 5G network simulator, a MEC deployment infrastructure, a Big Data engine, and a data analysis pipeline platform. This continuum was used to realize a concrete application in the area of IoT-based automated agronomy. Such application is capable of handling the collection, ingestion, analysis and visualization of on-field data. Such complex modern application requires guarantees on a set of advanced non-functional properties that were verified adopting the assurance methodology defined in the thesis.

The obtained results demonstrate the utility and usability of the assurance in the context of modern data-intensive application as well as the limited impact in terms of performance obtained thanks to the approach based on infrastructure-level monitoring and lightweight evidence collection.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors and mentors, Prof. Marco Anisetti and Prof. Claudio Ardagna, for their valuable advice, support, and guidance during my PhD studies. Their experience and knowledge have inspired me and helped me grow both as a researcher and as an individual. I also extend my thanks to Dr. Samira Maghool, Prof. Paolo Ceravolo, Prof. Valerio Bellandi, and Prof. Ernesto Damiani for their support in my research. I would like to thank Dr. Massimo Valla, Prof. Massimo Banzi and Christina Fra' for their supervision and welcoming collaboration on the more technical aspects of this work. A special thank also goes to my colleagues and friends in the Sesar Lab, Annalisa Barsotti, Dr. Samira Maghool, Antongiacomo Polimeno, Dr. Gabriel Marques Tavares, Jonatan Maggesi, Nicola Bena, Rafael Seidi Oyamada, and Ruslan Bondaruc. Their support and the many laughs we shared made my study and life in Milan a wonderful experience. Finally, I would like to thank my family for always encouraging me to pursue my interests and pushing me to do my best.

This work has been funded by TIM S.p.A., Services Innovation Department, Innovation Lab Milano.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Non-Functional Properties (NFPs), such as performance, security, and privacy, are becoming more and more important for modern applications, as they affect the quality and reliability of the services they provide to the users and customers. However, designing and operating infrastructures that can continuously meet these requirements is not an easy task, especially when the deployed applications are composed of multiple services that run on different platforms and locations, such as edge and cloud facilities. These distributed environments pose various challenges and opportunities for the assurance of NFPs, such as scalability, heterogeneity, dynamism, and complexity. Infrastructure assurance is the process of providing evidence and guarantees that the infrastructure can satisfy and maintain the non-functional requirements of the applications throughout their lifecycle. Infrastructure assurance involves various methods and techniques to assess and improve the NFPs of the infrastructure, such as monitoring, testing, verification, optimization, and adaptation.

We concentrated on two key research lines in this dissertation: i) assurance methodologies for complex distributed infrastructures, including platforms composed of ecosystems of services or service-based network infrastructures, and ii) solutions for assurance-aware deployments in distributed settings, such as edge and cloud facilities.

The primary objective of this research was to develop strategies and methodologies for assessing and improving the NFPs of complex distributed infras-

tructures, including security, privacy, performance, availability and scalability. A particular emphasis was laid on creating a comprehensive assurance methodology that could be applied to generic service-based infrastructures. The objective was to use existing evidence and metrics as much as possible to assess NFPs and to reduce the need for ad hoc probes. The proposed work significantly advances transparent and verifiable monitoring and verification solutions. Infrastructure components share state measurements with the network whilst assurance agents use an open framework for collaboratively verifying NFPs. Our proposed novel assurance framework integrates static and dynamic analysis techniques to verify the NFPs of service ecosystems and service-based network infrastructures. We applied our framework in various domains and scenarios, such as content delivery networks, telecommunications edge networks, and Big Data platforms.

The second research line aims to develop techniques and methods to deploy apps in distributed settings ensuring non-functional aspects such as performance, security, privacy, and cost. Our analysis focused on exploring the correlation between the deployed services and the infrastructure beneath. We created an assurance methodology for deploying edge-cloud, providing a guarantee to fulfil Service Level Agreements (SLAs) for the services. We proposed a novel deployment system that employs optimisation strategies to ascertain the most efficient deployment configuration for an application, taking into account the application's non-functional requisites and the attributes of the available resources. The aim was to continuously oversee and confirm the deployment facilities throughout the continuum, whilst aligning service requirements with facility capabilities. Our methodology was tested using a fully functional 5G simulator, adhering to the most current standards. Multi-access Edge Computing (MEC) capabilities were integrated into the simulator, which in turn helped enhance the current European Telecommunications Standards Institute (ETSI) deployment infrastructure, leading to heightened responsiveness and assurance capabilities.

We experimentally evaluated our solutions for assurance in continuum architectures and deployment on the project MIND Foods Hub[1], where the we also contributed to the designing, implementation and deployment of a big data ecosystem. The aim was to gather agronomic data from on-field sensors, ingest it into a continuous analysis cycle, and automatically generate Machine Learning (ML) models, data visualisations, and real-time updates

[1] `https://www.mindfoodshub.com/`

for users. This system, developed in cooperation with TIM S.p.A., comprises a set of applications deployed within the Edge-Cloud (E2C) continuum, with 5G enabled Internet of Things (IoT) devices operating in the field. The services were distributed partly through TIM's 5G edge network and partly in our on-premise data centre. The project had system significant requirements, particularly in terms of performance, as the users expect real-time response, and in security, given that the university and participating businesses have exclusive ownership of the collected data. Achieving high levels of NFPs was challenging due to the complex and distributed nature of the system, while the literature lacked specific assurance methodologies for E2C environments. The deployment of applications in the E2C continuum required adjustments to certain components chosen for the big data engine to enable transparent service monitoring, continuous verification of NFPs, and deployment automation, for compatibility with the cloud-ready environment of the deployment. These adjustments also influenced the assurance process, which led to the development of methodologies for the E2C continuum.

1.1 Contribution

The research contribution of this thesis can be summarised as follows:

- Analysis of E2C continuum infrastructures, focusing on the challenges and gaps of the current systems, the standards and solutions available for effective management of complex application deployments. Additionally, we examined the capabilities regarding continuous assurance and certification of advanced user-defined SLAs.

- A novel methodology for assurance and certification of infrastructures with a focus on network, computation, deployment and automation of the composition of applications in distributed E2C setting. The methodology has been employed in multiple case studies including telco edge networks (5G, satellite, and IoT), Content Distribution Networks (CDNs), deployment systems for E2C continuum, big data analysis pipelines, and complex application compositions.

- Development of a 5G simulator in a high availability environment using the latest standards. The simulator has been employed as a testing

ground for experimental research on 5G edge and its integration in the continuum.

- Identification of novel NFPs suitable for 5G networks, the relative platform-specific metrics for verifying them, and the current missing functionalities to support their adoption.

- Definition and implementation of a decentralised and collaborative verification and certification methodology based on Information Centric Networking (ICN) networks, resulting in efficient distribution of evidence and certificates among peers.

- Design and implementation of a novel NFP-aware big data engine for E2C continuum and its application in agronomic analysis automation.

- Assurance of big data analytics architecture integrating controls at pipeline, service ecosystem and infrastructure level.

1.2 Organisation

This thesis is organized as follows:

In Chapter 2, an overview of the current state-of-the-art E2C infrastructures is presented. These distributed platforms facilitate the delivery of computing resources and services over both the edge and the cloud. We described the main elements that make up the E2C continuum, such as edge nodes, cloud nodes, network links and orchestration systems. We also discussed the technologies used to implement and manage these elements, such as virtualisation, containerisation, software-defined networking and intent-based provisioning. Moreover, we identified some of the challenges and gaps that need to be addressed to improve the performance, security, privacy, and trustworthiness of edge-cloud infrastructures. In this chapter we presented a thorough and current reference architecture for Edge-Cloud Infrastructures, while also drawing attention to the research prospects and areas in this domain.

In Chapter 3, we presented our approach to assurance for infrastructures and platforms. We defined our infrastructure model as a service-based network,

where each element is represented by a service that exposes its functionalities, monitoring endpoints and NFP. We also described the methodology that we applied to verify the NFP of the infrastructure. We utilised a blend of static and dynamic analysis methodologies to assess the infrastructure based on metrics-produced evidence. Furthermore, we demonstrated the continuous verification process we employed validate the infrastructure throughout its life cycle.

In Chapter 4, we examined the field of edge computing in telecommunications networks. We provided a comprehensive overview of the 5G architecture, including its essential components and established standards, such as Network Function (NF), Network Slicing (NS), and service-based architecture. We then described the implementation of our 5G network simulator built for research purposes. Next, we examined the automation of service deployment and configuration through intent-based networking, which facilitates the expression of network objectives and policies at a high level. Finally, we considered the implementation of these technologies in other specific networks, including satellite edge and IoT networks.

In Chapter 5, we introduced the research topic of assurance in edge networks, with a specific focus on 5G edge networks. We provided an analysis of the primary stakeholders involved in the 5G networks scenario, including network operators, service providers, regulators, and users, and their respective roles and interests in assurance. We also outlined the functionalities of 5G networks that are relevant to non-functional properties, such as network slicing, orchestration, and intent-based networking. Finally, we demonstrate how to extend the methodology presented in chapter 3 to other edge networks.

In Chapter 6, we examined the operation of CDNs in edge networks. We presented an implementation of a CDN based on ICN. We utilised Named Data Networking (NDN), a particular execution of ICN, for constructing and deploying a CDN that can manage diverse content types, and effectively distribute them to other nodes. We presented a decentralised and collaborative methodology for assurance of NFPs using NDN. We utilised an open framework that enables CDN nodes to share their state measurement with the wider network. Furthermore, assurance agents work collaboratively to objectively verify NFPs.

In Chapter 7, we addressed the issue of reliability and assurance NFPs in big data analytics pipelines. We explored challenges and opportunities for

assurance in these pipelines, including data privacy and security, and code quality and security. We also presented our approach to continuous assurance in such a scenario through a template model, a generic representation of a big data analytics pipeline that captures its structure, functionality, and NFPs. We used the template model to specify and verify the NFPs of the pipeline, as well as to monitor and optimize the pipeline performance.

Chapter 8 provides a comprehensive investigation of the latest deployment infrastructures for the E2C continuum. This paradigm facilitates the smooth and dynamic provisioning oof applications and services across various layers of computing resources, ranging from the edge devices to the cloud servers. Furthermore, we examined analyse the integration of SLAs into these systems, allowing the specification of the quality and non-functional guarantees that service providers and consumers agree upon. Follows a detailed assessment of 5G's MEC, which currently represents the standard solution for application deployment at the 5G edge, as it provides low-latency, high-bandwidth, and context-aware services by leveraging the proximity and capabilities of edge nodes.

In Chapter 9, our methodology for achieving NFP in deploying applications in distributed multi-cloud environments is outlined, with a specific emphasis on the E2C continuum. Deploying applications in distributed multi-cloud environments presents numerous challenges due to environment heterogeneity, complexity, uncertainty and dynamism. Therefore, we introduced a new methodology that uses the concept of deployment matching to select and integrate the most appropriate deployment alternatives for every application component based on their NFP prerequisites and the features of the available resources. We provided an example of a data analysis pipeline and demonstrate how our solution, which takes into account the system properties, ensures conformity to user-defined Service Level Objectives (SLOs).

Chapter 10 outlines our practical experimental setting, in which we developed an edge-deployed big data engine to continuously collect and analyse data from an automated agronomy robotic platform. The big data engine, deployed on the edge, utilises the benefits of the E2C framework to deliver precise and timely insights for the field of agronomy. The IoT platform deployed on the field produces a significant quantity of data from various sensors and actuators. This data was collected and analysed by the big data engine deployed on the edge, before being made accessible to network users.

In Chapter 11, we presented the experiments and evaluations that validated our proposed methodology and assess its efficacy and performance in realistic conditions. Our methodology has been applied across various domains and scenarios, such as computing infrastructures, 5G edge networks, data analysis and application deployment compositions. We compared our methodology with those published in literature, where available, and examined the findings and their implications in detail.

Chapter 12 presents a list of further gaps and challenges that we identified during this research, which may be addressed in future work.

In Chapter 13, we conducted a literature review on the topics covered. Our aim is to provide a critical overview of current research, identify gaps and challenges in these areas, and contextualise the contribution of this thesis within the field. We also compared our approach with existing solutions and methods, and evaluate their advantages and limitations.

In the final chapter of this dissertation, Chapter 14, an impartial assessment of the results yielded from the experiments and evaluations is presented. The central contributions and discoveries of the research are summarised, while taking into account any restrictions and implications to the research methodology.

Chapter 2

Reference architecture

E2C architectures seek to address the challenges and opportunities presented by the increasing demand for low-latency, high-bandwidth and context-aware applications in various domains, such as smart cities, healthcare and Industry 4.0. However, E2C architectures encounter a number of gaps and challenges that require resolution, including how to guarantee the quality and performance of applications and services, how to handle the diversity and complexity of resources and environments, and how to deal with the unpredictability and dynamism of user and network conditions. This chapter presents a comprehensive overview of E2C architectures and their essential components. It identifies gaps and challenges in existing research and outlines the primary objectives and contributions of this thesis to address these issues.

2.1 Modern Edge-Cloud Infrastructures

Computing infrastructures are rapidly evolving to meet the growing demand for scalability, ubiquity and performance, as well as the need for reliable NFPs. This has led to the development of large-scale data centres and solutions collectively known as the "Cloud". Cloud providers offer users access to resources and services hosted in their data centres for a fee. Transferring data to and from users and data centres has numerous disadvantages, particularly cost and latency. The progress in network and computing management now

permits us to allocate some computation in smaller compute centres closer to the user that operate as intermediary points, hence the emergence of "Edge" nodes. This solution mitigates these issues at the cost of increased complexity. Data and requests can be preprocessed or resolved directly at the edge, avoiding the need to connect with the central data centre, improving the user experience. The increasing popularity of Fog computing, thanks to improved IoT devices and local networks, means that computation is now pervasive. Devices that require or provide computation capabilities surround us, and can communicate with each other, enabling the implementation of small-scale systems within end-users' residences. This creates an opportunity to quickly resolve requests in close proximity to the source. However, this requires significant management and standardization of each layer and the abstraction of resources. Figure 2.1 summarises the architecture components and how the nodes are connected. The available solutions employ competing

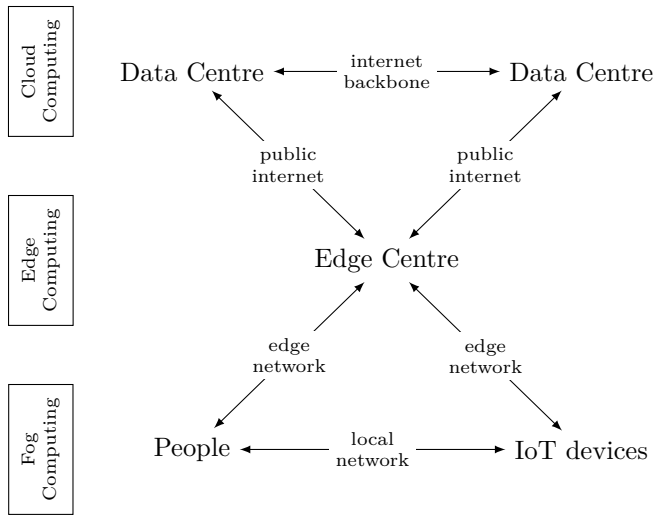


Figure 2.1: An overview of the E2C Continuum infrastructure.

standards and show considerable heterogeneity, yet generally comprise five macro components:

Computing The computing part of Cloud platforms entails all the hardware resources, software and interfaces necessary to execute processes. This

includes Central Processor Unit (CPU) cores, memory and dedicated hardware (e.s. GPUs, ASICs, FPGAs, hardware accelerators), but also the libraries and interfaces that provide access to these resources and regulate the applications execution. Cloud platforms provide their users with *on-demand* computing resources, ensuring high availability without limits (virtually). The approach to computing is more and more shifting towards immediate availability and scalability, with applications based on Function as a Service (FaaS) and managed containers that are scaled along with the increase and decrease of user demand.

Network Cloud platforms provide networking capabilities to their users. The most common service is high-speed connectivity between the allocated computing resources and the public Internet. Having control over edge-networks is also becoming more valuable, as Cloud providers can provide their services in edge nodes with stronger guarantees, such as latency and throughput, but also security and availability. The more established Cloud solutions are expanding their offering to include more advanced network solutions, such as CDNs, 5G mobile networks and satellite networks. We will explore these topics in detail respectively in Chapter 6 and sections 4.2 and 4.5.

Storage Cloud providers offer for storage are substantially divided in two types: block storage, where storage space is allocated and served as a disk device to the computing layer, and the newer object storage, where the data is modelled to the file level, allowing for transparent replication across multiple data centres and ease of access through RESTful Application Programming Interfaces (APIs). Block storage is the de facto standard for Cloud-hosted Virtual Private Server (VPS) and generally follow a one-to-one relationship when the host requires write access or one-to-many if only read access is necessary. Contrary, object store services can scatter the data in multiple clustered hosts, simplifying high availability guarantees and resulting in higher efficient use of the resources.

Monitoring Monitoring focuses on the collection of evidence on the behaviour of the system. In the past, logging was mostly limited to processes debug messages and ad hoc hardware monitoring (i.e. through Simple Network Management Protocol (SNMP) daemons). With the increase in automation and request for reliability it has become imperative to have more

in-depth knowledge about the system in order to identify misbehaviour and optimization processes. The market is pushing towards transparency of systems, using common standards of formatting and encoding, and is requesting more focused tools, supporting logging, span traces and metrics.

Management Automation distinguishes on-premises computing solutions from Cloud-based ones: providers manage the life-cycle of all data centre resources and services being provisioned to users in the Cloud. This relieves users of the management burden and creates common business models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and at the extreme, FaaS. This implies that the Cloud provider can take on the responsibilities for computing, networking, storage, and monitoring solutions. Requests for services and resources, as well as deployments, are managed through code-based configurations. We will delve further into this in Section 2.3.

2.2 The role of containerization and virtualisation

Decoupling of the application being deployed from the target infrastructure has a number of benefits: i) replicability of the execution environment, ii) isolation from other applications, iii) simplified deployment and its automation, iv) consistent resources requirements and limits. The two prominent technologies in this context are Virtual Machines (VMs) and Containers, each with its own peculiarity. As the industry standardized and integrated these solutions, frameworks for automating build, test and deployment rapidly gained traction for both these technologies.

Virtualisation A hypervisor provides resources to a guest system through a standardized abstraction layer, providing both a high level of control over the system and strong isolation of the running processes from the rest of the environment. VMs have a complete operating system (kernel, libraries, applications, …), which introduces some overhead in allocating and accessing resources. Portability of the VMs is tied to the implementation and support of standards by the hypervisor: different implementations may have different

levels of support. Applications deployed in VMs generally have middle to long life-cycle, as the boot and initialization of the system may take several minutes.

Containerization Containers have their own libraries and applications, but share the host kernel, so isolation is reduced compared to VM. Security can still be enforced using namespaces, cgroups and seccomp, minimizing the permissions of the process and thus the attack surface. Containerization also has a lower resource overhead, closer to running native processes on the host system, and generally does not require preallocation. Containers are more portable than VM, thanks to the strong work of standardization of their interface and the loose coupling with the underlying operating system. Portability is reduced if the containerized application relies on features of the host operating system, such as vendor-specific libraries (e.g. NVIDIA CUDA[1]). Containers have a very short initialization time, typically less than a second, making them suitable for the FaaS use case and rapid scaling.

Security Both virtualisation hypervisors and container engines have advanced security features to guarantee strong isolation between processes running on the same system. Nonetheless, there have been examples of attacks allowing an attacker that has access to a compromised application to gain control over the host, respectively VM escape [1] and container escape [2]. Moreover, the literature registers several families of attacks against supposedly isolated processes, including but not limited to side-channel attacks [3] and path prediction attacks [4, 5]. Therefore, we should carefully consider how possibly insecure or untrusted applications are being executed in shared hosts.

2.3 The importance of Deployment

Deployment of applications and services in the E2C continuum is especially challenging due to the several additional domains to be considered: the distributed nature of the system introduces reliability issues and latency; the different execution environment increase heterogeneity; the consistency of

[1] `https://developer.nvidia.com/cuda-toolkit`

data and configurations in geographically distant locations. The drive to automate deployment and infrastructure configuration has given rise to application lifecycle management solutions. These take code-based configuration files as input and manage all aspects of the deployment, such as resource allocation, network configuration, logging and secrets management facilities, user and application authentication. The most notable example of these platforms is Kubernetes[2], a highly scalable and extensible container management solution commonly used in E2C environments to run highly available applications. It uses a distributed control plane to monitor and manage the underlying hardware, and to schedule the deployment of applications and any additional services they may require. This includes automatic configuration of networks and network services such as Domain Name System (DNS), Virtual Private Networks (VPNs) and load balancers. The configurations provided by the user are stored in a distributed configuration repository, ETCD[3], and act as a target for deployment. The Kubernetes controller acts on the running systems to achieve the target configuration, e.g. by monitoring process lifecycle, spawning containers, allocating resources. Complex deployments are often packaged in charts, deployment templates that offer minimal working examples and can be customized for the specific use, making sharing of configurations easier and more replicable.

A more recent trend in this context is intent-based deployment: similar to configuration-based deployment described above, it uses code to describe how the system is to be deployed and configured, but in this case the user also requires guarantees about the feasibility of the deployment and about NFPs the system. These may include a network latency threshold, bandwidth, availability or geographical (or network) locality. To achieve this new level of control, the deployment controller checks the feasibility of the new configuration before applying it. If the new target state is not achievable, it will initiate a negotiation with the user, lowering the requirements, until a suitable middle ground is reached, or the user rejects all alternatives.

[2] `https://kubernetes.io`
[3] `https://etcd.io`

2.4 Gaps and challenges

Currently, the solutions available for the E2C continuum have reached a good level of maturity, but there are still gaps that must be addressed before making these technologies accessible to the wider public.

G_1 **Non-Functional Property** For the last few decades we mostly concentrated on performance NFPs such as service latency or network throughput. This has led us having a huge number of metrics to evaluate the performance quality of our systems and an equally large number of property definitions. Other NFPs such as security, privacy and locality, are less commonly formalized, with ad hoc definitions that are difficult to evaluate and compare. This is a significant problem when trying to implement a verification system, resulting in ambiguous requirements, duplication and undefined behaviour. Additionally, there is a shortage of a uniform property framework that would enable Certification Authorities (CAs) to specify metrics and contracts through a standardized interface.

G_2 **Strong reliance on infrastructure** Applications deployed in Cloud or E2C environments increasingly rely on the infrastructure that provides both the necessary resources for their operations and the NFPs defined through SLAs. This simplifies the work of developers, who can think in terms of abstractions of the underlying levels. However, it also introduces coupling and reliance on the infrastructure implementation, potentially leading to unsatisfied properties and vendor lock-in.

G_3 **Transparent monitoring** Components included in a deployment, ranging from the lower levels of the infrastructure up to services, typically provide some form of monitoring for their status. The three types of monitoring utilized by state-of-the-art solutions are i) logs, straightforward time-tracked text messages, ii) traces, which are context-aware logs that record the stack of execution of a specific part of the application, including function arguments, and iii) metrics, time-tracked values that describe particular aspect of the application. Most applications currently only provide monitoring based on logs, which significantly hinders our ability to understand their internal state. Additionally, logs are primarily designed for human-readability rather

than machine interpretation and lack common encoding standards. However, there has been a recent push to unify monitoring technologies, particularly for Cloud-hosted services, enabling informed service management. Nonetheless, the adoption of de facto industry standard solutions, such as Prometheus and OpenTelemetry, is still fairly limited.

G_4 **Multi-layer assurance** Assurance techniques in literature typically concentrate on just one layer of the deployment stack, with some focusing solely on the service layer, others on the infrastructure layer, and still others on the networking layer. This approach relegates other levels to black boxes, significantly curtailing the information sources employed in property verification and impairing its effectiveness. This is mainly attributable to the increasingly large and complex process of verifying and certifying the entire stack. While some preliminary research on multi-layer assurance is available in literature, its adoption remains limited.

G_5 **Deployment automation** Automation of deployments is a crucial factor step in the growth and advancement of E2C solutions. Presently, numerous systems already offer automation, which can take the form of configurations-based or intent-based deployment specifications. Kubernetes, an industry standard mainly supports configurations-based automation. Intent-based deployment are still in the early stages of development, with some standardized implementation currently available, such as Open Source MANO (OSM)[4]. However, these implementations are not yet widely used and lack a common standard.

G_6 **Failure during assurance** The assurance process involves multiple steps that could experience unexpected failures, such as unattainable targets for metrics calculations and logic errors in contracts that lead to early termination. Many methodologies for continuous assurance and certification of systems do not specify or clarify the handling of failures. Additionally, verification is often affected by the sense of uncertainty. In case of failure, it may be unclear which level of the system is causing the malfunction, whether it is the network, the service, or the communication medium.

[4] `osm.etsi.org`

Chapter 3

Infrastructure assurance

This chapter details our approach for ensuring the NFPs of the infrastructure that comprises the E2C continuum. This chapter details our approach for ensuring the NFPs of the infrastructure that comprises the E2C continuum and platform supporting computations in the continuum. Our methodology is based on the continuous verification and validation that the infrastructure meets the SLAs specified by the service provider and the consumers. The procedure for assurance of infrastructures includes the following steps:

Infrastructure Modelling A formal and comprehensive model is defined for the infrastructure, covering its service interfaces, functionalities, components, configuration and state, and monitoring endpoints.

Assurance Methodology We propose a methodology for measuring and evaluating NFPs of the infrastructure, based on evidence collected through the monitoring endpoints of the services. We utilise contracts to formalise property verification and test the collected evidence against them.

Assurance Process We detail a systematic and automated approach to infrastructure assurance, comprising four stages: evaluation mapping, measurements collection, contracts evaluation, and report composition.

The primary aim of this chapter is to offer a comprehensive overview of our infrastructure assurance approach, focusing on its significant traits, benefits, difficulties, and limitations. In addition, we aim to demonstrate, through the usage of examples and scenarios, how our approach can be practically adopted.

3.1 Our Assurance Approach at a Glance

Our infrastructure level assurance approach is founded on a model of the target infrastructure that abstracts implementation details, enabling us to target different versions of the operating infrastructure. Additionally, we rely on an assurance methodology that facilitates continuous and consistent measurement collection from the operating infrastructure to assess its NFPs. Compared to traditional assurance processes, our process does not require invasive active agents to be deployed in the third-party infrastructures. It is also capable of addressing the typical NFPs of interest in modern E2C continuum infrastructures. Specifically, our assurance process is guided by the NFP that requires verification. We model the NFPs **P** as families (e.g., confidentiality) grouping specific types (e.g., confidentiality of the transmission).

In the following, we present the set of relevant property families that we will consider in this work:

- **Access control** Only authorized users can access resources, data, and services, in accordance with administrator-defined policies.
- **Automation** The degree of automation indicates the percentage of system processes that do not require human intervention. This encompasses activities such as configuring network setups, managing service deployments, and administering users, among others.
- **Availability** The system's availability is defined as its capacity to handle user requests.
- **Confidentiality** The confidentiality property indicates the protection of information from disclosure.

- **Integrity** We indicate the integrity of a system as its capability of retaining consistency of the data and of its internal state.
- **Locality** We refer to the concept of locality in the context of data, computation, or message dissemination.
- **Performance** We define performance characteristics as the system's capability to meet users' demands within a restricted time frame.
- **Resilience** It refers to a system's ability to remain partially functional, even in the event of catastrophic failures.

We note that the NFPs are generic and do not provide guidance regarding their verification. Our assurance methodology assumes responsibility for this task.

Example 3.1.1. Let us consider a simplified scenario in which an advanced 5G infrastructure provides on the MEC the ability to store binary data objects through a RESTful API, while offering token-based access control. This enables 5G users to efficiently store and retrieve small data pieces, enhancing their experience with other applications on their device. The storage service offers not only storage functionalities but also advanced NFPs, such as low latency data access. For simplicity, let us define the NFP $p_{data\ conf}$ = *Data Confidentiality* of a data object as the prevention of unauthorized users from accessing the stored data.

In the following, we present our infrastructure modelling in Section 3.2 and assurance methodology in Section 3.3 considering Example 3.1.1 as the simplified target scenario.

3.2 Infrastructure Modelling

Figure 3.1 shows how we model modern assurance in continuum infrastructures. The purpose of this model is to ensure that the assurance process remains independent of technical implementation differences at the infrastructure level. In this model we consider infrastructure as made of several

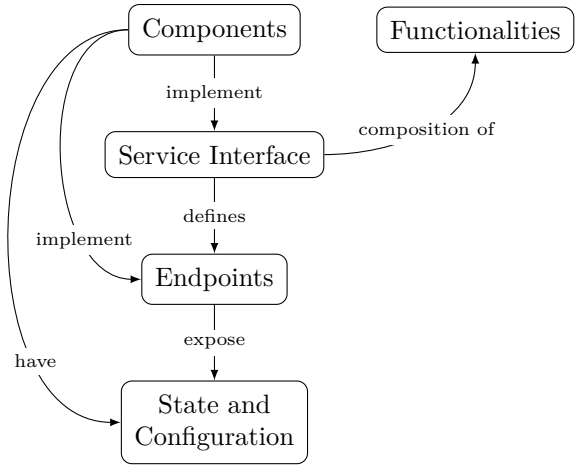


Figure 3.1: A model for modern E2C infrastructures.

functionalities grouped to form *service interfaces*. Here the word Service follows the more general meaning of meeting a need, rather than the more typical of network service. Service interfaces, eventually describing a specific protocol (e.g., O-Radio Access Network (RAN)), are implemented by one or more *components*, including hardware devices or software services, that collaborate to fulfil the interface requirements. Components are entities that have an internal dynamic *state* and a static *configuration* which defines their behaviour. State and configuration are exposed by the component to our assurance process via monitoring *endpoints* (see Section 3.2.3). We note that endpoints can be natively present in the components (e.g., access to configuration files) or may need to the implemented by extending them (e.g., latency probing).

3.2.1 Service Interfaces and Functionalities

A *service interface*, denoted by $s \in \mathbf{S}$, is a collection of functionalities, represented as $f \in \mathbf{F}$, which are composed to satisfy a protocol used to call the interface s. Each functionality is a minimal executable action and defines input, output, constraints and data formats. We note that different service interfaces s_i may have overlapping functionalities ($s_1 \cap s_2 \neq \varnothing$) or be a subset

of another service interface ($s_1 \subseteq s_2$).

Example 3.2.1. Let us consider a service interface s_{S3} for the object storage in Example 3.1.1 based on the Simple Storage Service (S3) object storage protocol[1]. s_{S3} defines the service protocol following the S3 specification and identifies a set of functionalities $\{f_{S3GetObject}, f_{S3PutObject}, f_{S3GetBucketPolicy}, \ldots\}$. The protocol establishes how the functionalities can be accessed. For instance S3, mandates to check the ownership policy applied to the storage location (bucket) before providing access to its contents. At functional level, constraints on arguments and outputs can be specified. For instance, $f_{S3GetBucketPolicy}$ requires a valid bucket ID as input and returns a configuration entry containing a list of resources that the user can access and available actions.

3.2.2 Components

A component $c \in \mathbf{C}$ is a software or hardware item that, alone or collaborating with other components, implements a service interface. The implementation relationship $\mathbb{I}$ is a mapping from sets of components to sets of service interfaces as in $\mathbb{I}_{\wp}(\mathbf{C}) \mapsto \wp(\mathbf{S})$. We say that the set of components $\mathbf{c} \subseteq \mathbf{C}$ implements a certain service $s \in \mathbf{S}$ if and only if $s \in \mathbb{I}(\mathbf{c})$, which also means that $\mathbf{c}$ implements all the functionalities in $s = \{f_1, \ldots, f_n\}$.

We may have multiple sets of components, possibly disjointed, implementing the same set of functionalities. In this case $\mathbf{c}_1, \mathbf{c}_2 \subseteq \mathbf{C}$ and $s \in \mathbf{S}$ we have that $s \subseteq \mathbb{I}(\mathbf{c}_1) \cap \mathbb{I}(\mathbf{c}_2)$ and $\mathbf{c}_1$ and $\mathbf{c}_2$ are said to be interchangeable for s. Similarly, a certain $\mathbf{c} \in \mathbf{C}$ may implement multiple service interfaces at the same time, in this case for $s_1, s_2 \in \mathbf{S}$ we may have that $s_1 \cup s_2 \subseteq \mathbb{I}(\mathbf{c})$.

Example 3.2.2. Let us consider Example 3.2.1, the service interface s_{S3} requires combination of the three components $s_{S3} \subseteq \mathbb{I}(\{c_a, c_n, c_s\})$ as follows. A server-level component c_a implementing the RESTful API for S3 protocol including every functionality f requested by s_{S3}, network-level component c_n providing network connectivity, and a storage component c_s providing allocable storage space. We note that although components c_n and c_s do not directly implement functionalities f in s_{S3}, they are needed to implement the protocol of service interface s_{S3}.

[1] `https://docs.aws.amazon.com/AmazonS3/latest/API`

3.2.3 Configurations, states and endpoints

Each component has an associated configuration and an internal state. The configuration is static and describes the component's context and its execution environment (e.g., bootstrap initialization, environment variables). The state of a component changes dynamically over time and includes information such as variables status and resource usage. Both configurations $\mathcal{C}$ and states $\mathcal{S}$ are mappings $K \mapsto V$ where K is a set of unique keywords and $V \subseteq (\mathbb{B} \cup \mathbb{I} \cup \mathbb{R} \cup \mathbb{S})$ is a set of boolean, integer, real or string values. Infrastructure components normally expose standardized monitoring interfaces (e.g., Syslog, OpenTelemetry[2], Prometheus[3]) to access internal state and configurations for audit purposes. In this study, we refer to these monitoring interfaces as *endpoints* since they are the primary target of our assurance process for gathering measurements from the infrastructure.

Example 3.2.3. Let us consider Example 3.2.2 and the server-level component c_a only. It has a configuration that includes rules on network port bindings, storage paths and data replication. Such a configuration can be exposed by the application through standard monitoring endpoints or extracted by ad hoc probes.

The internal state of the server is exposed through a logs and state collector protocol (e.g., OpenTelemetry or Prometheus), allowing for dynamic monitoring. For instance, let us consider c_a, the relative endpoints can expose the following measures of the components' behaviour: i) $e_{health\ check}(time)$: health check to verify the component reachability; ii) $e_{acl}(time)$: the map of keys that have access to any bucket;. Both endpoints are implemented using a RESTful API.

Using standard monitoring techniques and a unified framework of endpoint specifications facilitates towards a more transparent and auditable generation of E2C solutions. This approach provides a solution to Gap G_3 *Transparent monitoring*. We note that, in this work we assume that every component exposes specific endpoints for our assurance process. We also note that, this work contributed in defining endpoints for standard components that are nowadays not exposing them (e.g., some 5G core network components).

[2] `https://opentelemetry.io`
[3] `https://prometheus.io`

3.3 Assurance Methodology

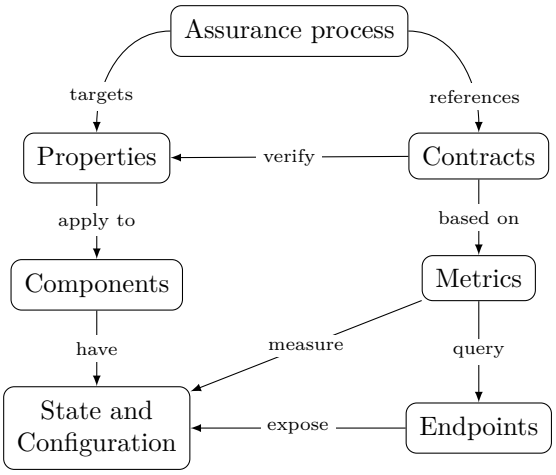


Figure 3.2: A schema of our assurance methodology.

Figure 3.2 shows our assurance methodology based on the infrastructure modelling in Section 3.2. More specifically our assurance methodology uses the infrastructures endpoints as its primary source of measurements. For instance, an infrastructure component can expose endpoints offering a dynamic measure of a volume's available space and a static measure of the space originally allocated to the volume. Endpoints measurements, are aggregated into metrics, that are focused on evaluating a specific behaviour which is suitable for non-functional *property* verification (e.g., uptime metric to verify the availability NFP). Metrics are themselves aggregated into *contracts*, which refer directly to a specific NFP and describe how it can be verified in terms of metric values.

3.3.1 Metrics

A *metric* is a function m defined by a functionality that aims to measure its aspects or behaviours on the components that implement it. Metrics are defined over the states and configurations exposed by the monitoring endpoints. The evaluation of metrics provides the necessary evidence confirm

if a particular property holds.

Example 3.3.1. Let us consider the monitoring endpoint e_{acl} from Example 3.2.3 and a metric $m_{obj\ perm}(time, object)$ that measures the access permissions of a given object. To define $m_{obj\ perm}$, we require data from the e_{acl} endpoint at instant $time$. We can then define the target metric as follows.

$$
\begin{aligned}
&m_{obj\ perm}(time, object) = let \\
&\quad bucket = getBucket(object); \\
&\quad acl = e_{acl}(time); \\
&in\ getPermissions(acl, bucket)
\end{aligned}
$$

First the metric calculates the name of the bucket that contains *object*, then the Access Control List (ACL) is retrieved and finally object's permissions is extracted.

3.3.2 Contracts

Contracts are boolean functions that formally describe how to validate properties of services based on evidence obtained through metric evaluations of the components monitoring endpoints. Contracts are of the form $contract(time, \mathbf{c})$, where $\mathbf{c} \subseteq \mathbf{C}$ is a set of components that implement a target service interface and $time$ is the instance of evaluation. To ensure consistency, we aim to have only one contract for each property. In general, we want to define contracts that are time-dependent, so that their outcome varies depending on when they are valued. Follows that the time instant in input to the contract influences the metric evaluations that are time-dependent, as shown in Example 3.3.2.

Example 3.3.2. Let us consider the property of data confidentiality $p_{data\ conf}$ and the metric $m_{obj\ perm}$ from Example 3.3.1. We define a simple contract that verifies whether the components $\mathbf{c}$ implementing the service s_{S3} have the property $p_{data\ conf}$ of data confidentiality for the data object *object* at the instant $time$ as shown in Equation 3.1.

$$
\begin{aligned}
&p_{data\ conf}(time, \mathbf{c}, object, expected) = let \\
&\quad allowed = m_{obj\ perm}(time, \mathbf{c}, object); \\
&in\ allowed \leqslant expected
\end{aligned}
\tag{3.1}
$$

Notice how the contract definition is parametric, allowing us to check whether $p_{data\ conf}$ is true with different *expected* arguments. In addition, we do not set any limit on the metrics' time of evaluation, allowing us to express even cases where measurements are predicted in the future.

Contracts address the Gap G_1 *Non-Functional Property* by providing a framework for formal and unambiguous framework for defining properties and verifying their implementation. Moreover, the generality of metrics and contracts allows to effectively define multi-layer properties, addressing also Gap G_4 *Multi-layer assurance*.

3.4 Assurance Process

An assurance process employs an assurance methodology to furnish its users with the requisite tools and frameworks to validate properties. The solution we provide supports continuous and collaborative verification of NFPs of infrastructure components, extending our previous work [6] and focusing on the broader target of the continuum infrastructures.

The assurance process, shown in Figure 3.3, implements the methodology described in Section 3.2 in four steps:

Evaluation mapping the first step involves constructing the dependency graph of the evaluation. This includes identifying the metrics and contracts that need evaluation, beginning with the list of targeted properties. To achieve this, contracts for each target property are located by reversing the V mapping. Then, any potential duplicate evaluations are eliminated by enumerating each mentioned metric and its corresponding arguments.

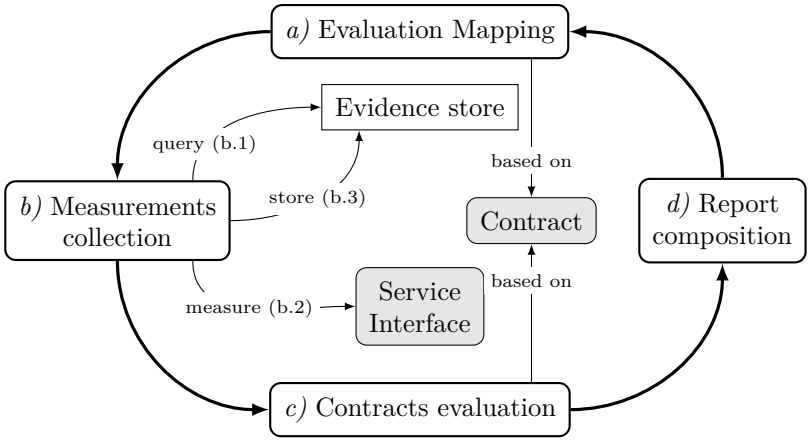


Figure 3.3: A graphical representation of the assurance process. Steps *a* through *d* of the process are repeated cyclically. In this example, Component 1 is retro-fitted to the Service Interface using a probe.

Measurements collection the next step involves collecting measurements of the metrics listed. The services' defined interface dictates the means of collection, such as exposing metrics through a RESTful API or periodically transmitting health check messages to a queue service channel. Metrics can be evaluated either on-demand or periodically. On-demand evaluation is more resource-efficient, but it may introduce latency when calculating the metric is expensive. Periodic evaluation, on the other hand, can be effective when retaining the measurement for the entire period does not affect the quality of the result.

Contracts evaluation the third step performs the actual property verification by computing the outcome of each contract using the previously collected measurements and producing a boolean output. The map generated during the *Evaluation mapping* step can be utilised to create a priority queue for efficient contract evaluation.

Report composition finally, the last step compiles the partial results, which comprise both measurements and contract evaluations. A report is

generated that includes a description of whether each property has been verified and the corresponding supporting evidence. This report uses a predetermined machine-readable format for use in other automated processes, such as certification.

The methodology proposed for the assurance process is based on static evaluation, aiming for independent evaluation steps regardless of the system's state during evaluation. This is achieved by operating on measurements collected and stored in a time-indexed temporary storage instead of using direct measurements. This separation enables time-range based contracts and consistent replicability in their evaluation.

The measurements collection and contracts evaluation steps are considered as "impure" in the assurance process due to the potential unforseen behavior during their execution. Such issues include metric implementation problems, environment failures, and logical mistakes in contracts. As a consequence, it is necessary to establish a default course of action for managing failure scenarios. This protocol may involve reverting to a probabilistic estimate of outcomes, producing an "undecidable" result, or terminating the evaluation altogether.

We previously demonstrated that a distributed and collaborative pull-based implementation of the assurance process is highly effective [6]. This is due to the ability to cache and reuse previous measurements and evaluations, even partially, in newer verification evaluations.

Finally, the mapping of multiple contracts to the same property can result in duplication and inconsistencies. For simplicity, a bijective mapping is employed in this work. The more general case necessitates the handling of these edge cases.

Chapter 4

Telco Edge Networks

Nowadays, there is a growing trend in the quantity and speed of data, commonly utilized by data-intensive Artificial Intelligence (AI)/ML-based services, necessitating wider dissemination of more efficient Edge computing techniques. Furthermore, there is a rise in the usage of critical applications which involve vast amounts of sensitive data, calling for enhanced security and privacy safeguards. 5G Edge technology could promote wider Edge computing adoption, although interoperability hurdles persist. Private edge networks and computing infrastructures, such as those enabled by 5G and satellite, are capable of providing the necessary security and privacy assurances for such applications whilst also delivering high performance and availability. This enables remote users to access E2C computing solutions, connecting them to Edge and Cloud data centres.

These networks have few fundamental traits, which are summarized in Table 4.1. Optical fibre connections are faster compared to other alternatives but have limited coverage, especially in rural areas. 5G networks have significantly higher coverage and maintain a high level of throughput. In contrast, satellite networks have higher latency and can only achieve lower speeds, however their widespread coverage enables them to reach users even in remote areas. The Fibre to the Home (FTTH) Council Europe's report from April 2023 reveals that 62.2% of the European population and 55.5% of Italians were covered by FTTH [7]. The April 2023 report from the European 5G observatory indicates that 81% of the European population and 99.7% of the Italians have access to 5G coverage [8]. Optical fibre networks have served as

Table 4.1: Comparison table of network features.

Feature	Optical Fibre	5G	Satellite
Latency	0.003 ms	1 ms	30 ms (LEO) to 600 ms (GEO)
Throughput	100 Gb/s	20 Gb/s	0.5 Gb/s
Availability	Limited coverage, especially in rural and remote areas	High coverage. Limited coverage in rural and remote areas or indoors spaces.	Global high coverage
Reliability	Stable performance	Affected by radio interferences, trade-off between reliability and speed	Affected by radio interferences, generally high reliability
Automation	Reliable network configuration automation techniques	Reliable network configuration automation techniques. Early automation techniques for computation	Reliable network configuration automation techniques. Early automation techniques for computation

a primary medium for the Internet backbone for decades and as the principal connection type between data centres. Contrary, 5G and satellite networks have only recently emerged as viable solutions for edge computing, placing them at a disadvantage in the realm of automated computation deployment. Nevertheless, their solutions are rapidly maturing.

4.1 Edge Computing

Nowadays many distributed applications, including many IoT, FaaS and PaaS frameworks, mostly reside in the central Cloud to ease the integration with existing service platforms. Edge computing brings new opportunities allowing proximity computing and better match with the need for distribution, but also new challenges such as integration/orchestrations with services in Cloud or in other Edge nodes. Edge computing enables new generation applications to work close to the user allowing i) low-latency access to user plane traffic, ii) context-aware adaptation of application behaviour depending on the user environment. Edge also provides a positive side effect on reducing the traffic to the remote server (e.g., in the Cloud) avoiding congestion in scenarios where a centralized entity is involved and therefore potentially improving the performance. This aspect is very relevant for data-intensive applications, where the network can be easily saturated due to the intense data exchange. Advanced Edge computing solutions also allow deploying services in specific geographical areas in order, for instance, to provide resilience in case of emergency on the centralized nodes or differentiate services based on the location context.

4.2 5G architecture

The 5G architecture deeply relies on network virtualisation to create flexible and on-demand instances of functional networking entities. To facilitate the deployment over a virtual infrastructure, it replaces the point-to-point interfaces used in 3G and 4G with producer/consumer-based communication among 5G core network functions. Thanks to this architecture, 5G allows to scale up and down and with tight control of all network resources, for instance through virtual network slices designed specifically on service requirements and dynamically deployed on need.

In the following, we describe the core network components of the 5G architecture (see Figure 4.1) as defined in [9].

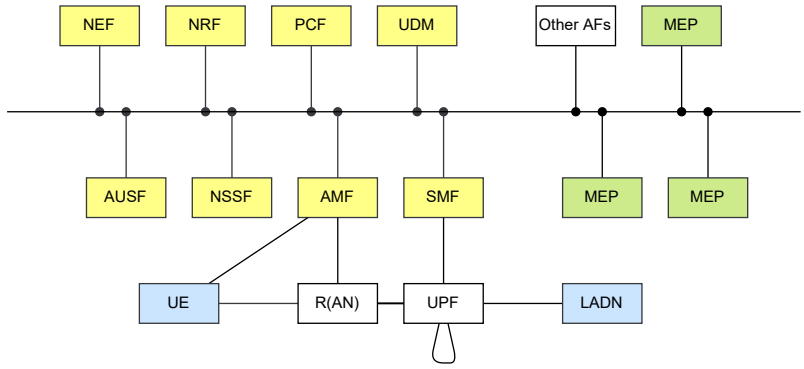


Figure 4.1: 5G Core System Architecture Network Functions.

Network Function (NF) The core network of the 5G system is composed of NFs that interact with each other exposing services to the network itself and interfacing with the outside. These services are generally deployed in a container-based Cloud-native solution like Kubernetes, which provides additional features like automatic replication, auto-healing and monitoring.

Authentication Server Function (AUSF) Authentication Server Function (AUSF) is the network function that implements the Extensible Authentication Protocol (EAP) authentication server used for secure operations in combination with the Access and mobility Management Function (AMF). It also stores the keys used by the AMF providing security and integrity features.

Network Slice Selection Function (NSSF) In 5G all traffic is relegated to a set of slices depending on its intrinsic characteristics, i.e. massive IoT slices with a large number of connections but low bandwidth, mission-critical slices with very low latency and increased reliability, to name but a few. Network Slice Selection Function (NSSF) manages the network slicing. It ensures that the resources required for each slice are available, to match the defined SLAs to best suit the demand of the target application. The NSSF selects the network slice to serve to each connected User Equipment (UE), selects the most appropriate Network Slice Selection Assistance Information (NSSAI) for each device, mapping it to its virtualised network interface, and

identifies the appropriate AMF set based on the UE subscription information and location of the device.

Network Repository Function (NRF) Network Repository Function (NRF) provides service discovery functionalities in the core network for the Service Communication Proxy (SCP) and other NFs responding to discovery requests and notifying any update in the available services. It also stores information about the health of the registered services and the available or mapped network slices.

Application Function (AF) An Application Function (AF) is an application that has access to the services of the core network. These are services that are deployed in the same container solution and that provide the functionality to the network (i.e., voice or Rich Communication Services (RCS) chats). AFs can access the Network Exposure Function (NEF), interact with the Policy Control Function (PCF)'s policy service and the network traffic. Depending on the trust the provider has in the function, the AF may access the network traffic directly or through the NEF.

Access and mobility Management Function (AMF) AMF is the NF that manages the interaction of the UEs on a hardware level. It handles signalling, security control, handovers, idle mode UE reachability, to name but a few. It also supports network slicing, being queried by the NSSF for availability and capabilities. AMF can represent a bottleneck of the mobility network control plane handling the UE requests. The VM hosting AMF should be dimensioned in order to avoid performance degradation and should be handled with elasticity [10].

Unified Data Management (UDM) The Unified Data Management (UDM) manages all the data regarding users and shared state between NFs. It is responsible for authentication key generation, identification, access authorization, exposing subscription data to other NFs, session continuity, subscription and Short Message Service (SMS) management, to name but a few.

Unified Data Repository (UDR) The Unified Data Repository (UDR) handles the storage and retrieval of data on behalf of UDM and PCF. It is responsible for the integrity and availability of such data throughout the network.

Session Management Function (SMF) The Session Management Function (SMF) is the NF that is in charge of the users' sessions. It collaborates with the AMF to handle signalling with the UE, allocating Internet Protocol (IP) addresses and selecting the most appropriate User Plane Function (UPF). The SMF is the core network component that collects and enforces policies provided by the PCF, tracks usage and Quality of Service (QoS) of the network and manages its configurations.

Policy Control Function (PCF) PCF is the function dedicated to the management and enforcement of security policies on the network. It provides a unified network policy framework that arranges the network behaviour and the management of the policy rules that are enforced on the network by the Control Plane (CP) functions. It also reads the information linked to the user subscription from the UDR in order to formulate relevant policy decisions and send the policies to the SMF (e.g., management of traffic and other User Plane (UP) functions).

User Plane Function (UPF) UPF is the data plane function that handles the user's network traffic. It receives signals for the SMF for allocating IP addresses and prefixes. It routes and forwards the packets, both within the 5G Virtualised Network (VN) group or to the external networks (i.e., the Internet). It enforces the policies provided by the PCF, QoS levels and traffic speed limits. It reports traffic usage and provides common network functionalities (e.g., Address Resolution Protocol (ARP), packed duplication).

4.2.1 Mobile Edge in 5G

In the typical mobile network scenario, the Data Network (DN) where the application resides is located very far from the device requesting the access. This leads to issues in guaranteeing a specific QoS and limited use-cases.

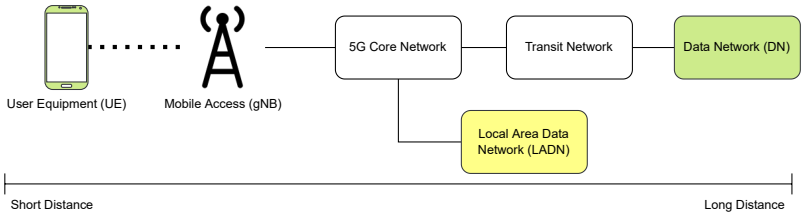


Figure 4.2: The Edge computing concept in 5G.

In the traditional network, this was solved via edge computing and using a more flexible deployment approach for the application allowing complete or partial deployment on the edge node. In 5G this notion of edge computation is supported by the notion of Local Area Data Network (LADN) connected to the 5G code network implementing the concept of MEC (see Figure 4.2). According to ETSI, MEC is a service environment at the edge of a mobile network within the RAN and in close proximity to the mobile subscribers [11] and is defined in [12]. The 5G core network can intelligently be aware of the presence of the application in the telco edge thanks to the 5G UPF routing the request to the edge application. To do this, UPF communicates with AF and AF provides back feedback on where to steer the packets. The MEC deployment manages the local area data network and communicates with the PCF to request traffic routing by identifying user traffic to be routed. PCF transforms the request into a policy affecting the user Protocol Data Unit (PDU) session and provides the routing rules to the SMF. The SMF identified the UPF present in the PDU of the user. UPF will then be able to route the user data to the relevant local area data network.

Figure 4.3 shows a more abstract view of the MEC application interaction with the rest of the mobile network via what is called Mobile Edge Platform (MEP) having an interface for MEC applications to expose and consume MEC services (Mp1), and another to interact with the mobile network (Mp2). In the 5G architecture, the MEP is integrated as a 5G AF [13]. A MEC application can register to expose a service (e.g., streaming processing) to other MEC applications using the MEP interface. The life-cycle of the MEC application is handled by Mobile Edge Application Orchestrator (MEAO) acting as an interface between MEC and Operational Support System (OSS)/Business Support System (BSS). The MEAO relies on Network Function Virtualisation (NFV) for handling MEP and MEC so that instantiation and management of

relative resources follow the NFV interfaces. The MEP and MEC applications are described using a Virtualised Network Function Descriptor (VNFD) and Application Descriptors (AppDs), having the scope to provide the necessary information for the Virtualised Network Function Orchestrator (VNFO) and Virtualisation Infrastructure Manager (VIM) to deploy instances of virtual applications (in Clouds or at the edge).

Slicing and Edge NS is the 5G technique to share common physical infrastructure across multiple users providing virtual networks tailored to the service/application needs. Slicing and Edge in 5G share the same objective to support low latency for time-critical services (URLLC services). Ksentini et al., [14] envisioned two deployments approach for MEP: i) MEP in multi-tenancy where MEP is already deployed by the network operator as an AF for all slices or ii) in slice deployment, where MEP is deployed along with MEC application. In the case of MEP in slice, MEC can provide services only to other MEC in the same slice. In the multi-tenant MEP case, the new MEC service should be advertised both to MEAO and Communication Service Management Function (CSMF), which needs to include it in their available function catalogues. In slice MEP provides better isolation but it is more costly since it requires one MEP per slice.

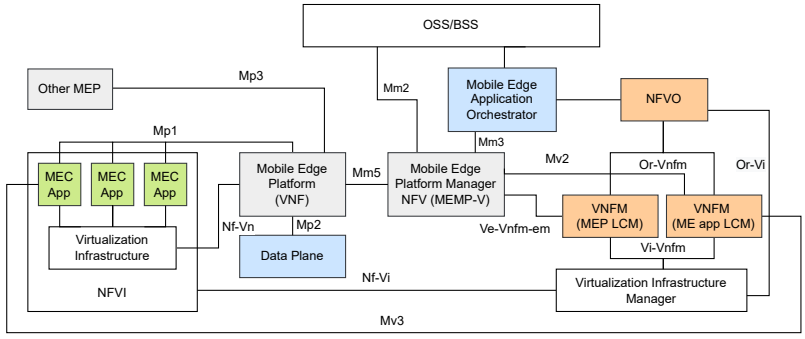


Figure 4.3: MEP infrastructure based on the NFV MANagement and Orchestration (MANO) architecture.

4.2.2 Security and Privacy in 5G

5G security is addressed, in general, by defining and applying security policies across the entire network using the Network Function PCF (5G System by 3GPP standards - working group SA3). Many international standardization bodies are contributing to the 5G security including International Telecommunication Union (ITU), ETSI, Internet Engineering Task Force (IETF), Next Generation Mobile Networks (NGMN), 5G Infrastructure Public Private Partnership (5G-PPP), National Institute of Standards and Technology (NIST), GSM Association (GSMA), and some of them focused on specific use case verticals such as mission-critical systems. In addition to standardization bodies, academic research projects such as the 5G-ENSURE (part of the 5G-PPP) contributed to the 5G security architecture, providing a set of security design principles and a set of security functions and mechanisms to implement security controls necessary to achieve the desired security objectives. The 5G System (5GS) architecture (see Section 4.2) was extended to support network data analysis (e.g., workload, QoS, anomalies) via Network Data Analytics Function (NWDAF). The 5G security may subscribe to NWDAF network analysis notifications and use them to compute policy updates. This can be seen as feasible support for security assurance and incident response. In the case of security assurance advanced monitoring methodologies are requested to monitor the network behaviour and find weaknesses and unexpected behaviours resulting from security leakages. In terms of security incidents. An incident detected by NWDAF can be used to trigger changes at the policy level via PCF. For instance, it can be possible to move users from a potentially compromised section or slice to a quarantine one to carry out further investigations. Security and privacy in MEC are still open challenges [15, 16]. The current literature provides only a limited overview and many aspects, such as authentications between MEC and Core network components, remain quite unexplored [17]. According to Ksentini et al., [14], while slicing is involved in MEC, other security and privacy issues arise. In MEC slicing the traffic redirection should preserve privacy meaning that NS cannot specify a traffic redirection policy for traffic that it does not own. In addition, the NS should not be able to use MEC service to get unauthorized access to information about another NS or consume MEC services that are not available for the given NS. Other security issues may arise if an application declares in its AppD specific traffic rules or DNS-related rules allowing traffic offloading. A malicious app can be able to intercept traffic flows causing confidentiality breaches and perform a kind of black hole denial of service by dropping pack-

ets. According to Ksentini et al., [14], Network Slice Subnet Management Function (NSSMF) should be augmented with security features and access control to check the permission to ask for a traffic redirection as in their AppD (e.g., using Public Key Infrastructure (PKI) technologies). Another issue is the exposition of privacy-sensitive data by the MEC such as location and channel quality. Ksentini et al., [14] suggested that for multi-tenancy MEP, it is needed to check the MEC authorization to access specific services providing potentially sensible information.

4.2.3 O-RAN Alliance

An important effort to reshape the Radio Access Network industry is put forward by the Open-Radio Access Network (O-RAN) Alliance[1]. O-RAN Alliance is a global consortium comprising mobile network operators, vendors, and research institutions working in the Radio Access Network field. Its primary objective is to improve the enhance RAN deployment and operation efficiency for mobile operators[2]. O-RAN standards aim to deliver comprehensive ML applications to RANs to enable data-driven control.

4.3 5G Network Simulator

One of the key objectives of my research has been to implement and deploy a comprehensive and functional 5G network simulator that can be used for research purposes in both current and future works. To achieve this, we established the following requisites: i) the simulator must employ open-source software to allow us to analyse, alter and publish alterations to the original code; ii) it must conform with the existing open standards (O-RAN and the most up-to-date 3GPP 5G revision); iii) the simulator must support Standalone deployment, with a focus on the future of 5G networks and without the need for the retro-compatible Non-Standalone version; iv) it should be deployed as a cloud-ready solution, within a containerized environment; v) the simulator should include MANO support, that allows for the deployment of NFs as containerized applications and provides E2C capabilities.

[1] `https://www.o-ran.org`
[2] `https://www.o-ran.org/about`

The chosen package provides a Standalone 5G network adhering to the guidelines of 3GPP Release 15, with O-RAN architecture implemented for the RAN segment.

4.3.1 Aether

For the 5G implementation, we opted to utilise *Aether*[3], an open-source 5G connected edge platform which is built upon the following Open Networking Foundation's projects:

- ROC: a Runtime Operational Control that manages configurations modules for all other components of the infrastructure.
- SD-Core: a 5G Core Network stack.
- SD-RAN: a near real-time Ran Intelligent Controller (RIC) supporting the development of xApps.
- SD-Fabric: an interface that supports a Software Defined Network (SDN) controller.

Figure 4.4 shows the architecture of Aether.

Runtime Operational Control (ROC) It's the brain of the Aether Project that allows users to configure profiles and subscribers as well as implement policies. Its architecture is shown in Figure 4.5. It is composed by three components:

- aether-config: receives configurations and policies from Aether Portals through REST APIs. Aether Portals has the responsibility controlling and observing. The control function involves pushing configurations, while the observation function involves monitoring metrics. It also offers a Graphical User Interface (GUI) to better show data to users. Aether-Config is accountable for administering YANG models that specify UEs, network slices, and UPFs, as well as Core NFs and RAN functions.

[3] `https://opennetworking.org/aether`

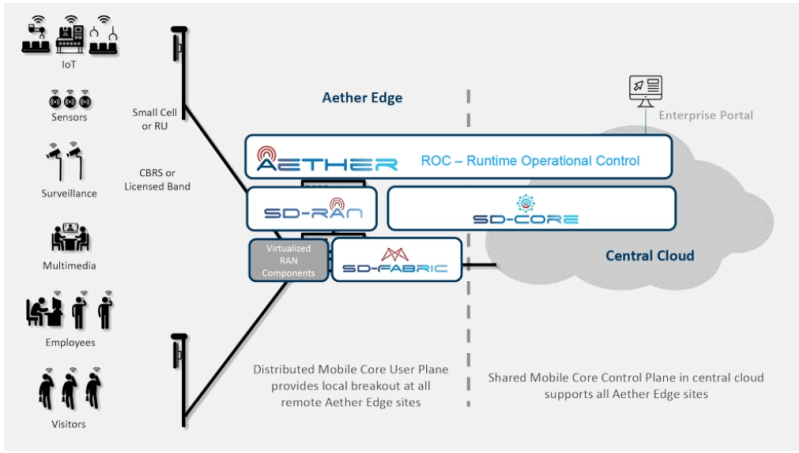


Figure 4.4: Aether architecture.

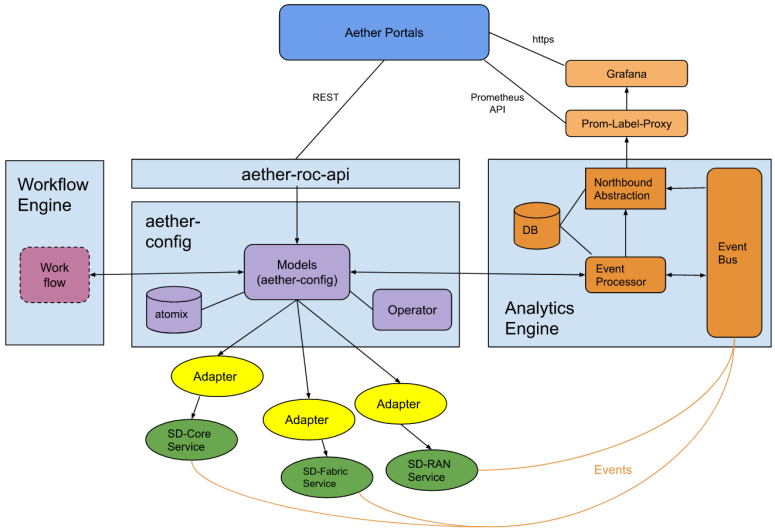


Figure 4.5: ROC architecture.

- Workflow Engine: the workflow engine, to the left of the aether-config stack, is where multi-step workflows may be implemented. The work-

flow engine is a placeholder where workflows may be implemented in Aether as they are required. The workflow engine is responsible for reading and writing the aether-config data model as well as responding to external events.

- Analytics Engine: raw metrics and analytics are processed by the analytics engine, enabling users to monitor all system events using tools such as Prometheus and Grafana.

SD-Core SD-Core[4] is a mobile core implementation for 4G/5G that presents 3GPP compliant interfaces. Additionally, it effectively integrates with the ROC specifications to allow the deployment of the Core Network as-a-service. The Aether control plane consists of components obtained from the open-source *Free5GC project*[5], which implements 5G Core Network components defined in 3GPP Release 15 and beyond, and the ONF *OMEC project*[6], which offers elements compliant with 3GPP Release 13. During installation, the user may choose between the 4G (OMEC) and the 4G (Free5GC) implementations of SD-Core. The figure 4.6 summarises the previous points. SD-Core

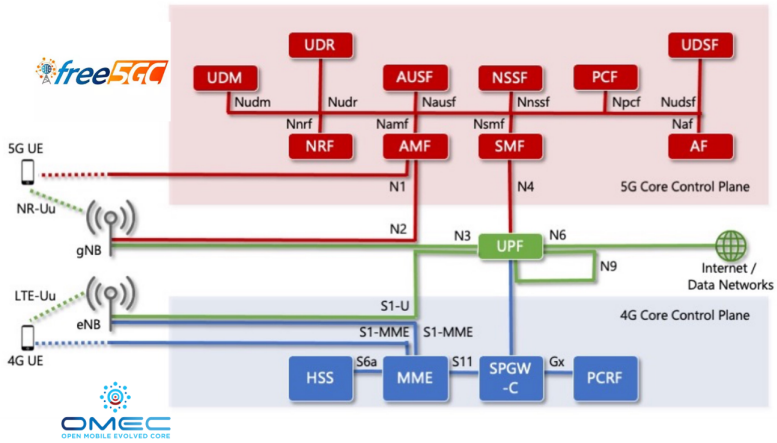


Figure 4.6: SD-Core 4G vs 5G implementation.

[4] https://opennetworking.org/sd-core
[5] https://www.free5gc.org
[6] https://opennetworking.org/omec

provides the capability to manage subscription additions, removals, and modifications, network slice additions, updates, and deletions, and telemetry KPI monitoring through Prometheus, demonstrated in Figure 4.7.

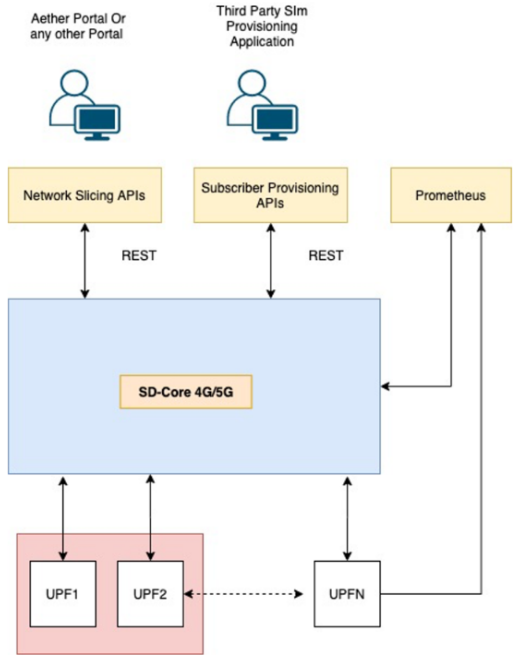


Figure 4.7: SD-Core Block diagram.

Finally, SD-Core implements by itself a gNodeB simulator that simulates UEs and gNodeB (gNB). It provides registration, PDU session establishment, deregistration, service request and ICMP data flow testing. However, Aether project is deployed with a O-RAN compliant RAN (SD-RAN), so there is no need to use this simulator.

SD-RAN and SD-Fabric SD-RAN[7] is the 3GPP compliant software defined RAN implemented by Open Networking Foundation. It is consistent

[7] `https://opennetworking.org/open-ran`

with the O-RAN proposed architecture. It offers a Near Real-time RIC and support for xApps as shown in Figure 4.8. SD-RAN is built on top

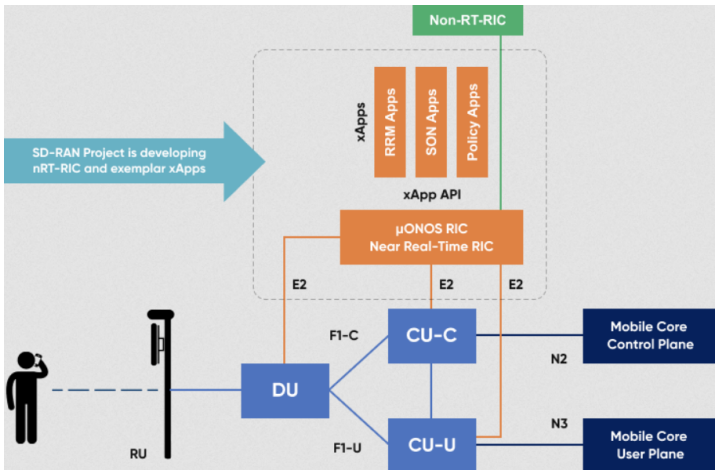


Figure 4.8: SD-RAN components.

of *µONOS*[8], an open-source SDN control and configuration platform. All the components of the µONOS implementation compose the SD-Fabric stack. Figure 4.9 shows the architecture of SD-Fabric[9].

The main component of the SD-RAN implementation inside the µONOS project is the µONOS RIC. Figure 4.10 shows the architecture of the RIC. Its main components are described below:

- Software Development Kits (SDKs): the RIC is composed of Software Development Kits (SDKs) compatible with *Python*[10] and *Go*[11]. Providers can build xApps using the SDKs provided to simplify the development process and to avoid the proliferation of similar code-patterns. xApps are applications that third parties can build to add functionalities to the RAN ecosystem. xApps are managed by the near real-time RIC.

[8] `https://docs.onosproject.org`
[9] `https://docs.sd-fabric.org/master/index.html`
[10] `https://www.python.org`
[11] `https://go.dev`

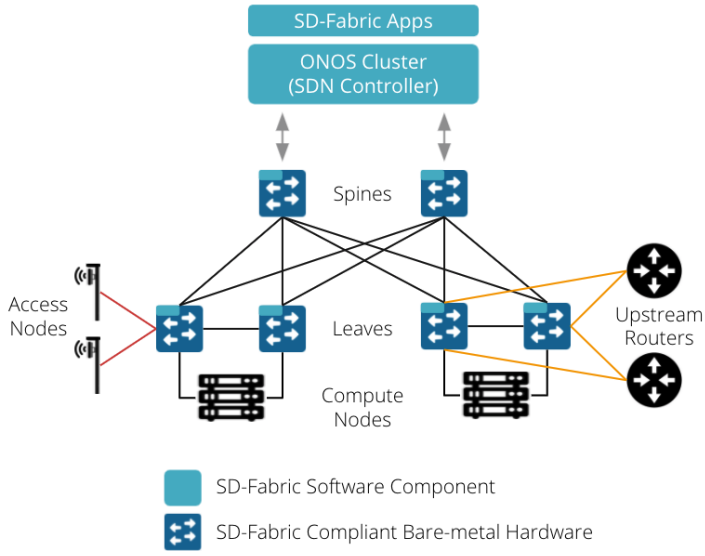


Figure 4.9: SD-Fabric architecture.

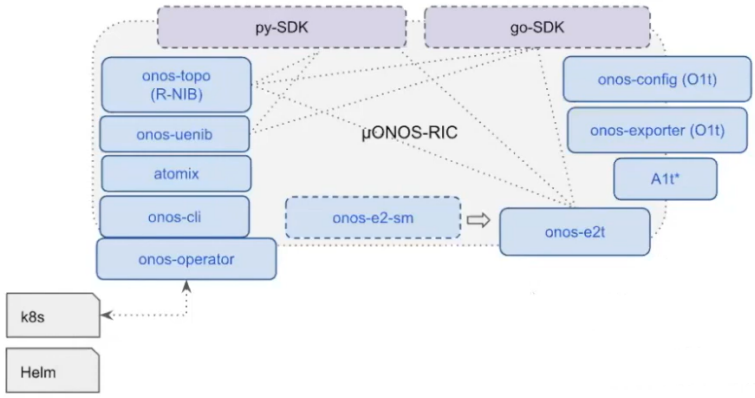


Figure 4.10: µONOS RIC components.

- onos-topo: the main purpose of onos-topo is to provide an abstract representation of the network topology by gathering information about

the network's devices, links, and connectivity. It interacts with network controllers and devices to obtain real-time information about the network's state and dynamically updates the network topology accordingly.

- onos-uenib: tracks and disseminates information about the UEs.

- Atomix Controller: by utilizing the Atomix Controller, ONOS can manage distributed resources, coordinate network operations, and maintain a consistent network view across multiple nodes in the ONOS cluster. This enables ONOS to provide robust and scalable network control and management capabilities for SDN environments. It can for example offer storage facilities for applications to maintain their state in a distributed, scalable and highly-available manner.

- onos-cli: it provides a set of commands that enable users to manage and configure the network, monitor network devices and traffic, retrieve network information, and perform various administrative tasks.

- onos-operator: the component responsible for deploying, operating and managing the components over Kubernetes.

- onos-config: provides a unified approach to configure network devices in a vendor-agnostic manner. It abstracts the details of different device-specific configuration protocols and presents a common interface for configuring and managing network devices within the ONOS controller.

- onos-exporter: the onos-exporter enables the extraction and transmission of network-related metrics, statistics, and events generated by the ONOS controller to external systems for further analysis, visualization, or storage. It provides a means to integrate ONOS with third-party monitoring tools, network management systems, or data analytics platforms.

- A1t: handles external JSON/HTTP REST API requests from northbound orchestration systems and non-real time applications, but it is currently under development.

- onos-e2t: provides the services to facilitate the exchange of control and management information between various network components, such as

the Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU). These components collectively form the base station or radio access point in a 5G network as described in Section 4.2. E2 nodes facilitate coordination and synchronization among different network elements, ensuring efficient resource allocation and management within the radio access network. They also handle the exchange of control signalling messages, enabling the configuration, control, and orchestration of radio access network functions, and, finally, they enforce policies related to radio resource management, traffic prioritization, and QoS enforcement within the network.

Why Aether?

We opted to implement the 5G network via Aether because it is the most complete package available nowadays. It offers 5G Core components compliant with 3GPP Release 15 and RAN components compliant with O-RAN's architecture. We tried and analysed many different solutions before choosing Aether:

- Mosaic5G[12]: it offers a complete 5G stack based on Open Air Interface (OAI[13]) 5G Core Network, OAI RAN, OAI User Equipment and FlexRIC (a RIC built on O-RAN's architecture). A valuable option but it requires to sign a partnership with the group.

- free5GC[14] + UERANSIM[15] (towards5GS-helm[16]): it offers the same 5G Core Network as Aether but the RAN part is not O-RAN compliant.

- Open AI Cellular (OAIC)[17]: it offers a O-RAN compliant RIC and uses srsRAN[18] as the RAN simulator (a 5G non-standalone version of the RAN not deployed on Kubernetes). However, it does not offer the 5G Core Network.

[12] https://gitlab.eurecom.fr/mosaic5g/mosaic5g
[13] https://openairinterface.org
[14] https://www.free5gc.org
[15] https://github.com/aligungr/UERANSIM
[16] https://github.com/Orange-OpenSource/towards5gs-helm
[17] https://openaicellular.github.io/oaic
[18] https://www.srslte.com

Finally, it is worth to mention that Aether is offered by Open Networking Foundation (ONF) as an open-source platform and so it's possible to retrieve the source code for further developments and researches. Table 4.2 shows a comparison between the different 5G solutions.

Table 4.2: Comparison between the different 5G implementations available in literature.

Solution	Open source	Core Network	RAN	3GPP	O-RAN
Mosaic5G	With partnership	Yes	Yes	Yes	Yes
towards5GS-helm	Yes	Yes	Yes	Yes	No
OAIC	Yes	No	Yes	-	Yes
Aether	Yes	Yes	Yes	Yes	Yes

4.3.2 NFV orchestrator and MEC applications

For the implementation of the MEC part of the Testbed we relied on OSM[19] Release 13. OSM is an open-source MANO stack capable of automating network services by minimizing integration efforts. OSM is built on top of two key aspects:

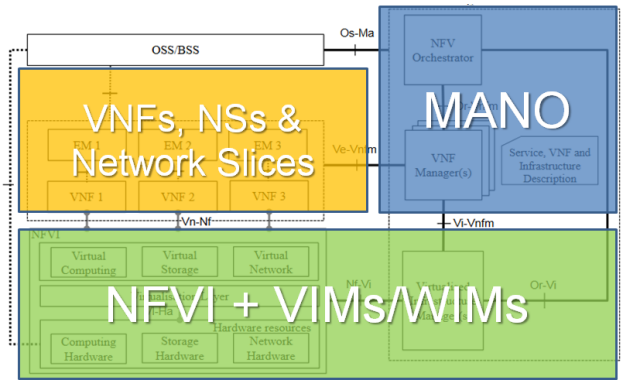


Figure 4.11: OSM alignment with ETSI NFV.

- A well-known Information Model (IM): it is aligned with ETSI NFV

[19] `https://osm.etsi.org`.

standard to support the full automated management of network functions (virtual VNFs, cloud-native Cloud-native Network Functions (CNFs), physical PNFs), network services and also network slices. This lifecycle management comprises instantiation, prior configurations (Day-0, Day-1) and, finally, their daily monitoring (Day-2). The IM aligned with ETSI NFV permits to do all these operations in an "infrastructure-agnostic" manner enabling support for a large variety of VIMs. Figure 4.11 shows the alignment between OSM and ETSI NFV architecture.

- A unified North Bound Interface (NBI): the NBI enables all the operations described in the Information Model over network services and network slices instances. It offers APIs to interact with them.

Figure 4.12 shows IM operations via NBI[20]. OSM can be intended as a

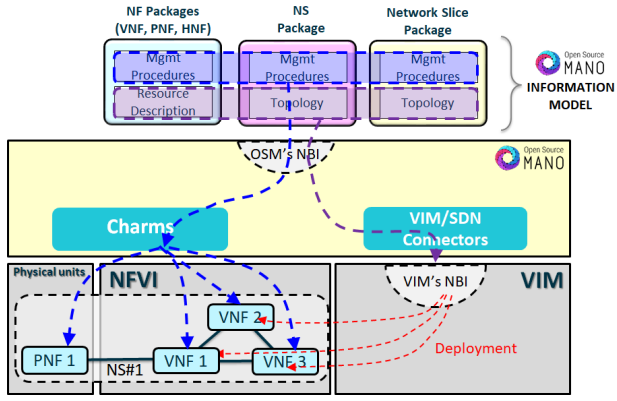


Figure 4.12: IM operation via NBI.

provider of network services on demand, in fact a network service bundles in one single *service object* a set of interconnected network functions (CNFs in our use case) which can run on different infrastructures underneath and in different geographical areas. All of these is obtained through API calls to NBI and descriptors following the model defined by ETSI. The lifecycle of a network service follows precise steps:

[20] `https://osm.etsi.org/wikipub/index.php/OSM_Scope_and_Functionality`

- Modelling: the Information Model provides mechanisms to define the complete expected behaviour of the network service, including its topology, its automation code and its lifecycle operations. Since network services are composed by different network functions that can come from different providers, the IM standardizes them.

- On-boarding: the models are loaded on OSM via NBI APIs. These APIs offer Create, Read, Update, Delete (CRUD) operations over the models. OSM provides support to the on-boarding of network function packages (the set of network functions of the network service to be deployed) and network service packages (that define the network service).

- Network service creation: in this phase, OSM interacts with the VIM and the network functions to create the instance of the network service.

- Network service operation: the instance created during the previous stage is subjected to different types of operations. For example scaling actions, monitoring, upgrades, but also service specific operations defined inside the network service package. It is important to mention the possibility of exporting metrics. This turns out to be, as we will see, very important for the extension of NFV components to integrate intents.

- Network service finalization: it is possible to remove all the resources that had been assigned to a network service, preserving the components that should not be removed like persistent volumes.

OSM consists of several key modules that work together to enable the management and orchestration of network services. Here is an overview of the main modules (Figure 4.13 shows them):

- NBI: the NBI module provides the interface for external entities, such as service orchestrators or operation support systems, to interact with the OSM system. It allows for the creation, modification, and deletion of network services, as well as monitoring and reporting of their status as we said before.

- LCM: the LCM module is responsible for managing the lifecycle of

network services and VNFs. It handles tasks like instantiation, scaling, healing, upgrading, and termination of services. LCM ensures that the network services are deployed and maintained according to the defined policies and service level agreements.

- RO: the RO module is in charge of managing the underlying physical and virtual resources required for the deployment and operation of network services. It interacts with the infrastructure controllers and allocates resources based on the service requirements, such as compute, storage, and network resources.

- VCA: the VCA module focuses on managing the virtualised compute resources, including VMs or containers. It handles tasks like VM lifecycle management, placement, scaling, and monitoring. VCA works closely with the RO module to ensure efficient resource utilization.

- POL: the POL module enables the definition and enforcement of policies related to network service management and orchestration. It allows operators to set rules and constraints for service instantiation, scaling, placement, and other aspects. POL ensures that the system operates within the defined policies and governance guidelines.

- MON: the MON module provides real-time monitoring and performance measurement capabilities for the network services and infrastructure components. It collects data from various sources, such as VNFs, virtual infrastructure managers, and physical resources, to monitor the health, performance, and availability of the network services. It basically serves as the metric exporter used to get the relevant metrics. Prometheus is the solution chosen by the developers of OSM to get the metrics from the infrastructure and from the single VNFD instances. As we will explain later on, OSM's Prometheus instance is fundamental in our research because it permits us to retrieve the available computational resources on the Kubernetes cluster on which the CNFs are going to be deployed.

- Kafka bus: the Kafka bus in OSM facilitates the flow of information between modules. It allows them to communicate asynchronously and decoupled from one another, enabling better scalability, fault tolerance, and flexibility in the system.

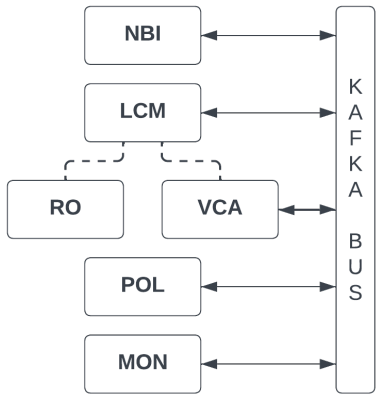


Figure 4.13: OSM modules.

In our deployment we decided to use *Charmed OSM*[21], a package developed and maintained by *Canonical*[22] that provides by default an installation of *MicroK8s*[23], *MicroStack*[24] and a minimal installation of Open Source MANO. MicroK8s is a lightweight version of Kubernetes that is fully conformant and works perfectly on any Linux machine. MicroStack is the Virtual Infrastructure Manager for our NFV implementation. We opted for this deployment flavor because MicroStack offers native support for MicroK8s; OSM can interface with MicroK8s APIs to manage and configure CNFs, and with MicroStack APIs to manage virtual resources used by the CNFs. In this way OSM can automate and simplify the creation, configuration, management and destruction of virtual resources required by the CNFs, improving the efficiency and scalability of the infrastructure.

The "Charmed" version of OSM offers *Juju*[25] as an alternative to Helm to deploy cloud infrastructure and applications and manage their operations from Day 0 to Day 2. Nowadays, Helm is the most used package manager for Kubernetes, but ETSI standards opened the door to this new type of

[21] `https://charmed-osm.com`.
[22] `https://canonical.com`.
[23] `https://microk8s.io`.
[24] `https://microstack.run`.
[25] `https://juju.is`.

deployment flavor.

The installation procedure is quite straight forward and needs only to link the MicroK8s cluster generated by the installation script to the MicroStack VIM. OSM offers a command line interface *osmclient* and a GUI to interact with the NBI for operations from Day-0 to Day-2.

Why OSM?

Choosing OSM over other NFV orchestrators like *ONAP*[26], *Cisco NSO*[27], and *Huawei MANO*[28] can be driven by several factors. OSM closely aligns with the specifications and standards set by the European Telecommunications Standards Institute (ETSI) NFV framework. This compliance ensures interoperability with other NFV components, simplifying integration with existing network infrastructure and reducing vendor lock-in. OSM's adherence to industry standards can provide a more stable and widely supported solution. It is also an open-source NFV orchestrator, offering the benefits of community-driven development and collaborative innovation. The active participation of a diverse community fosters continuous enhancements, bug fixes, and the development of new features (their objective is to provide two releases per year and support to Long Term Service). Finally, OSM has been proved to be as an efficient orchestrator when developing Intent Based Networking. This abstraction promotes agility, reduces manual configuration efforts, and enhances service delivery velocity.

4.4 Automation in networks: Intent Based Networking and services

The manual configuration of network infrastructures implementing SDN, NFV, MEC architectures could lead to human error and in addition, the elimination

[26] `https://www.onap.org`

[27] `https://www.cisco.com/c/en/us/products/cloud-systems-management/network-services-orchestrator/index.html`

[28] `https://carrier.huawei.com/en/solutions/cloud-enabled-digital-operations/mano-solution`

of the expertise as a requirement to handle these types of system could be a valuable simplification of the overall chain of processes to manage them. *Network automation* is one of the key themes that Telco researchers are coping. Intent Based Networking (IBN) is a novel technology for network automation that can dynamically adapt to applications and services. The authors of [18] define an intent as "a set of operational goals (that a network is supposed to meet) and outcomes (that a network is supposed to deliver) defined in a declarative manner without specifying how to achieve or implement them". The idea behind IBN aims to lead towards networks that require only minimal outside intervention using intents to define goals and outcomes in a declarative way, specifying what to accomplish, not how to achieve it: i) Users do not need to know low-level logic. ii) Users do not need to be concerned with how to achieve a given intent. In [19] the authors define important properties that an intent must have in the context of autonomous networks:

- Intent is comprehensible: it has to be understandable by humans, while being formally and unambiguously specified to be processable by machines (e.g., natural language should be an option).

- Intent is declarative: it gives hints for finding the optimal solution without specifying it. Intent declares the wanted results rather than prescribing a specific solution. Ideally, intent describes the properties of a satisfactory outcome rather than requiring a specific outcome.

- Intent is infrastructure agnostic and portable: the expectation expressed by intent originates from contracts and business strategy. It does not change if the underlying system is replaced or modified. While implementation and capability differences between system vendors will continue to exist, intent can be ported between system generations and implementations.

- Intent is complete: intent defines all goals and expected behaviour. If it is not specified as intent, it is not a goal the system needs to consider. This also means that concerns that were common sense in human operated systems would need to become explicitly expressed as intent.

- Intent is composable: multiple intents are given to the Autonomous Networks and it is expected to consider them all together. Unlike traditional software systems, where requirements are analysed offline to

detect and resolve conflicts prior to implementation, intents are added during runtime. Therefore, an essential capability of an autonomous system would be to detect and resolve conflicts.

- Intent is persistent: intent is valid as long as the goals and requirements it expresses are relevant. For example, an intent that specifies that a service needs to be delivered would not become invalid once the service is initially provisioned. Intent is rather the reason for keeping the service operational and assuring its performance. Therefore, intent has a lifecycle that is actively managed by the user or function that has generated it.

- Intent is measurable: it uses measurable and ideally standardized metrics to define the target state. This allows automated evaluation of success as well as identification of issues and optimization opportunities.

An example of intent could be the following: "Minimize network latency even if it means degrading other parameters like packet loss unless packet loss reaches the threshold X". A network architecture implementing IBN should be able to translate that intent into device-specific rules and courses of action to guarantee that the content of the intent is satisfied. It means that the IBN components of the network should be able to continuously check if the intent is satisfied or not and if it is not, the network should apply some types of intervention to correct the issue.

One of the main characteristics of an Intent Based System (IBS) is its ability to learn. In contrast to *Policy Based Systems* that simply apply rules, IBSs create the conditions for continuous learning and optimization. This type of process is called "Cognitive Loop". The intent cognitive loop shown in Figure 4.14 is divided in five phases:

- Measurement: in this phase the system is continuously checked to verify if all the intents declared are being satisfied. This is a critical phase of the cognitive loop because it needs to continuously be in touch with monitoring applications on the hosts. Hosts are required to export metrics in a way that measurement agents are capable to understand (e.g., *Prometheus*, a well-known open-source monitoring and alerting system, supports a variety of exporters that can collect and export

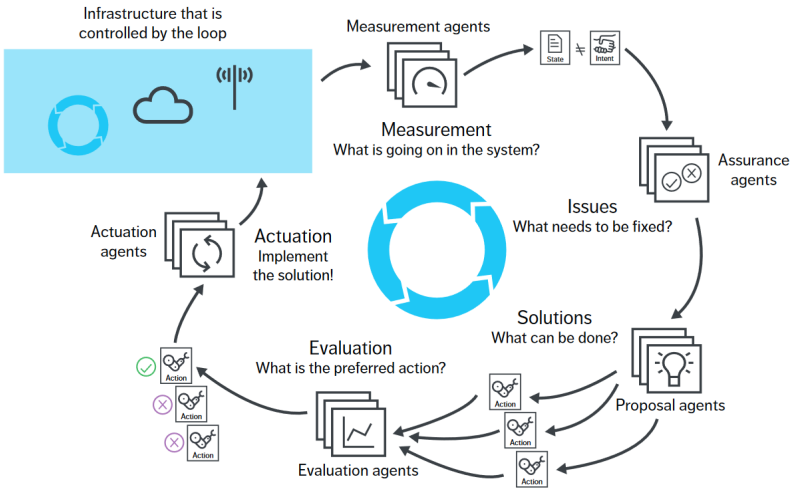


Figure 4.14: IBN cognitive loop [20].

metrics from various sources, including HTTP endpoints, databases, and messaging systems). Finally, this phase uses predictive models to anticipate the onset of problems.

- Issues: if one or more intents are not satisfied what should the system do to change this situation? Special entities called "Assurance agents" verify what needs to be fixed in order to satisfy the intents.

- Solutions: "Proposal agents" propose solutions to fix the issue by receiving as input the issues found by Assurance agents. Typically, the solution proposed by a Proposal agent aims to improve just one metric; this could raise other problems for example when more than one requisite need to be fixed inside the same intent (e.g., a proposed solution can fix the value of a parameter that was preventing the intent to be satisfied but only incrementing other parameters that the intent should also guarantee).

- Evaluation: "Evaluation Agents" choose the best solution among those proposed by trying to predict which one will have the best impact on the system.

- Actuation: the preferred solution is applied in this phase.

4.5 Satellite networks

Communications satellites provide connectivity and services in remote and inaccessible locations.Over the past decade, they have served mainly as network bridges and repeaters, but now researchers are investigating their potential as edge nodes. They can act as service hosts and provide intermediate computation capabilities. While satellite networks share many similarities with more common 5G and LTE networks, they operate in a harsher and more complex environment. Firstly, it is important to note that resource constraints are not solely dependent on node computational abilities, such as processor frequency or available memory, but also on the availability of power generated by the satellite's solar panels and stored in its accumulators. Additionally, the dissipation rate of heat provided by the radiators also plays a crucial role in limiting resources. These constraints vary according to the quantity of electricity collected or heat dissipated.

The prevailing software deployment architectures on the satellite edge are built around minimal distribution of Kubernetes, such as KubeEdge[29] or K3S[30]. These solutions connect Kubernetes agents hosted on edge nodes to a central controller service, which handles resource management and configuration of the cluster. The agents, which are hosting containerised applications, require very limited resources and can be deployed even on small System-on-a-Chip (SoC) platforms, such as Raspberry Pis[31] or similar hardware. KubeEdge additionally incorporates a Message Queuing Telemetry Transport (MQTT) broker to facilitate seamless integration between the edge node and other devices on the same network, for instance in IoT edge deployments. The controller service is responsible for monitoring the activity of connected nodes and managing the life-cycle of service deployments.

The satellite edge field is in its infancy and still has numerous gaps to be addressed:

[29] `https://kubeedge.io`
[30] `https://k3s.io`
[31] `https://www.raspberrypi.com`

Monitoring Satellites operate within an extremely unforgiving environment and endure harsh conditions during deployment and throughout their operational lifespan. Thorough monitoring is essential to detect misbehaviour, perform predictive maintenance [21, 22] (where possible) and collect data to improve the next generation of satellites.

Deployment reliability and availability Reliability is essential at all satellite system levels. However, current deployment solutions are tailored for Cloud-Edge environments and depend heavily on the dependability of the underlying platforms. Moreover, these solutions overlook the specific power and heat dissipation budgets of satellites.

Intercompatability between networks Satellite networks are often closed systems controlled by a single entity or organization that may collaborate with multiple Internet Service Provider (ISP). As a result, satellite networks are strictly separated and cannot communicate directly with each other without using a ground station. More open intercommunication would enable the establishment of a mesh network similar to the Internet on the ground, with all its accompanying benefits.

Heterogeneity and Closure Satellite platforms typically employ custom hardware that is project-specific and may undergo multiple revisions over time. This can introduce internal heterogeneity, software management complexity and the risk of undefined behaviour. Therefore, when deploying applications on such platforms, it is important to consider these factors.

Our research aims to enhance the dependability of services installed on satellite edge nodes by introducing more sophisticated and comprehensive monitoring and assurance capabilities to the deployment platform, while considering its dynamicity and limitations. In order to achieve this, we suggest two ways of extending the Satellite system: i) firstly, enhancing the monitoring capabilities of the edge nodes regarding the hosted services and satellite infrastructure; ii) secondly, implementing an assurance-specific service that collects monitoring events, calculates metrics from the host (satellite) and the services, and provides assurance levels measuring non-functional properties, including reliability, privacy, security, and performance.

4.6 IoT networks

IoT networks are private networks comprising physical devices that communicate and exchange data with one another. The devices connected to these networks are show high heterogeneity and may include different functions, such as sensors, actuators, monitoring, and alerting. Moreover, these devices may use different technologies, including 5G, WiFi, Bluetooth, ZigBee, and Lora. In addition to this, the network size may range from relatively small, such as a single household, to extensive enough to cover a large city. Networking protocols and technologies are diverse and often proprietary and heavily regulated by the device manufacturer, often resulting in complete reliance on the manufacturer for compatibility, support and security patches. A compromised device could serve as a pivot point of attack within the network, requiring us to minimise trust in other devices and in the network itself. The IoT field is nascent and has a multiple gaps remain to be addressed:

Authentication and authorisation numerous IoT devices implement inadequate authentication techniques and frequently employ simplistic or default passwords that users do not modify, resulting in increased vulnerability to security breaches. It is also common for these devices to lack a strong mechanism for verifying the identity and permissions of users and other devices that communicate with them.

Encryption data in transit and at rest on the device may lack encryption, resulting in user data being exposed to attacks and being compromised. It is unfortunate that data contained in IoT devices is often deemed insignificant for privacy. However, even sensor or usage data can reveal information about a user's habits.

Vulnerabilities in Firmware IoT devices frequently lack sufficient quality assurance in their software implementation, as manufacturers are not required to disclose their source code for verification and certification, leaving security and privacy gaps unpatched and undisclosed. Recent security scan-

dals involving IoT devices[32][33] and their insecure implementation are just the last episodes of a more deeply rooted issue of lack of (or limited) liability for companies not securing their products.

Patches and updates privileged access to certain internal components of the device, namely the system memory and the trust chain, is essential for updating the device's software. Regrettably, these components are often inadequately secured, leaving them vulnerable to compromised firmware updates that may contain rootkits and backdoors[34].

Insecure Communication Protocols and Channels IoT devices may rely on communication channels that are inherently secure. Potential security flaws include include unencrypted HyperText Transfer Protocol (HTTP) traffic, insecure SNMP versions or WiFi networks with inadequate security levels. Currently, IoT devices, particularly those aimed at consumers, prioritise compatibility rove security and privacy. The device must always adopt to a secure behaviour, either by using secure channels if viable or by declining to share data insecurely (unless forced to do so by the user after being clearly informed about the associated risks).

Locality and sovereignty IoT devices, in particular those aimed to the consumer market, tend to offer Cloud-based services, such as remote control and monitoring capabilities. This feature is much appreciated by users but it makes their data at risk of being collected and exploited by the Cloud Service Provider (CSP). If the data does not follow a correct end to end encryption approach, private information such as user behaviour and camera images may be leaked to the provider. Moreover, even if the channel is securely encrypted, much information about the user can still be inferred by the device behaviour (network traffic, frequency of updates, flow of execution) [23, 24].

[32] https://www.theverge.com/2022/11/30/23486753/anker-eufy-security-camera-cloud-private-encryption-authentication-storage
[33] https://www.bbc.com/news/technology-23971118
[34] https://defcon.org/html/defcon-31/dc-31-speakers.html#Giese

Chapter 5

Assurance of Telco Edge Networks

The 5G core network is a composition of services that tightly collaborate to provide network functionalities, i.e. traffic routing, user authentication, radio management. Research on 5G services mostly focused on functional aspects of 5G network and services, often neglecting assurance solutions verifying and maintaining its NFPs. Currently, the 5G-edge ecosystem is mostly focusing on performance properties, that is, latency, data rate, and connection density of the network, as they are the most interesting for ordinary consumers [25]. While their importance is significant and provides a good point of view on the services' status, there are aspects that are left out of the picture. Users may want to know in advance the profile of services running in a certain host, providing stronger guarantees in the form of SLAs. For instance, properties such as *parallel data ingestion* are particularly important for massive IoT networks and for service deployment scheduling, as they can help to identify the best deployment solution during the decision process [26].

In this context, the collection and measurement of Key Performance indicators (KPIs) such as memory, network bandwidth and computational power consumption are fundamental to develop assurance solutions aiming to verify NFPs of 5G network and services. The formalization of such properties is especially useful in commercial scenarios, where the service providers need to quantify, and possibly certify, the capabilities of their services to clients. The

services in the 5G core network periodically measure their KPIs and expose the information to other NFs using standardized APIs. The current API is however limited to the performance metrics and does not provide easy means to extend its capabilities, restricting its functionalities and flexibility. Let us consider the scenario where a user that needs total isolation for their services, requiring the node to not be shared with others.

5.1 Research objective

Our research objective is to extend the monitoring mechanism of the 5G core network services by expanding the set of NF metrics [27]. To this aim, we propose a specialized assurance component that aggregates the collected measurements, validates the system state against abstract representations, and exposes the relevant information through an extensible interface. Such component extends the 5G network with data logging, model checking, anomaly detection, and metrics prediction capabilities. It is important to note that since part of the collected data may be complex or sensitive, the new component has to expose a query interface abstracting the underlying layers and providing access control capabilities. This approach can also simplify integration with other services, such as deployment schedulers and NFs deployed on the node.

The core network extension in this work introduces several new assurance functionalities providing benefits for both the network providers and its users. The additional metrics permit to define more specific and diverse SLOs and to better specify the deployments requirements. The newly introduced component permits to identify possible issues with deployment configurations and to verify compliance against specific SLAs. Finally, the API supports simplified integration with other services, which can provide enhanced functionalities based on additional non-functional information.

5.2 Motivating Scenario: 5G enabled edge computing

In this section, we present our motivating scenario to show the feasibility and utility of our infrastructure assurance methodology applied to a multi-level service deployment with edge-cloud continuum capabilities using 5G enabled networks. The 5G Core network infrastructure, thanks to MEC will be increasingly involved in business process, such as vehicular network, critical health applications and smart cities, to name but a few. In the near future MEC will play a fundamental role in Cloud Edge Continuum, providing standardized high reliability infrastructure for service delivery and execution. In such scenario the deployment is distributed across edge and cloud computation nodes, exploiting the capabilities provided by the 5G core network, such as low latency and locality of the computation. Common applications of such techniques are to provide low latency services in mobile contexts, hardware acceleration and resources to low power devices for computation offloading and to preprocess streams of data before forwarding it to the cloud e.g., filtering and compressing information coming from large groups of IoT devices.

Business ready solutions for this kind of deployment are already available to the public in some regions (e.g., Amazon Web Services (AWS) wavelength, Google Distributed Cloud and Microsoft Azure private MEC) and are expected to become more common in the next years as the number of ISPs providing similar solutions grows.

The properties of the service composition are the ones associated to its cloud and edge components. This includes properties previously studied in literature, such as [28–30]

While performance SLAs and monitoring techniques are already present and common in both cloud and edge environments, these platforms are lacking in assurance and certification solutions to allow verification of low level NFPs. Although cloud-ready monitoring solutions, such as the widely established Prometheus[1] and the newer and more extensive OpenTelemetry[2], are frequently employed in production environments, their deployment displays a

[1] `https://prometheus.io`
[2] `https://opentelemetry.io`

significant lack of consistency. More specifically, the services monitoring implementations do not follow a common service-level metrics standard, so that contracts have to be defined on a per-component implementation basis. This substantially limits the transparency while comparing alternative solutions and makes contracts definition more cumbersome. Verification and certification of NFPs are of interest to the services' users, especially in safety or privacy critical applications e.g., health and automotive. This work focuses on defining a more fine-grained framework for verifying NFPs associated with functionalities in such a way that it is transparent to the environment and their specific implementation.

5.3 Actors

Our model identifies three types of actors interacting with each other.

Telco providers offer 5G solutions to deploy services in the edge nodes. They interface with the other actors as infrastructure and/or hosting service providers, provisioning both the networking and execution resources necessary for to run the services.

Cloud service providers offer users the infrastructure and management capabilities necessary to deploy and monitor services. They are interested in expanding their facilities by integrating their systems with the ones owned by telco providers, complementing their services with user-local hosting capabilities.

Cloud service users deploy their services on CSPs infrastructures and are interested in achieving the best performances at the smallest price. They are also interested in maintaining control on the NFPs of both their services and, by extension, the ones linked to the hosting platform.

5.3.1 5G core network services

Table 5.1 lists the services composing the 5G core network and resumes their purposes. Figure 5.1 shows how the core network is composed and how the services are connected.

Table 5.1: Core network services and their purposes.

Service	**Description**
Host	Provides the resources required for the CNs to operate and manages their processes life-cycle
NSSF	Network slice selection service focusing on resource optimization
NEF	Access control to CN's internal data and management of collaboration with other networks
NRF	NF's service discovery
PCF	User policies management and charging handler function
MEC	CNF and VNF deployment and management platform
AUSF	Authentication server
AMF	Devices authentication, access control, encryption and session authorization
SMF	Session management service, UE IP address allocation and management, DHCP, NAS signalling, traffic steering
UDM	Service that manages user related data from other processes
UE	A device that connects to the core network through radio communication
RAN	Radio network infrastructure (antennas and controllers)
UPF	Packet routing and forwarding, packet inspection, QoS handling,
LADN	Private network or bridge to the public Internet

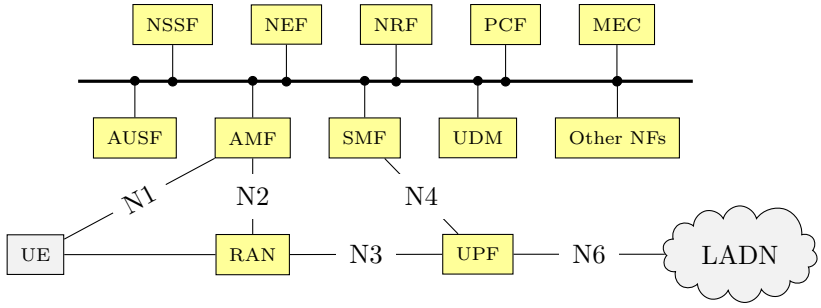


Figure 5.1: Scheme of the 5G core network architecture.

5.4 5G Functionalities

Table 5.2 describes peculiar 5G functionalities that can be of interest for an application workflow, a subset of which is modelled in this work. We also mentioned the 5G core network services collaborating to achieve the functionality. In Table 5.3 we list 5G functionalities that are not available yet, but can be implemented with limited effort to complement the actual functionalities.

Table 5.2: 5G core network implemented functionalities.

Id	**Name**	**Description**	**Core network components**
f_1	DN-level connectivity	The CN can connect to the DN	RAN, UPF, SMF
f_2	RAN-level connectivity	The CN can connect to the UEs through its RAN	RAN, UPF, SMF
f_3	UE-DN routing	Network packet routing from UEs to DN	RAN, UPF, SMF
f_4	UE-NF routing	Network packet routing from UEs to NFs	RAN, UPF, SMF
f_5	NF-DN routing	Network packet routing from NFs to DN	RAN, UPF, SMF

f_6	Massive IoT Networks	Connection with large number of IoT devices	Host, RAN
f_7	Low Latency Networks	Low latency networking functionalities on mobile devices, providing fast access to the edge nodes and to the cloud	Host, RAN
f_8	Network hop-based deployment selection	The deployment host can be selected on the number of network hops required to reach a given application	Host, MEC, NRF
f_9	RTT-based deployment selection	The deployment host can be selected on the latency reaching a given application	Host
f_{10}	Secure network channels	The CN provides secure channels for the deployed applications	Host, MEC, RAN
f_{11}	Private network slice management	5G offers network slices as private networks with SLO negotiation	RAN, UPF, SMF, NSSF
f_{12}	Bandwidth allocation	Network bandwidth allocation	Host, MEC
f_{13}	CPU allocation	CPU timeshares allocation	Host, MEC
f_{14}	RAM allocation	Memory allocation	Host, MEC
f_{15}	Hardware acceleration	Hardware accelerator allocation	Host, MEC
f_{16}	CNF-based application deployment	The CN can be the target of a CNF-based deployment	Host, NEF, SMF
f_{17}	VNF-based application deployment	The CN can be the target of a VNF-based deployment	Host, NEF, SMF, MEC

f_{18}	Policy management	Core network policy management (users, applications, network routing)	Host, NEF, MEC
f_{19}	Tenant management	The CN provides tenant management policies (i.e., limit one user per node)	Host, MEC
f_{20}	Certifications exposure	Platform certificates are exposed and available to the user	Host, MEC, RAN
f_{21}	Protocol compliance	The NFs are compliant with a given protocol (storage, network, deployment)	Host, MEC, RAN, NRF
f_{22}	UE-based Identification	UEs and their users can be uniquely identified in the network by the device IMEI and (e)SIM identifiers	Host, RAN, UDM
f_{23}	Ephemeral storage	The CN provides an ephemeral storage service	Host
f_{24}	Permanent storage service	The CN provides an permanent storage service	Host, NEF
f_{25}	Storage level integrity	The CN storage provides integrity guarantees (hardware redundancy, data replication)	Host, MEC, RAN
f_{26}	Storage level encryption	The CN storage is encrypted	Host, MEC, RAN

Table 5.3: 5G core network functionalities likely available in the near future.

Id	Name	Description	Core network components
f_{27}	Private Cross-Network Slices	CNs' functionality of collaborating with each other to form inter-network and inter-ISP private network slices. This allows the creation of secure channels across a variety of networks	RAN, NEF
f_{28}	Contextual Deployment Policies	The controller that manages application deployments in the 5G network is aware of the context and can be extended to support more advanced requirements than resource constraints, including privacy and availability NFPs	Host, NEF
f_{29}	Geo-based deployment selection	The deployment host can be selected on its geographical position	Host, NEF
f_{30}	Geo-fenced applications	f_{28} can be extended with f_{29} to support user-defined geographical barriers, so that data and applications cannot accidentally be shared outside the selected region	Host, NEF, MEC
f_{31}	UE-based authentication	UE-based authentication service extending f_{22}	RAN, SMF, AMF, NEF
f_{32}	UE-Edge Resource Sharing	f_7 and f_{11} allow to link UE with resources on the edge network using private and low-latency connections	Host, NEF, RAN

f_{33}	Distributed permanent storage	The edge network can extend f_{24} acting as a fast-access intermediate layer in a distributed storage solution	Host, NEF
f_{34}	UE-unique encryption	Extends f_{23} and f_{24} with a UE-unique transparent encryption layer	Host, MEC, RAN, PCF
f_{35}	Storage Certified Ephemerality	f_{28} can help track the accessed resources providing f_{23} with certified ephemerality	Host, MEC, RAN, PCF
f_{36}	Secure deployment channels	The f_{11} and f_{27} capabilities of the 5G network can be used to provide both the users and the applications with secure channels applying a transparent encryption layer on any channel within and outgoing the network	Host, MEC, RAN, NEF
f_{37}	Deployment isolation	We can extend f_{28} to support isolated deployment strategies, where one user can allocate the full edge node	Host, MEC
f_{38}	Multi-ISP deployment (service roaming)	We extend f_{11} and f_{28} to allow continuous application deployment and migration across ISPs, moving the applications to another hoster once the SLOs cannot be respected	Host, MEC, NEF

5.5 5G Properties

Table 5.4 contains the definitions of the fine-grained NFPs of interest given the 5G functionalities in Section 5.4.

Table 5.4: 5G network properties.

Id	Name	Types	Functionalities	Description
p_1	Network connection availability	t_{avail}	f_1, f_2, f_3, f_4, f_5	The infrastructure provides network connection capabilities to and from its applications and devices
p_2	Network deployment availability	t_{avail}	f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}	The infrastructure meets all the requirements for the deployment of a given application
p_3	Network capacity availability	t_{avail}	f_6	The infrastructure can accept new connections
p_4	Network latency performance	t_{perf}	f_7, f_9	Network infrastructure with minimal latency to a certain application
p_5	Network hops performance	t_{perf}	f_7, f_8	Network infrastructure with minimal number of hops to a certain application
p_6	Network confidentiality	t_{conf}	f_{10}	Can provide secure network channels
p_7	Cross-network availability	t_{avail}	f_{27}	The CN can coordinate with other networks
p_8	Network management automation	t_{auto}	f_{18}, f_{19}	The network manages its configuration automatically
p_9	Storage availability	t_{avail}	f_{23}, f_{24}, f_{33}	Can guarantee storage availability
p_{10}	Storage confidentiality	t_{conf}	f_{26}, f_{31}, f_{34}	Can guarantee storage confidentiality

p_{11}	Data deletion traceability	t_{conf}	f_{23}, f_{35}	Ephemeral data is guaranteed to be deleted as soon as not required by the user (data deletion traceability)
p_{12}	Storage UE-based access control	t_{conf}	f_{34}	The infrastructure provides UE-based encryption capabilities to the storage service
p_{13}	Storage integrity	t_{inte}	f_{25}	Can guarantee storage integrity
p_{14}	Geographical vicinity	t_{conf}, t_{perf}	f_{29}	The target UE is connected to the geographically closest infrastructure's node
p_{15}	Deployment availability	t_{avail}	f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}	The infrastructure can be used to deploy applications
p_{16}	Exposed host certifications	t_{auto}, t_{avail}, t_{conf}, t_{inte}, t_{perf}	f_{20}, f_{21}	Valid external certifications of the infrastructure's software, hardware and configurations are exposed to the user
p_{17}	Resource sharing support	t_{avail}	f_{32}	The infrastructure provides resource sharing capabilities to the connected UEs
p_{18}	Deployment roaming support	t_{auto}, t_{avail}	f_{38}	The infrastructure allows for multi-ISP deployments, coordinating with other CNs

p_{19}	Execution isolation	t_{conf}, t_{perf}	f_{19}, f_{37}	The infrastructure can provide an isolated application execution environment
p_{20}	Network isolation	t_{conf}	f_{11}, f_{36}	The infrastructure can provide an isolated network channel
p_{21}	Authentication provider	t_{conf}	f_{31}	Provides 5G-based user authentication services
p_{22}	Geo-fenced deployments support	t_{conf}	f_{29}, f_{30}	The infrastructure supports geo-fenced deployments, limiting access to applications and data from within a certain geographical region
p_{23}	UE-based authentication support	t_{auto}, t_{conf}	f_{31}	The infrastructure supports UE-based authentication for its applications deployments
p_{24}	CN policy based deployment automation	t_{auto}	f_{18}	The deployment of applications in the infrastructure is automated following the CN policies
p_{25}	Context-based deployment policies	t_{auto}	f_{28}	The infrastructure supports context-based deployment policies, enabling advanced NFPs based SLO definitions

5.6 Assurance in satellite and IoT networks

Satellite edge networks present peculiarities that must be considered when utilized for edge computing. One such peculiarity is latency, which grows when data is transmitted over great distances, as in the case of satellite networks. This circumstance significantly delays communication between satellites and ground stations, and impacts the performance of any application that employs this network. This circumstance significantly delays communication between satellites and ground stations, and impacts the performance of any application that employs this network. Therefore, it is vital to account for this latency in any satellite edge network implementation. The availability of satellites varies depending on their target altitude (Low Earth Orbit (LEO), Medium Earth Orbit (MEO) and Geostationary Earth Orbit (GEO)), causing changes in their relative position with respect to a point on the Earth's surface over time, leading to sporadic direct-connection availability. Furthermore, the computing capacity of satellites is limited by the energy stored on board, typically produced by solar panels and contained in batteries. Heat is also a significant limitation due to the reduced dissipation in space, which is influenced by how much of the satellite surface is not exposed to the sun.

In this context, it is crucial to take into account the particularities and variations over time, while ensuring non-functional properties. We can accomplish this by enhancing the monitoring capabilities of deployment targets by extending the host and service metrics APIs and implementing more comprehensive metrics on their status. This would result in a more precise and specialised examination of process usage, including estimation of power consumption and execution patterns, such as expected load variations. The expanded metrics provide more comprehensive data that can then be used to build predictive models (for example, to estimate a more sophisticated resource allocation for a particular service deployment). A further process involves setting up an assurance service to collect data from the metrics APIs. These evaluations enable the acquisition of assurance evaluations and potential forecasts regarding the status of the host and services. A set of predesignated checks can be made available as templates for usual circumstances.

The expected results of this research effort are twofold: i) integration of a more precise and detailed monitoring solution for services deployed at the edge can enhance the quality of service in the satellite continuum's deployment strategy; ii) the project aims to develop an assurance-focused service for

centralised monitoring and verifiable assurance verification on the satellite.

Similarly to satellite nodes, IoT devices have limited computing and availability capabilities. They are typically battery-powered units that periodically awaken from a deep sleep state to perform calculations, which makes them intermittently available. Depending on the network type and protocols employed, security issues can arise in IoT networks when unsecured channels are utilised. In addition, the common occurrence of lack of security patches from their vendors can lead to a compromised node in the network. To improve the quality and security of IoT networks, it is recommended that assurance and certification methodologies be implemented. This will enable users and CSPs to transparently verify and certify device and service behaviour. Through this approach, automatic checks can detect security inconsistencies and abnormal behaviour. The verification process for the IoT platform should be comprehensive and conducted by a trusted third-party Accredited Lab (AL). This establishes that the devices and services offered by the CSP are free from any erroneous or malicious behaviour, thus mitigating risks to consumers. The manufacturer of the device and the CSP may provide the user with a certificate of compliance, which is periodically re-verified by an AL.

Chapter 6

Assurance for CDN networks

A CDN is a network service that caches contents on servers that are closer to the end users, reducing the latency and bandwidth consumption of delivering the content from the origin server. A CDN also provides other benefits, such as improving availability and scalability of the content.

A CDN service usually includes the following elements:

- a collection of servers disseminated throughout various geographical zones, referred to as Points of presence (PoPs), which hold duplicates of the content.
- the service utilizes a mechanism to direct user requests ti the most suitable PoP based on factors such as server performance, network conditions and the distance.
- a mechanism to synchronize and update the content across the PoPs, ensuring eventual consistency and freshness.
- a mechanism is needed to optimize content delivery, which may include compressing, encrypting, or transforming the content to suit the user's device or preferences (e.g. transcoding of videos to different quality

levels and formats).

- a mechanism to monitor and analyse the traffic, performance and usage of the CDN service is essential for providing insights and feedback for improvement.

CDNs are typically integrated into the application-level networking layer, such as via HTTP proxies. However, they can also operate in lower lower layers, embedded within the transport layer, offering transparent caching of shared network data.

In this chapter, our assurance methodology is applied to verify properties of content delivery networks (CDNs) with the aim of certifying their operational status. This provides guarantees to both the service provider and users.

6.1 Information Centric Networks

ICN is a network paradigm that addresses contents in the network using unique URI-like names. It is becoming more popular as a replacement for the conventional Transmission Control Protocol (TCP)/IP network stack, particularly when in-protocol content distribution and privacy features are crucial [31–36]. The transition from the traditional host-centric paradigm of the TCP/IP stack eradicates the necessity for uniquely identifying the network nodes involved in the communication through network addresses, while combining the network, transport, and application layers into a unified hybrid layer. Furthermore, ICN networks are also agnostic over the transmission medium, enabling deployment on various physical network layers, including Ethernet, WiFi, Bluetooth, or other network protocols. The primary benefit of ICN-based networks is their in-protocol distributed caching system. In contrast, media sharing solutions based on TCP/IP are not scalable and depend on CDNs to meet the needs of numerous clients. ICN networks permit each node to cache content. As a result, multiple client requests can be immediately satisfied by closer nodes and the network's overall load is reduced.

In recent years, significant progress has been made in the research and development of high-quality, high-performance, and functionally resilient ICN

networks [36, 37]. Furthermore, there has been extensive research into the security of ICN, particularly in regards to specific attacks [38, 39] and countermeasures [40–43]. Monitoring solutions have been proposed to deepen the observation of network the network status. These solutions consist of software and hardware tools [44–48] that measures the conditions of nodes and their communication links (e.g., load, traffic utilisation, exposed services and uptime). Multiple protocols have been implemented for network monitoring, such as SNMP and Internet Control Message Protocol (ICMP), to facilitate the detection and configuration of network nodes and aggregation of monitoring measurements.

Relying solely on monitoring may not always be adequate in evaluating the security level of a network. Hence, the adoption of security assurance has become prevalent in enhancing the safety of a target system with the guarantee that it functions as intended despite possible failures and attacks. In this context, certification emerges as a favoured assurance technique, collecting evidence to demonstrate a precise property of a system. The gathered evidence undergoes assessment by an AL that awards a certificate to the system proving a specific (set of) property. Certification schemes have been applied beyond traditional software (Common Criteria [49]) and targeted web and cloud services [50–53] and, more recently, complex service compositions, where the collected evidence is based on monitoring, testing, or formal proofs. The peculiarities of recent certification schemes [50, 51], being dynamic, continuous, lightweight, make them an opportunity even for verifying properties of complex network protocols. However, to the best of our knowledge, security assurance and certification of ICN are still in their infancy. Transparency and trustworthiness of information-centric networks become then a major hurdle against their widespread adoption and can open the door to persistent threats that affect the network behaviour to its foundation. In addition, weaknesses to poisoning attacks and system malfunctioning can impair the entire network operation [44, 47, 54].

This work applies the methodology described in Chapter 3 extending our network-level certification approach in [55] and its contribution is threefold. It first provides an enhanced assurance model capturing the evolution of the system over time (Section 6.3.1), and new services providing certification functionalities (Section 6.4), which are fully integrated with the original protocols reducing the performance impact on the overall network. It then defines two deployment models (Section 6.5 and Section 6.6), centralized and decentralized, which fully integrate with ICNs improving their trustworthi-

ness. It finally provides a discussion on application scenarios of interest for ISPs or cloud providers offering certified services (Section 6.7).

6.2 NDN-based CDN

NDN is the leading implementation of ICN and supports multiple platforms (e.g. Linux, Windows, MacOS, Android, Arduino, etc.) and physical layers (e.g. Ethernet, WiFi, Bluetooth, etc.), soon becoming the major target of research in ICN. In NDN, any server-produced content must to be signed with a valid signing certificate. This enables any receiver to easily and transparently verify authenticity of the data. This feature facilitates in-protocol caching, as data can be cached in untrusted network nodes without losing its validity. The NDN framework also implements the concept of content freshness, permitting the content producer to specify the duration for which an item remains useful.

The of these features allows NDN to offer users with an in-protocol caching solution where each router in the network can function as a cache node and distribute the accumulated data to other clients. This method proves particularly advantageous in contrast to extensive CDNs as it negates the necessity for tailor-made setup and remains inconspicuous to users. Furthermore, although CDNs are typically situated in large regional data centres, NDN caches can be situated closer to users, thereby decreasing network usage and enhancing latency.

6.3 Assurance Methodology and System Model

Although NDN's distribution system is inherently secure from impersonation and content manipulation, possible attackers can still affect its caching mechanism through cache poisoning and cache pollution attacks [6,40,41,55], leading to Denial Of Service (DOS) and reduced availability and performance. In this context, continuous assurance and certification of ICN networks aid in

verifying the correct operation of the system, identifying potential misbehaviour and the guaranteeing advanced NFPs to the network's users.

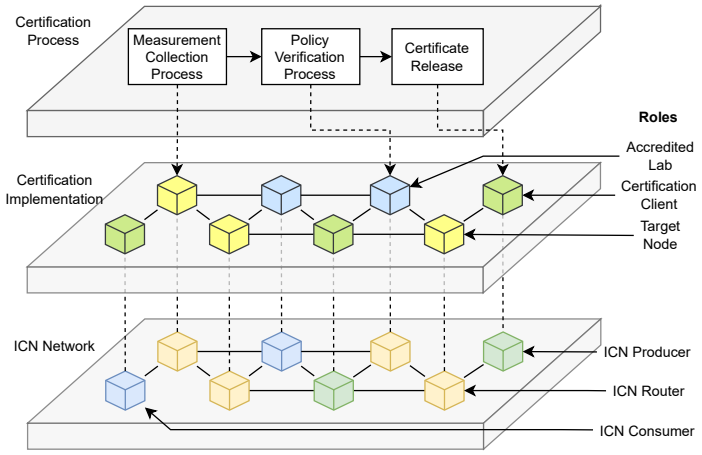


Figure 6.1: A layer-based view of our System model.

Our system model is a standard ICN network extended with the certification methodology presented in this work. In ICN, content consumers (ICN Consumer) request contents by sending *interest packets* to neighbour nodes only including the content name and optional request configuration parameters. Content producers (ICN producer) register a series of prefixes for which they can respond with *data packets*, containing the content itself and a signature that guarantees the integrity and non-repudiability of the data. *Data packets* shared through the network can be cached by any nodes in the packet path: each router (ICN Router) that receives an *interest packet* first checks for a matching *data packet* in its Content Store (CS) and alternatively forwards the request to its neighbours. Figure 6.1 graphically shows our complete system model as a traditional ICN network extended with the certification methodology in Figure 6.2. Any nodes in the ICN network can play any role in our certification methodology as follows: i) the *Certification Client* asking for a certification process execution, ii) the *Accredited Lab (AL)*, responsible for the entire certification process, and iii) the *Target Node*, as the target of the certification process. The *Certification Process* orchestrates two sub-processes to release and distribute certificates: the *Measurements Collection Process* and the *Contract Verification Process*. The certification client queries the AL to execute a certification process on a set of target nodes to prove a given NFP.

The target nodes are empowered with our Measurements Collection Process that produces metrics to be evaluated by the Contract Verification Process. The Contract Verification Process generates evidence based on metrics to support the award of the certificates that are attached to the target nodes and retrieved by the certification clients.

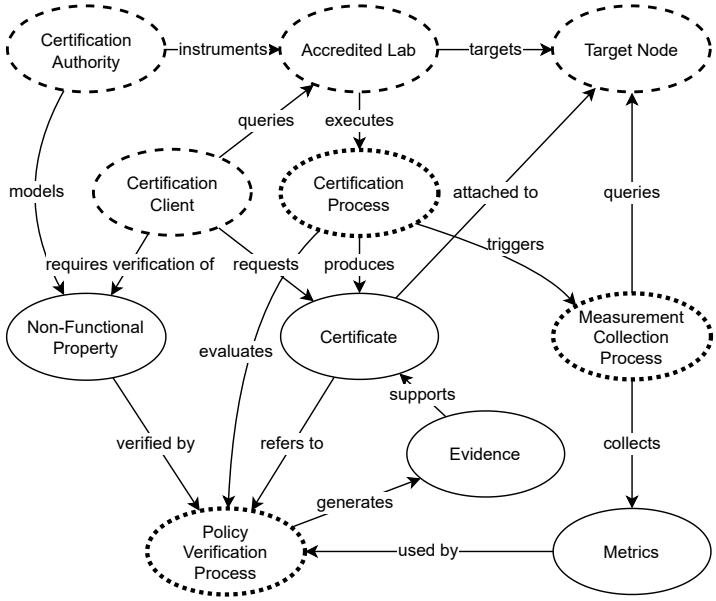


Figure 6.2: Certification methodology. Dashed lines refer to certification roles, dotted lines refer to certification process, and black lines refer to components of the certification methodology.

More in detail, the certification roles have the following responsibilities.

- Accredited Lab (AL) is a (set of) network node orchestrating the entire certification process and implementing the Contract Verification Service. It is responsible for the verification of contracts, and the production of certificates assessing the collected evidence. ALs listens for certification requests from certification clients. For each valid certification request the AL starts a certification process as described in Sections 6.5 and 6.6, and once terminated returns a list of the produced certificates names, each of which is associated with one of the target

nodes. Depending on the network and configuration scale, ALs has a different view on the status of the whole network; an in depth analysis of the possible solutions is presented in Sections 6.5.1 and 6.6.1.

- Certification Client is a network client having interest in the certification process. Any devices in the network, including those that are not acting as routers, can request a certification to one or multiple ALs to verify properties over a set of target nodes. The obtained information helps the client in its internal processes by identifying nodes suitable for the deployment of a service, suggesting a more efficient or secure routing path, improving privacy by excluding untrusted nodes, to name but a few.

- Target node is a network node whose status can be measured by ALs and can be targeted by certification requests from certification clients. It runs the Measurements Collection Service that measures its internal state and exposes it to trusted ALs[1].

The three roles are independent from each other and a ICN node can play multiple roles; for instance, a certification client can request a certification to an AL for its own status or be an AL itself. A node can act as an AL for its own status (self-certification), although the produced certificates cannot be considered reliable from other clients. This decoupling property is particularly interesting in the case of a totally decentralized certification model as we discuss in Section 6.6.1.

Our certification methodology is built on three main building blocks, an abstract certification model (Section 6.3.1), the certification services (Section 6.4), the certification process (Sections 6.5 and 6.6), which completes our system model. In the following, we consider NDN, the most common and studied ICN protocol, as the reference protocol.

6.3.1 Abstract Certification Model

Similarly to what happens for complex service-based systems, an abstract certification model is defined to cope with the network complexity, where each network implementation differs from the others and needs a specific

[1]Encryption can be adopted to preserve confidentiality over the shared measurements.

certification process. The certification model discussed in this chapter is an implementation of the more generic model presented in Chapter 3. The main differences are i) monitoring endpoints are implemented using the NDN protocol, allowing data to be cached in the network; ii) we adopt a more practical approach to contracts using a simple specification language described here in Backus-Naur Form (BNF)-notation. Figure 6.3 shows the certification model

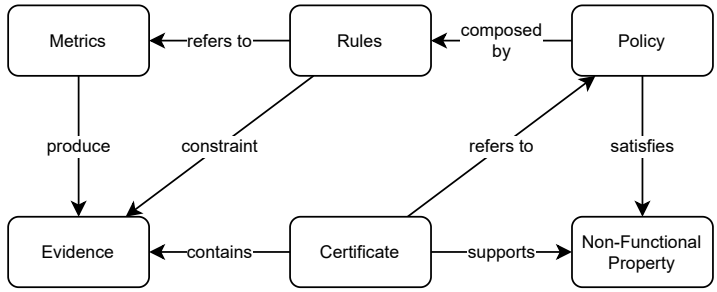


Figure 6.3: Abstract certification model.

at the basis of our certification process, where: i) *Certificate* is the result of a *Contract* verification aimed to prove a behaviour supporting a given NFP to be certified; ii) each *Contract* is composed of set of *Rules* that are expressed in our specification language and are based on *Metrics* captured on the system under certification; iii) each metric produces the *Evidence* stored in the certificate to support the given NFP.

6.3.2 Metrics

A metric, as retrieved by the Measurement Collection Process, is a function defined i) on a single node focused on measuring its attributes (e.g., the amount of available memory), or ii) on multiple nodes to measure their interaction (e.g., minimum network bandwidth between any points). A metric function should be compliant with key requirements that influence its accuracy and correctness:

- each metric must focus on a singular aspect of the system, preventing unnecessary duplication;

- the computational effort necessary must be negligible, compared to the descriptive value of the output;
- metrics must be calculated in parallel, minimizing the total waiting time;
- metrics must show the temporal evolution of the system;
- metrics must not interfere with the system processes, including the network protocol;
- metrics should be as much as possible scenario and user independent.

Given the fact that the measurements used to compute metrics spans over a time frame, metrics are often significantly affected by the interval of time considered during their evaluation. For instance, a metric that calculates the average load in a network node in a large time span may hide significant spikes that a stricter evaluation could identify. Given a specific input time interval, a metric function produces a measurement of a specific aspect of the system for that time interval. Formally, we define a metric as follows.

Definition 6.3.1. Given $\mathbf{N}$ the set of all network nodes, $\mathbb{T}$ a space of time intervals and $\mathbb{V}$ the space of possible values that a metric can assume, metrics $\mathbb{M}$ can be defined as:

$$\mathbb{M} : \mathbf{N} \times \mathbb{T} \rightarrow \mathbb{V}$$

We note that a metric $m \in \mathbb{M}$ can be evaluated for any sets of nodes $\mathbf{n} \subseteq \mathbf{N}$ in an interval of time $\mathbf{T} \in \mathbb{T}$, denoted as $m(\mathbf{n}, \mathbf{T})$. Each metric may evaluate the target nodes' state in several time points inside the chosen interval, depending on its implementation. We also note that metric values $\mathbb{V}$ are bounded to specific data types and values ensuring a finite and concrete representation. Metrics evaluated on a target system produce a simplified vision of its internal state, hiding unnecessary complexity while extracting significant information.

We note that, evaluating the metric against a chosen subset of nodes in $\mathbf{N}$ in a given instant of time we can map each node to a partial order. This approach helps in efficiently exploring network nodes during the evaluation of a set of contracts by defining appropriate heuristics.

6.3.3 Rules

Rules are Boolean functions based on one or more metrics and an interval of time. We formally define a rule as follows.

Definition 6.3.2. Given $\mathbf{N}$ the set of all network nodes, $\mathbb{T}$ a space of time intervals, and $\mathbb{B} = \{\texttt{true}, \texttt{false}\}$, rules $\mathbf{R}$ is defined as:

$$\mathbf{R} : \mathbf{N} \times \mathbb{T} \rightarrow \mathbb{B}$$

Each rule is a Boolean expression grounded on the simplified BNF described in Figure 6.4.

$$\begin{aligned}
D_Value &::= \langle constant \rangle \mid \langle metric\ evaluation \rangle \\
D_Acc &::= \langle D_Value \rangle [\langle constant \rangle] \mid \\
&\quad \langle D_Value \rangle [\langle constant \rangle : \langle constant \rangle] \\
D_Expr &::= \langle D_Value \rangle \mid \langle D_Acc \rangle \mid \\
&\quad \langle D_Expr \rangle \langle D_Op \rangle \langle D_Expr \rangle \mid \\
&\quad \langle D_Transf \rangle (\langle D_Value \rangle) \\
D_Op &::= + \mid - \mid \times \mid / \\
D_Transf &::= \sum \mid \prod \mid min \mid max \mid abs \mid len \mid \ldots \\
D_Cmp_Op &::= < \mid \leqslant \mid = \mid \neq \mid \geqslant \mid > \\
B_Value &::= true \mid false \mid \langle rule\ evaluation \rangle \mid \\
&\quad \langle D_Expr \rangle \langle D_Cmp_Op \rangle \langle D_Expr \rangle \\
B_Op &::= \wedge \mid \vee \mid \equiv \mid \oplus \\
B_Expr &::= \langle B_Value \rangle \mid !(\langle B_Expr \rangle) \mid \\
&\quad \langle B_Expr \rangle \langle B_Op \rangle \langle B_Expr \rangle \\
Rule &::= \langle rule\ name \rangle (t_1, t_2) = \langle B_Expr \rangle
\end{aligned}$$

Figure 6.4: Rules expressed in a BNF notation. For the sake of simplicity, we will skip some trivial non-terminals.

In this notation, $\langle constant \rangle$ is a value in $\mathbb{V}$, $\langle metric\ evaluation \rangle$ is in the form $m_i(\mathbf{T}', \mathbf{n}')$, and $\langle rule\ name \rangle$ is an unique rule identifier in the form

$r_i(\mathbf{T}', \mathbf{n}')$. Following the abstraction notation, $\mathbf{T}, \mathbf{T}' \in \mathbb{T}$ and $\mathbf{n}, \mathbf{n}' \in \mathbf{N}$. Since several rules can share parts of definition, we included $\langle rule\ evaluation \rangle$ in the BNF, allowing a rule evaluation to be called from within another rule, even on a different time interval. Metric and rule evaluation can only be applied within the time interval and nodes of the original rule, such that $\mathbf{T}' \subseteq \mathbf{T}$ and $\mathbf{n}' \subseteq \mathbf{n}$, forbidding recursively defined rules over shifting time intervals.

For each rule a partial order $(\mathbf{R}, \preceq_r)$ that indicates the strictness of the rule is defined as follows.

$$r_a \preceq_r r_b \iff \forall\, \mathbf{n} \in \mathbf{N}, \mathbf{T} \in \mathbb{T} \\ r_b(\mathbf{n}, \mathbf{T}) \Rightarrow r_a(\mathbf{n}, \mathbf{T})$$

Example 6.3.1. Consider the two rules $r_i(\mathbf{n}, \mathbf{T}) = m(\mathbf{n}, \mathbf{T}) \leqslant 10$ and $r_j(\mathbf{n}, \mathbf{T}) = m(\mathbf{n}, \mathbf{T}) \leqslant 5$ where m indicates the maximum RTT in ms allowed between the nodes in $\mathbf{n}$ in the time interval $\mathbf{T}$. We can see how r_j is stricter than r_i, since all nodes in which r_j is valid will also be valid for r_i while the opposite is not true, therefore $r_i \preceq_r r_j$.

6.3.4 Contract

A contract describes the expected behaviour of a group of nodes by indicating a set of rules that should be positively evaluated. We define a contract as a subset in the powerset of the space of all possible rules. More formally we define the set of all contracts as

$$\mathbf{P} = \wp(\mathbf{R})$$

Notice that we use $\mathbf{P}$ with the assumption that for each property exists one and only one contract that verifies it. In case of several version of contracts for the same property, we may annotate p with and identifier. Contract are evaluated by combining the output of each of their rules: a contract is verified for a given set of nodes $\mathbf{n} \in \mathbf{N}$ in an interval of time $\mathbf{T} \in \mathbb{T}$ if and only if all of its rules evaluated in $\mathbf{n}$ and $\mathbf{T}$ produce a positive output. We note that a contract including pairs of conflicting rules, for instance $\{m(\mathbf{T}, \mathbf{n}) = 1, m(\mathbf{T}, \mathbf{n}) = 2\}$, produces a negative output. By contrast, the contract corresponding to the empty set of rules produces a positive output by default, regardless of which set of nodes and time intervals are given as input.

Due to the inherent compositional nature of rules in our model, we can define a partial order $(\mathbf{P}, \preceq_p)$ as follows:

$$a \preceq_p b \iff \forall\, r_a(\mathbf{n_a}, \mathbf{t_a}) \in a \,\exists\, r_b, \mathbf{t_b}, \mathbf{n_b} \text{ where } r_a \preceq_r r_b, \mathbf{t_a} \subseteq \mathbf{t_b}, \mathbf{n_a} \subseteq \mathbf{n_b} \mid r_b(\mathbf{n_b}, \mathbf{t_b}) \in b$$

$a \preceq_p b$ if and only if b contains at least the same or stricter rules than a and each rule in b is evaluated on a superset of the time intervals and on a superset of the set of nodes of its counterpart in a.

Contracts are defined as sets of rules, therefore we can combine multiple contracts together in a single set. Exploiting the partial order $(\mathbf{P}, \preceq_p)$, we can define a contract $\mathbf{P}$ as the Least Upper Bound (LUB) of several contracts, producing the equivalent of concatenating the contract rules with the logical *and* operator. This more effective than a simple union as we can shrink multiple version of the same rules to a single stricter one.

Contracts can also be used as a selection mechanism for identifying a subset of nodes within the network with peculiar characteristics. Given a target contract p that we want to validate, we can verify which subsets of network nodes in a set $\mathbf{N}$ verify the contract. A contract-based filter can be generated by combination of multiple contracts using the Greatest Lower Bound (GLB) operator such that $p = glb(\mathbf{P})$, where $\mathbf{P}$ is the set of target contracts. This is equivalent to concatenate the contract rules with the logical *or* operator. A typical use case for such an approach is the deployment of a service over a set of nodes all ensuring a contract such as data replication grade or channel encryption.

6.3.5 Certificate

Certificate is the outcome of the certification process and are composed of:

- the contracts a certificate proves;
- the validation parameters, such as the target set of nodes and the time interval;

- the evidence supporting the certificate and verified by the involved contracts.

Definition 6.3.3. We define a certificate as a tuple $\langle \mathbf{T}, \mathbf{n}, \mathbf{P}, \mathbf{v} \rangle$ where $\mathbf{T}$ is a time interval in $\mathbb{T}$, $\mathbf{n}$ is a set of nodes in $\mathbf{N}$, p is a contract that has been verified in the interval $\mathbf{T}$ for all nodes in $\mathbf{n}$, and $\mathbf{v}$ is the set of evidence related to nodes in $\mathbf{n}$ evaluated during the verification of contract p.

$\mathbf{v}$ is optional to protect against the release of sensitive information. A certificate is awarded by a AL if and only if the contract $\mathbf{P}$ has been successfully verified.

6.3.6 Non-Functional Properties

A NFP is an abstract concept that identifies the expected status of a system. Our model treats property as a generalization of one or multiple verified contracts, meaning that a group of nodes $\mathbf{n}$ has a property p in the interval $\mathbf{T}$ if a selected set of contracts has been verified. Property is verified according to different and non-intersecting contracts. For instance, property *confidentiality* is verified if all contracts ensuring an encrypted network traffic are verified. Properties that describe the same concept can have several degrees of satisfaction depending on which of the associated contracts have been verified. In the previous example, a contract that ensures all traffic is encrypted using an RSA key of length 2048 bits is weaker than a contract requiring a key length of 4096 bits.

We note that different clients can define properties using different sets of contracts, depending on their requirements and use cases.

6.4 Certification Services

The abstract certification model in Section 6.3.1 is instantiated in the certification processes in Sections 6.5 and 6.6 using the the measurement collection and contract verification services described in the remaining of this section.

6.4.1 Measurement Collection Service

The measurement collection service allows ALs to query the internal state of any target nodes through their metrics. There are three different ways to implement our measurement collection service: i) pull, ii) push, and iii) hybrid. The pull solution works as follows:

1. each target node executes a service that binds to a known prefix listening for measurement requests in the form `/…/`⟨*node*⟩`/measure/`⟨*metric*⟩`/`⟨*params*⟩, where ⟨*metric*⟩ uniquely identifies the metric chosen by the AL, while ⟨*parameters*⟩ indicates the metric parameters such as the interval of time used to measure;

2. a AL can repeatedly send interest requests to a node with the necessary fields to query its state;

3. when a target node receives a valid measurement request, it replies with a data packet containing the result of the metric evaluation with the given parameters;

4. data packets can be cached, supporting the efficient distribution of the measurements to the ALs that sent a matching request;

5. the contents of the data packets can be encrypted to preserve confidentiality.

This approach has the disadvantage of requiring the ALs to know the prefix of a possibly large number of nodes, but leaves total control on the ALs over which metrics need to be queried and when.

The push solution works as follows:

1. each AL executes a service that binds to a known prefix listening for measurement updates request in the form `/…/`⟨*node*⟩`/update/`⟨*measurements*⟩, where ⟨*measurements*⟩ is an encoded list of measurements;

2. each target node hosts a service that periodically evaluates all metrics using a fixed set of parameters;

3. after each iteration, the node sends the AL an update over the newly obtained measurements.

This solution moves the responsibility of maintaining synchronization from the AL to the nodes and reduces synchronization delays. Unfortunately, it also increases the total amount of data sent, as even unnecessary measurements can be contained in the packets. Moreover the ICN caching mechanism cannot be used for requests, therefore the total network traffic would increase significantly in the case of multiple ALs. While this method is possible, it can experience efficiency and scalability issues.

The hybrid solution combines the pull and push implementations. Depending on the amount of updated data to share and the size of the network, it can improve the synchronization with a limited increase in traffic. It works as follows:

1. when an update is ready a node sends a small notification request to the ALs;

2. then, the ALs can request the status of the node as in the pull solution.

This addition helps in synchronizing the two parties, reducing the idle time from when the information is ready and when it is collected by the ALs, while maintaining the advantages of ICN caching mechanism. However, it also introduces complexity and additional traffic compared to the push solution.

Our model implements the pull solution and supports the hybrid one, providing the best tradeoff in complexity and network usage. Each node in the network exposes a predefined prefix in the form `/…/`⟨*node*⟩`/measure/list`, which returns the list of available metrics and a prefix in the form `/…/`⟨*node*⟩`/measure/`⟨*metric*⟩`/[to]/[from]` allowing other nodes to query its metrics. Depending on the type of metric, the two parameters `to` and `from` can be optional.

6.4.2 Contract Verification Service

Contract Verification Service formalizes how clients can request verifications to ALs and the certification response. The implementation of such service

with the pull approach can be summarized as follows:

1. each AL executes a service that binds to a known prefix listening for certification requests in the form `/…/`⟨*node*⟩`/verify/`⟨*parameters*⟩, where ⟨*parameters*⟩ indicates the verification parameters, including which contract, time interval, and nodes subset to use in the evaluation;

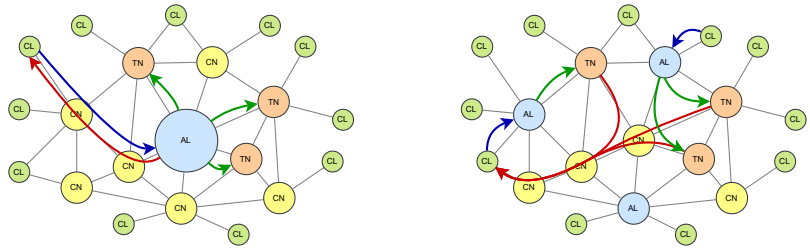
2. for each valid certification request, the AL service executes a certification process, as described in Sections 6.5 and 6.6, which produces a list of content names, each pointing to a certificate;

3. once the certification process is terminated, the service responds to the client request with the list of certificates produced in the form of content names.

These responses can be cached, allowing other clients with matching requests to be immediately satisfied.

6.5 Centralized Certification Process

We instantiate the abstract certification model in Section 6.3.1 using the certification services in Section 6.4 and the centralized certification process in Figure 6.5a, where the AL mediates all certification activities.



(a) Centralized certification process. (b) Decentralized certification process.

Figure 6.5: Abstract certification model instantiation. Clients (CL) request a contract verification on a set of target nodes (TNs) to Accredited Lab (AL).

6.5.1 Network Model

Figure 6.5a presents a centralized network model at the basis of a centralized certification process, where a single AL is responsible for all certification activities and any nodes can be both certification client and target. While this approach introduces a single point of failure on the AL, which also becomes a significant bottleneck in larger networks, it introduces some major advantages as follows.

- *Service discovery.* The AL knows the prefixes exposed by the target nodes to query their metrics. With a centralized network a common approach to service discoverability is to use a registration approach so that i) the AL prefix is known to any nodes in the network and ii) each node that connects to the network notifies its prefix to the AL through a registration request. This solution is simple to implement and does not rely on protocol-specific service discovery features.

- *Simpler certificate distribution.* The AL distributes the certificates it produces as contents of a self-owned predefined prefix. The nodes that are awarded with a certificate are notified by the AL. This solution allows any clients in the network to query for a certificate knowing only the AL's base prefix, while exploiting the caching capabilities of the network for an efficient distribution of common data requested by multiple clients.

- *Results caching.* The AL is the only actor receiving certification requests and producing corresponding certificates. Caching of previously verified contracts is effective, reducing the number of network requests necessary to evaluate new requests.

6.5.2 Certification Process

Algorithm 1 presents the centralized certification process and corresponding contract verification, where a is a certification client, c a AL, $\mathbf{n}$ a subset of nodes, and $\mathbf{T}$ a time interval. Figure 6.6 visually represents the steps of the algorithm in an example network.

A contract verification request sent by a certification client (line 22) is handled by a AL. The certification process starts by checking whether the locally cached certificates already verify the target contract; an initially empty contract is expanded by applying the GLB operator (line 2–6). If the target contract is smaller in $\preceq_p$ than the obtained set, the target contract is verified and the list of cached certificates returned as output (lines 7–8). If the cached certificates are insufficient, the certification process proceeds by evaluating each rule that is not verified yet (lines 9–13). The certification process finally checks if all the rules have been verified; if yes, it generates and returns a certificate to the client, otherwise, it returns an empty list (lines 14–19). Finally, the client receives the list of certificates (line 23).

Algorithm 1 Centralized certification process.

```
1:  function HANDLE REQUEST(contract: p(n, T))
2:      v_res = ∅
3:      for all v_cached ∈ cached_certificates() do
4:          for all rule r(n, T) ∈ p(n, T) do
5:              if r(n, T) ⪯_p v_cached then
6:                  v_res = GLB(v_res, v_cached)
7:      if p(n, T) ⪯_p v_res then
8:          return v_res
9:      for all rule r(n, T) ∈ p(n, T) do
10:         if r(n, T) ⋠_p v_res then
11:             for all metric evaluation m(b', T') ∈ r(n, T) do
12:                 m_res[m] = m(b', T')
13:     p_valid = ⋀_{r(n,T)∈p(n,T)} r(n, T)
14:     if p_valid then
15:         cert = new_certificate(p(n, T))
16:         v_res = {cert}
17:     else
18:         v_res = ∅
19:     return v_res
20:
21: function REQUEST VERIFICATION(contract : p(n, T))
22:     res = send_request(contract)
23:     certs = collect_certs(res)
```

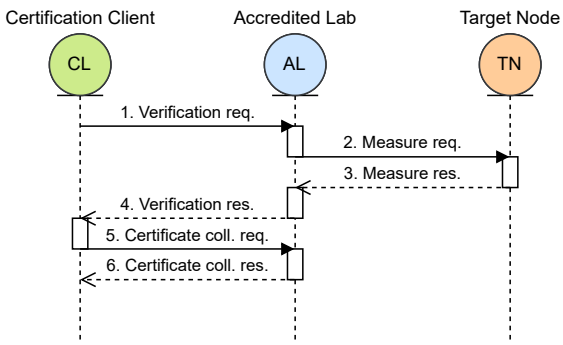


Figure 6.6: Centralized certification process: Communication flow.

6.6 Decentralized Certification Process

We the abstract certification model in Section 6.3.1 using the certification services in Section 6.4 and the decentralized certification process in Figure 6.5b, where multiple CAs manage the certification activities and their output can be independently combined.

6.6.1 Network Model

Figure 6.5b presents a decentralized network model at the basis of a decentralized certification process, where a every node in the network can act as a AL making the certification process completely decentralized. This approach eliminates the single point of failure of the centralized network model and allows clients to request certifications to several nodes in the network. It also enables clients to selectively specify the AL node on the basis of its trust level, possibly requiring the AL to filter out those certificates produced by untrusted sources. The decentralized approach provides additional advantages as follows.

- *Service discovery.* The distributed network model extends the previous one by including an automatic service discovery mechanism, which allows each node to search for ALs nodes in their proximity. As discussed in Section 6.4, ALs, as well as all nodes in the network, exposes

services on predefined prefixes. Depending on the capabilities of the underlying ICN protocol, the clients can either use in-protocol service discovery features to identify nodes with such prefixes, like Named Data Link State Routing (NLSR) in NDN, or send discovery requests to each network interface in a multicast fashion with an increasing maximum hops limit. This approach allows clients to operate independently from a central authority and to self-organize in spatially localized subnetworks.

- *Decentralized certificate distribution.* The tasks of storing and distributing the awarded certificates are outsourced from the AL to the target nodes. A AL that successfully certified a node sends a registration request submitting the signed certificate as a parameter of a predefined prefix of the target node in the form `/…/`⟨*node*⟩`/register/`⟨*certificate*⟩. A node that receives such a request stores the certificate in a local storage using a unique identifier, exposes the certificate as a content on a predefined prefix in the form `/…/`⟨*node*⟩`/certificate/`⟨*id*⟩, and responds to the registration request with the complete content name. The AL can then collect the list of certificate names, one for each target node, and answer to its certification client. The target nodes also expose their list of awarded certificates including the contract and parameters used on a predefined prefix in the form `/…/`⟨*node*⟩`/certificates/[filter]`, allowing ALs to easily query their storage for previous certificates that can used as a baseline for further verification. The certificates distribution is then decoupled from the ALs that produced them and rather controlled by the target nodes, while maintaining the effectiveness of the caching capabilities of ICN. The decentralized approach increases the total number of requests necessary to produce a certificate in small networks but strongly reduces traffic originated by packets being forwarded in large networks, with respect to what expected in the centralized solution. In other words, it prevents long paths from the periphery of the network to the central AL and vice versa.

- *Result caching.* Caching of previous results is more effective than the one in the centralized model. Each AL can store the certificates produced by itself and query others ALs's certificates directly to the target nodes. This approach enables a distributed and cooperative certification service, where each verification can exploit previously verified contracts to produce new knowledge.

- *Contract query service.* The contract verification process in our distributed network model employs an additional network service to allow ALs to query target nodes for stored certificates matching a minimum contract. These targets listen for query requests on a predefined prefix in the form `/…/`⟨*node*⟩`/certificates/`⟨*filter*⟩, where filter is an encoded contract definition. When a request is received the node iterates over its certificates, checks which ones pass the filter, collects their content name in a list and returns it to the requester. This solution allows ALs to rapidly collect information about previously verified contracts over their target nodes without requiring a network-wide level of synchronization over the status of certificates.

6.6.2 Certification Process

Algorithm 2 presents the decentralized certification process and corresponding contract verification, where a is a certification client, c a AL, $\mathbf{n}$ a subset of nodes, and $\mathbf{T}$ a time interval.

A contract verification request sent by a certification client (line 32) is handled by a AL. The certification process starts by checking whether the locally cached certificates already verify the target contract: an initially empty contract is expanded by applying the GLB operator (line 2–6). If the target contract is smaller in $\preceq_p$ than the obtained set, the target contract is verified and the list of cached certificates returned as output (lines 7–8). If the cached certificates are insufficient, the certification process queries the neighbour nodes for certificates that satisfy the inner rule evaluations and merges them to the previous partial solution (lines 9–13). If the obtained solution is sufficient, it returns the list of certificates (lines 14–15); otherwise, the certification process proceeds by evaluating each inner rule that is not verified yet (lines 16–21). The certification process then checks if all the inner rule evaluations have been verified; if yes, it generates and returns a certificate to the client, otherwise, it returns an empty list (lines 22–29). Finally, the client receives the list of certificates (line 33).

The evaluation of certificates stored locally or on neighbour nodes (lines 9–13) could be insufficient to verify the target property. This means that at least one among the set of rules, the set of nodes, or the time interval causes a failure in the evaluation. Automatic and timely identification of the cause of the failure

Algorithm 2 Decentralized certification process.

```
1: function HANDLE REQUEST(contract: p(n, T))
2:     v_res = ∅
3:     for all v_cached ∈ cached_certificates() do
4:         for all rule r(n, T) ∈ p(n, T) do
5:             if r(n, T) ⪯_p v_cached then
6:                 v_res = GLB(v_res, v_cached)
7:     if p(n, T) ⪯_p v_res then
8:         return v_res
9:     for all rule r(n, T) ∈ p(n, T) do
10:        cert_names = query_certs(r(n, T))
11:        certs = collect_certs(cert_names)
12:        for all v_cached ∈ certs do
13:            v_res = GLB(v_res, v_cached)
14:    if p(n, T) ⪯_p v_res then
15:        return v_res
16:    for all rule r(n, T) ∈ p(n, T) do
17:        if r(n, T) ⋠_p v_res then
18:            for all rule r ∈ p' do
19:                for all metric evaluation m(b', T') ∈ r(n, T) do
20:                    m_res[m] = m(b', T')
21:    p_valid = ⋀_{r(n,T)∈p(n,T)} r(n, T)
22:    if p_valid then
23:        cert = new_certificate(p(n, T))
24:        v_res = {cert}
25:        for all b ∈ n do
26:            notify_new_certificate(b, cert)
27:    else
28:        v_res = ∅
29:    return v_res
30:
31: function REQUEST VERIFICATION(contract : p(n, T))
32:    res = send_request(contract)
33:    certs = collect_certs(res)
```

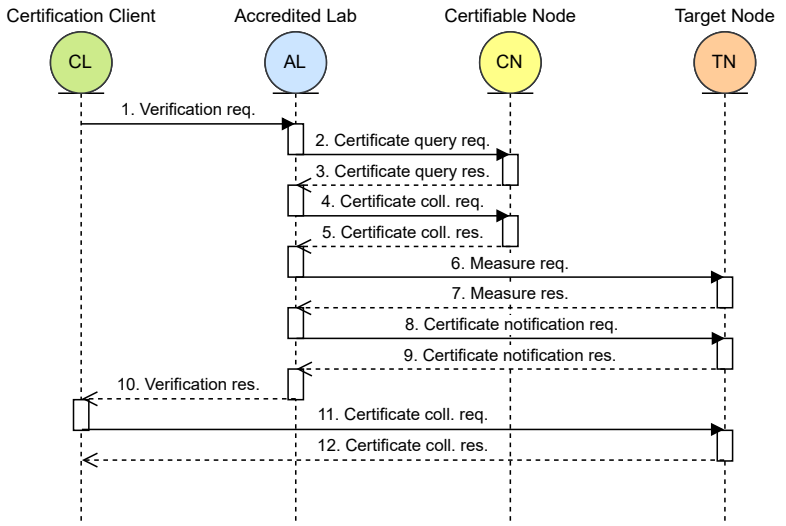


Figure 6.7: Decentralized certification process: Communication Flow.

permits to rapidly identify which inner rule evaluations are missing to meet the contract requirements and, in turn, ask them to other nodes or manually verify them. We present two examples where the evaluation of certificates stored locally or on neighbour nodes prevent the AL to re-evaluate already verified contracts or to reuse partial results.

Example 6.6.1. Consider the case of a contract that checks whether a rule r is valid for the interval $\mathbf{T}$ for the nodes $\mathbf{n}$. The AL queries the nodes in $\mathbf{n}$ and retrieves a certificate that validates r for all nodes in n' where $\mathbf{n} \subseteq \mathbf{n}'$ for the interval $\mathbf{T}'$ with $t \subseteq t'$. It follows that, if AL trusts the certificate results, the rule r has already been verified and thus also the contract.

Example 6.6.2. Consider the case of a contract that checks a rule r is valid for the interval $\mathbf{T} = \langle t_1, t_2 \rangle$ for the nodes $\mathbf{n}$. The AL queries the nodes in $\mathbf{n}$ and retrieves a certificate that validates r for all nodes in $\mathbf{n}$ for the interval $\mathbf{T}' = \langle t'_1, t'_2 \rangle$ with $t'_1 \in [t_1, t_2]$. It follows that, if AL trusts the certificate results, it needs to verify r only for the interval $[t'_1, t_2]$. A similar scenario considers a rule already verified on a subset of the nodes; in this case the AL could trust the certificate and verify only the unchecked nodes.

We note that the decentralized certification process proposes a collaborative approach designed to exploit the caching capabilities of Information-Centric Networks and improve the overall system performance. The decentralized approach outperforms the performance of the standard centralized process based on a AL managing the entire certification activities, also improving its security and addressing the problem of a single point of failure. The centralized approach has the main benefit of being inline with current certification frameworks.

6.7 Discussion

While current literature has already demonstrated the effectiveness of ICN networks in large and complex scenarios, it does not include any unified solutions for monitoring and certification of such networks. In this work we presented a certification methodology for such networks, capable of efficiently verifying complex contracts to ensure the expected levels of QoS. We defined an abstract certification model that can be easily adapted for a large variety of applications, from Service Level Agreements monitoring to attack detection systems. We believe this work is an important step in the evolution and diffusion of ICN based services in the field of edge and cloud computing, as a more efficient and trustworthy solution. The certification methodology in this work is fully compatible with ICN and does not require changes at protocol level. It can also substantially improve ICN functionalities in different scenarios that are summarized in the following of this section.

6.7.1 Network Adaptation

Modern networks have strong flexibility requirements, especially in mobile contexts, with devices entering and exiting the network, or even moving inside it, large spikes of traffic, and an ever increasing variety of network services. Effective adaptation is a hard problem, especially in large scale networks like national ISP networks, where the number of connected devices easily exceeds millions. The decentralized certification process in Section 6.6 allows clusters of devices to self-regulate based on inferred network properties, while maintaining high levels of trust and privacy.

As an example, let us consider a scenario in which the network is capable of detecting a malfunctioning or compromised node, requiring that all the traffic is routed to an alternative path. This approach is viable for large monitored network nodes, like ISPs, cloud centres, and large firms ingress points, where teams of experts are available and the computational power is not a limiting factor. On the other hand, small scale networks, like offices, districts switches, hospitals, and small companies likely cannot afford a dedicated team. A network adaptation solution based on our distributed certification process permits the definition of automatic security measures to respond to the emergency, while requiring far less resources and expertise.

6.7.2 Secure Service Deployment

The deployment of a service in a set of network of nodes (e.g., in a public cloud, a multi-tenant environment, a cluster of servers) raises security concerns. Which properties can the host guarantee? What level of security can we expect from the nodes? Would the nodes have the necessary resources to run the service?

A certification process permits to verify contracts and identify whether a certain set of nodes is suitable for the deployment of the chosen services. Our certification process can both evaluate if a given set of nodes is suitable for the deployment of the chosen services and filter from an arbitrary large set of nodes the most suitable ones. This can be achieved by first using a contract-based filtering over the whole network and then a metric-based ordering on the remaining nodes. We note that this approach can be integrated in service deployment schedulers, to improve their effectiveness and enforcing resource or security constraints.

6.7.3 Attack Detection

Our certification process can be used to continuously monitor a target network with the goal of identifying misbehaviour instead of certifying specific NFPs. This can be easily obtained with ad hoc contracts focused on anomaly detection. An example of contract that can be used for anomaly detection contracts is the one focused on detection of cache pollution attacks, a DOS

technique used to incapacitate the cache of an ICN network nodes by forcing them to store useless or unpopular data. Rules that analyse the network traffic statistics can rapidly identify changes in the type of contents stored across the network and alert of a possible attack event.

Chapter 7

Assurance in Big Data Analysis Platforms

We live and operate in a data-driven ecosystem where huge amounts of data are collected, shared, and analysed by multiple actors working within and across organizational boundaries. The benefits brought by this data-driven ecosystem in terms of value, performance, and quality, come at the price of increasing security and privacy risks. Data, in fact, can be sensitive and need to be protected and secured once stored and while processed, following strict regulations such as the General Data Privacy Regulation (GDPR) in Europe.

A number of different solutions protect the Big Data infrastructures and their data/processes by internal and external threats and attacks, resulting in the proliferation of ad hoc solutions that prove a specific property or compliance to a specific regulation [56–58]. Each solution targets a very small part of the whole problem [59–63], missing the full picture. For example, Terzi et al. [59] presented a survey on a global perspective of Big Data security and privacy, while Yakoubov et al. [60] specifically focused on cryptographic approaches. Zhang et al. [61] proposed a scalable differential privacy approach for Big Data multidimensional anonymisation based on MapReduce. In addition, the research community has started approaching the problem of protecting ML algorithms and corresponding modes [64, 65] at the core of Big Data systems, while neglecting the protection of the whole systems.

The need of trust and transparency of Big Data computations is clearly raising, representing a major barrier against the adoption of Big Data technologies especially in a multi-tenant environment. Service providers (data transformers/analysers) are reluctant to take full responsibility over security and privacy breaches of their services. Customers (data owners/suppliers) do not have access to all security intelligence and log information, which impairs their ability to estimate risks. There is no evidence that their computations and information are correctly managed and protected, as well as on the status of service security and correct behaviour of security and privacy controls.

This chapter completely rethinks our previous work [24], where we tried to address the above gaps but in the framework of DevOps processes applied to Big Data pipelines. In this chapter, we depart from the development process adopted in [24] and, to the best of our knowledge, present the first attempt to address the general problem of *Big Data pipeline assurance* in an holistic manner.

Assurance is the way to gain justifiable confidence that i) one or more security properties are consistently demonstrated by the target of an assurance evaluation and ii) the target operationally behaves as expected, despite failures and attacks [66]. Applying assurance to Big Data is a complex process that evaluates the trustworthiness at multiple layers: iii) the Big Data pipeline and all its tasks, iv) the Big Data engine and all services over which the pipeline is executed. The goal of our assurance solution is to increase the trustworthiness of Big Data applications, mitigating the typical user distrust in Big Data environments[1]. This distrust is typically based on the fact that service providers and customers lose, at least partly, control over the status of their data and applications, and Big Data technologies and analytics provide blazing fast inference on such data. Few approaches have already focused on Big Data assurance, tackling specific aspects like data integrity or authentication [67–69]. Gao et al. [70] focused on Big Data quality assurance, and just partially considered security and privacy quality metrics. Presenting a comparison of Big Data validation tools, the chapter underlines a set of needs including lack of well-defined quality validation and assurance standards, as well as lack of available research results on quality models/metrics and certification programs. Some initial assurance solutions also targeted the need of verifying NFPs of ML models to the aim of implementing trustworthy

[1]Ernst and Young - `https://www.ey.com/en_es/assurance/how-big-data-and-analytics-are-transforming-the-audit`

decision systems [71].

The scientific contribution of this chapter is twofold. First, we propose a novel assurance process and architecture that evaluates the trustworthiness of Big Data pipelines at all layers, including the Big Data environment. Second, we define an assurance methodology where: i) clients annotate a pipeline template with assurance requirements modelling their trust expectations in terms of NFPs ii) a pipeline instance is generated from the template mapping all tasks, services, and requirements on real components, and iii) an assurance confidence level is calculated modelling the trustworthiness level of the Big Data pipeline. We experimentally evaluated our approach in the context of the H2020 EVOTION Policy-Making Big Data Platform [72], where policy-makers design or select analytics templates to be instantiated and executed by the platform in a fully assisted and privacy-preserving way. We note that our approach further complements the 5V definition of Big Data [73] where *veracity* (i.e., messiness or trustworthiness of the data) is enriched with Big Data assurance, where Big Data trustworthiness is complemented with the Big Data transparency and the soundness of Big Data computations [74].

The chapter is organized as follows. Section 7.1 defines the assurance process and the software architecture implementing and automating its evaluation. Section 7.2 defines the abstraction over the data pipeline and the Big Data environment needed to represent the process. Section 7.3 introduces a practical scenario where security assurance techniques are applied to a real Big Data pipeline, describing its tasks and its ecosystem. Section 7.4 describes our assurance methodology. Section 11.4 shows our experimental results considering different execution scenarios. Finally, Section 7.5 presents our remarks.

7.1 Assurance Process and Architecture

Non-functional assurance is the degree of confidence to which a system supports non-functional (e.g., security and privacy) requirements. A number of recommendations and security benchmarks have been proposed to support assurance activities such as the ones by *Cloud Security Alliance* [58, 75]. In this chapter, we propose an assurance process (see Section 7.1.1) that measures how much the security controls in the system contributes to a specific NFP and the degree of confidence held by such verification. For instance, let us

consider a requirement on data confidentiality both in transit and at rest for a given target system. Let also assume that it insists on two *security controls* implementing i) a mechanism for channel and storage encryption and ii) an access control mechanism mediating data ingestion. Our process measures the support, and corresponding degree of confidence, for property data confidentiality, by verifying the configuration of the encryption algorithm (e.g., encryption algorithm and key length), on one side, and the configuration of the access control system (e.g., type of access control and soundness of defined policies), on the other side.

7.1.1 Assurance Process

Figure 7.1 shows a methodological view of our assurance process. It is driven by a given requirement $r \in \mathcal{R}$ and refers to a given target τ, and aims to compute the assurance confidence level of r on τ. The assurance confidence level is computed by our assurance process according to the outcomes of the evaluation process, which evaluates the evidence collected on the target system τ. Such evidence is collected by our assessment process inspecting the target system τ with respect to the given requirement $r \in \mathcal{R}$.

Definition 7.1.1 (Assessment process)**.** Given a non-functional requirement $r \in \mathcal{R}$ and a target τ, an assessment process $ev = P(r, \tau)$ is a process P that collects evidence ev related to a given requirement r on τ.

An assessment process is usually based on testing, monitoring, or formal methods [50,76,77], and produces an evidence in the form of test cases results, monitored events, and formal proofs, respectively. For instance, let us consider the encryption control for requirement confidentiality of data in transit. An assessment process can recurrently test whether the selected encryption algorithm follows specific standards/regulations. Service configurations can be then verified (e.g., by parsing the configuration files) to evaluate requirements on the key length. We note that multiple assessment processes P_i can be executed in relation to a single requirement $r \in \mathcal{R}$ and, in turn, multiple evidence ev_i can be collected.

We define an evaluation process to verify whether a set of collected evidence support a given requirement $r \in \mathcal{R}$ on a given target τ as follows.

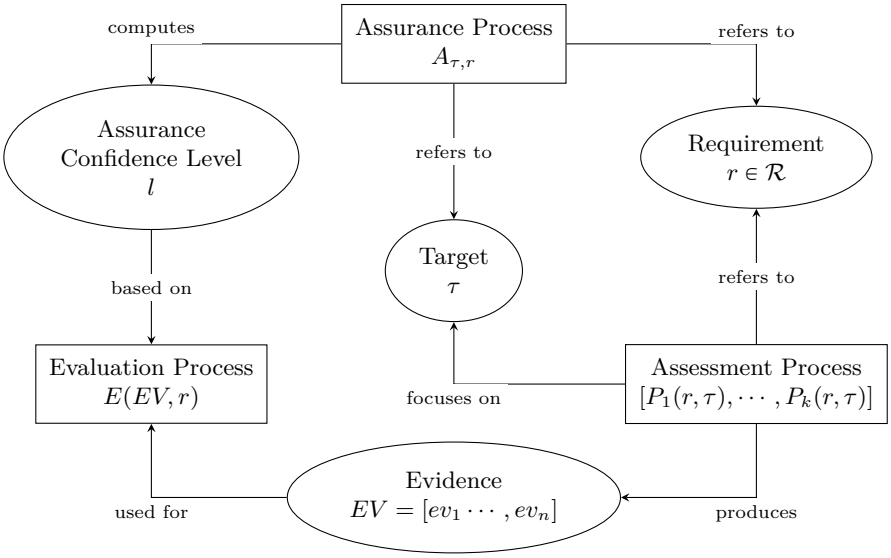


Figure 7.1: Methodological view of our assurance process.

Definition 7.1.2 (Evaluation process)**.** Let us consider a set of evidence $EV = \{ev_1, \ldots, ev_n\}$ retrieved by a (set of) assessment processes $\{P_1(r,\tau), \ldots, P_k(r,\tau)\}$ executed on target τ for a specific requirement r. An evaluation process can be defined as a function $E(EV, r)$ returning a value in $[0, 1]$ identifying the strength of the evidence EV supporting r. $E(EV, r) = 1$ means that the evidence $ev \in EV$ fully supports r; $E(EV, r) = 0$ means no support.

The strength of the evidence supporting the claim can differ depending on different evaluation parameters such as the type of probe, and the number and quality of evidence collectors, to name but a few. We consider the approach used to compute whether a set of evidence EV supports or not a requirement r and the relative strength, out of the scope of this work (a possible approach can be found in [78]).We note that multiple evaluation processes E_i can be executed on specific subsets EV_i of evidence. We also note that our process aims to distinguish between positive evaluations (>0) at different levels of strength. Negative evaluations, due to insufficient/disproving evidence or lack of information, are considered at the same level. This is a conservative

approach, meaning that all systems not reaching a minimum assurance level are simply ignored.

Out assurance process can be finally defined as follows.

Definition 7.1.3 (Assurance process)**.** An assurance process $A_{\tau,r}$ for a specific target τ is a process that takes as input a tuple $\langle EV, r \rangle$, where EV is the set of evidence and $r \in \mathcal{R}$ a requirement, and returns as output the assurance confidence level $l \in [0, 1]$. It represents the level to which the collected evidence EV satisfies requirement r according to a set of evaluation processes $E_i(EV_i \subseteq EV, r)$ executed on the collected evidence EV (see Definition 7.1.2).

We compute the assurance confidence level as follows.

Definition 7.1.4 (Assurance confidence level)**.** Given a set of evaluation results E_i, the assurance confidence level l is computed as follows: $l = \frac{|E_i>0|}{|E_i|} \cdot avg(E_i > 0)$ where $\frac{|E_i>0|}{|E_i|}$ is the frequency of positive evaluations and $avg(E_i > 0)$ is the average value of the positive evaluations.

We note that evaluations not supporting a given requirement (i.e., $E_i = 0$) contributes in the frequency factor coherently with the Definition 7.1.2, where the degree of positive evaluations are measured.

For instance, let us assume that i) a test-based assessment of an encryption service provides evidence satisfying the expectation on the algorithm but not on the key length used for data encryption (which is lower than expected) and ii) a monitoring-based assessment provides positive evidence on data ingestion (e.g., logs show that only authorized users accessed ingested data). Collected evidence is evaluated following Definition 7.1.2 and two evaluations $E_1 = 0.5$ and $E_2 = 1$ retrieved. Confidence level is then computed according to Definition 7.1.4 as $l = \frac{2}{2} \cdot \frac{0.5+1}{2} = 1 \cdot 0.75 = 0.75$ and returned by the assurance process $A_{\tau,r}$ in Definition 7.1.3.

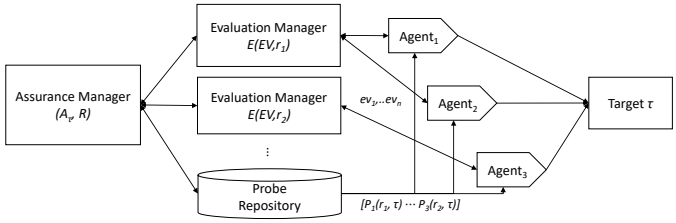


Figure 7.2: Assurance Architecture.

7.1.2 Assurance Architecture

Figure 7.2 describes our assurance architecture implementing the assurance process in Figure 7.1. *Assurance Manager* is the owner of the assurance process. It is responsible to i) set up the evaluation processes on the given target τ according to a specific set $\mathcal{R}$ of requirements and ii) collect the evaluation process results used to compute the assurance confidence level. *Evaluation Manager* manages all activities for the requirement evaluation. It is responsible to i) set up the assessment process and ii) collect the evidence used to evaluate whether a requirement $r \in \mathcal{R}$ is supported. The assessment process is implemented instrumenting different *Assessment Probes* that executes the assessment process on τ and retrieves the corresponding evidence *ev*. *Probes Repository* stores and maintains the probes, for instance, updating them in case new threats or guidelines are released. The probes are executed by *Assessment Agents* that are deployed and connected to the target of the assessment. We note that the probes in the probe repository can be configured by the assessment agents to inspect a specific target via specific target's hooks (e.g., APIs or the path of configuration files). They perform testing and monitoring activities, parse configuration files, execute code, and perform network traffic inspections (see Section 11.4).

When the assurance process verifies a set $\mathcal{R}$ of requirements, the following flow of actions is triggered. Assurance Manager instruments one or more Evaluation Managers with details on the evaluation to be executed, one for each requirement $r \in \mathcal{R}$. The Evaluation Managers communicate with the agents (deployed a priori) asking for the execution of probes addressing a specific assessment process. The agents download the requested probes from

the repository and execute them against the target returning the collected evidence back to the corresponding Evaluation Manager. The Evaluation Managers evaluate the collected evidence and return the evaluation results to the Assurance Manager. Assurance Manager finally computes the assurance confidence level.

7.2 Modelling Big Data Analytics Pipeline

We consider the Big Data analytics pipeline as the target τ of our assurance process in Figure 7.1, detailing it as i) a set of tasks $t \in T$ implementing the *processing pipeline* p and ii) a set of services $s \in S$ implementing the ecosystem e and supporting the deployment and execution of the processing pipeline. The pipeline and ecosystem are modelled at two levels of abstraction: i) *abstract level*, modelling the generic purposes of a task t and the abstract functionalities offered by a service s (template abstraction) and ii) *concrete level*, defining specific task implementations $\hat{t}$ in the pipeline and concrete services $\hat{s}$ used in the ecosystem (instance abstraction). We note that the decoupling of the pipeline of tasks from the ecosystem of services and the use of two levels of abstraction are fundamental to make the approach generic and its execution effective, separating the peculiarities of the pipeline from the capabilities of the ecosystem where the pipelines are executed.

7.2.1 Processing Pipeline

A processing pipeline transforms data according to a specific goal. We assume our pipelines to be a sequential composition of tasks t.[2] Its abstract view can be defined as follows.

Definition 7.2.1 (p)**.** An Abstract Processing Pipeline p is defined using a

[2]We note that more complex pipelines, including parallel or alternative tasks, can be generalized as a set of sequential pipelines similar to composition of services in [51].

BNF-like notation as

$$
\begin{aligned}
p &::= \langle T_I \oplus P \oplus A \oplus T_V \rangle \\
P &::= \epsilon \mid T_P \mid P \oplus T_P \\
A &::= \epsilon \mid T_A \mid A \oplus T_A \\
T_I &::= \mathit{stream} \mid \mathit{fileSystem} \mid \mathit{DBMS} \mid \ldots \\
T_P &::= \mathit{cleaning} \mid \mathit{normalization} \mid \mathit{selection} \mid \ldots \\
T_A &::= \mathit{modeling} \mid \mathit{prediction} \\
T_V &::= T_I \mid T_I \oplus \mathit{visualization} \mid
\end{aligned}
$$

Operator $\oplus$ is the sequence operator connecting tasks' input and output in a pipeline fashion. A generic task t is classified according to its processing type: i) ingestion tasks $T_I \subset T$ (e.g., *stream*, *fileSystem*, *DBMS*), ii) preparation tasks $T_P \subset T$ (e.g., *cleaning*, *normalization* and *selection*), iii) analytics tasks $T_A \subset T$ (e.g., *modelling*, *prediction*), iv) visualization tasks $T_V \subset T$ (e.g., *visualization*). We note that T_I and T_V are mandatory for p. We also note that ingestion tasks in T_I can be also used prior or replace the visualization in T_V.

An abstract processing pipeline p is instantiated in a concrete processing pipeline $\hat{p}$ as follows.

Definition 7.2.2 ($\hat{p}$)**.** Given an abstract processing pipeline p, a concrete processing pipeline $\hat{p}$ is produced by instantiating each generic task $t \in p$ in an executable task $\hat{t} \in \hat{p}$ in the form of a function call.

We denote this instantiation process as $p \xrightarrow{p} \hat{p}$.

Example 7.2.1. Let us consider an ingestion task $t_i \in T_I$, a preparation task $t_p \in T_P$ and a visualization task $t_v \in T_V$. An executable ingestion task $\hat{t}_i$ can ingest data from a queue system or files. An executable preparation task $\hat{t}_p$ can select data based on data columns/labels or apply a more sophisticated feature selection approach based on PCA or ICA. An executable visualization task $\hat{t}_v$ can save data to disk or send them to a visualization service.

7.2.2 Big Data ecosystem

A Big Data ecosystem is composed of services $s \in S$ supporting the execution of the processing pipeline. Its abstract view is defined as follows.

Definition 7.2.3 (e)**.** An abstract Big Data ecosystem e is a 5-tuple $\langle S_I, S_C, S_S, S_V, S_E \rangle$, where $S_S \subset S$ is a set of storage services, $S_C \subset S$ is a set of computational services, $S_I \subset S$ is a set of ingestion services supporting data collection, $S_V \subset S$ is a set of visualization services supporting the visualization of the pipeline outcomes, and $S_E \subset S$ is a set of environmental services offering additional non-functional capabilities.

We note that the ingestion services $s \in S_I$ (e.g., *streaming*, *load from batch*) are connected to tasks in T_I, while computational services $s \in S_C$ (e.g., *batch*, *stream*, *microbatch*) are connected to tasks in T_P and T_A to support preparation and analytics processing, and to tasks in T_I and T_V to support ingestion and visualization processing. Storage services $s \in S_S$ (e.g., *file storage*, *NoSQL storage*, *SQL storage*) are primarily connected to tasks in T_I and T_V and, if needed, to tasks in T_P and T_A to store/load temporary data during the preparation or analysis phases. Visualization services $s \in S_V$ (e.g., *dashboards*) are connected to tasks in T_V. Environmental services $s \in S_E$ (e.g., *access control*, *logging*, *annotation*, *authorization*) are connected to the entire pipeline.

Definition 7.2.4 ($\hat{e}$)**.** Given an abstract Big Data ecosystem e, a concrete Big Data ecosystem $\hat{e}$ is produced by instantiating each generic service $s \in e$ in a deployed service $\hat{s}$.

We denote this instantiation process as $e \xrightarrow{e} \hat{e}$.

Example 7.2.2. Let us consider Example 7.2.1 and a generic service $s_i \in e$ of type S_I, $s_c \in e$ of type S_C, and $s_v \in e$ of type S_V. s_i is instantiated into a service $\hat{s}_i$ deploying $HDFS$ and then used by the concrete ingestion task $\hat{t}_i$ in $\hat{p}$ to establish a connection and ingest data. s_c is instantiated into a service $\hat{s}_c$ based on $Spark$ computation engine used by the concrete preparation task $\hat{t}_p$ in $\hat{p}$ to elaborate ingested data. s_v is instantiated into a service $\hat{s}_v$ deploying

ApacheZeppelin(*params*) used by the concrete visualization task $\hat{t}_v$ in $\hat{p}$ to set up a *Zeppelin* visualization notebook where data can be visualized.

7.2.3 Building a Big Data Analytics Pipeline

A Big Data Analytics Pipeline instantiates the abstract processing pipeline and Big Data ecosystem as follows.

Definition 7.2.5. Given a pair $\langle p, e \rangle$, where p is the Abstract Processing Pipeline (Definition 7.2.1) and e is the corresponding Abstract Big Data Ecosystem e (Definition 7.2.3), a Big Data Analytics Pipeline is defined as a pair $\langle \hat{p}, \hat{e} \rangle$, where $\hat{p}$ is the concrete Processing Pipeline (Definition 7.2.2) and $\hat{e}$ is the corresponding concrete Big Data Ecosystem (Definition 7.2.4), such that $p \xrightarrow{p} \hat{p}$ and $e \xrightarrow{e} \hat{e}$.

In the following, we refer to $\langle p, e \rangle$ as Big Data Analytics Pipeline template Π and to $\langle \hat{p}, \hat{e} \rangle$ as Big Data Analytics Pipeline instance I. Template Π defines technology-independent processing pipelines and environment services; instance I is the concrete technology-dependent instantiation of Π. We assume a correct instantiation function $\xrightarrow{I} = \xrightarrow{p} \cup \xrightarrow{e}$ by using the approach in [79]. In other words, for each template Π, we assume one or more consistent instances I, defining a one-to-many relation between templates and instances.

7.3 Reference Scenario

Our reference scenario is the H2020 EVOTION Policy Making Big Data Platform (EVOTION Platform in the following) [72, 80], a collaborative solution offering analytics services or computing/data nodes based on different computation/storage frameworks. The platform, based on the Apache framework and Big Data Analytics-as-a-Service [79], offers an easy-to-use framework for policymakers to develop evidence-based policies following analytics results. Policymakers design or select specific analytics templates to be instantiated and executed by the platform in a fully assisted and privacy-preserving way. We note that this scenario is very challenging from an assurance evaluation

point of view, having pipeline composed as a service depending on the policymaker needs. We also note that assurance is fundamental to increase the trust in such a platform, which is executed in critical domains (e.g., health), where public policies and in turn pipelines use high sensitive data.

In our scenario, i) a policymaker c wants to continuously execute a Big Data analytics pipeline ensuring specific non-functional requirements $\mathcal{R}$, expressed in template Π; ii) the service provider *sp* offers via the EVOTION Platform a valid executable instance I for template Π.

Policymaker c requests verifiable assurance from service provider *sp* that its non-functional requirements $\mathcal{R}$ on template Π are guaranteed on the running instance I.

Such guarantees must be continuously checked by the assurance process to cope with emerging and evolving vulnerabilities and weaknesses. We remark that non-functional requirements $\mathcal{R}$ have the same level of abstraction used for the template, being independent by the specific pipeline implementation and deployment ecosystem, and taken from a controlled vocabulary.

Example 7.3.1 (Running Example)**.** Let us consider a template $\Pi = \langle p, e \rangle$ in Definition 7.2.5 defined as in Equation 7.1.

$$\begin{aligned} p &= \langle fileSystem \oplus normalization \oplus modelling(clustering) \oplus fileSystem \rangle \\ e &= \langle LoadFilesystem, \big[BatchProcessing, Orchestration\big], StoreFilesystem, \\ &\qquad \epsilon, AccessControl(AC) \rangle \end{aligned} \tag{7.1}$$

We note that the visualization service is not needed in this example, since it is focused on building a model. A concrete instantiation $\xrightarrow{I}$ of the above template Π into a *Big Data Analytics Pipeline Instance* I can be defined as $\langle \hat{p}, \hat{e} \rangle$, as in Equation 7.2.

$$\begin{aligned} \hat{p} &= \langle loadFromHDFS() \oplus normalization(all) \oplus kmeansModeling(k) \oplus \\ &\qquad saveToHDFS(model) \rangle \\ \hat{e} &= \langle \text{Hadoop}, [\text{Spark}, \text{Airflow}], \text{Hadoop}, \epsilon, [\text{Knox}, \text{Ranger}] \rangle \end{aligned} \tag{7.2}$$

We note that $S_V = \epsilon$ means that the ecosystem service for visualization is empty.

Table 7.1: Running example definition: tasks, template Π and instance I.

(a) Tasks in p and $\hat{p}$.

	t	$\hat{t}$
T_I	$t_1 = fileSystem$	$\hat{t}_1 = loadFromHDFS()$
T_P	$t_2 = normalization$	$\hat{t}_2 = normalization(all)$
T_A	$t_3 = modelling(clustering)$	$\hat{t}_3 = kmeansModeling(k)$
T_V	$t_4 = fileSystem$	$\hat{t}_4 = saveToHDFS(model)$

(b) Services in e and $\hat{e}$.

	s	$\hat{s}$
S_I	$s_1 = LoadFilesystem$	$\hat{s}_1 = Hadoop$
S_C	$s_2 = BatchProcessing$	$\hat{s}_2 = Spark$
S_C	$s_3 = Orchestration$	$\hat{s}_3 = Airflow$
S_S	$s_4 = StoreFilesystem$	$\hat{s}_1 = Hadoop$
S_V	$s_5 = \epsilon$	
S_E	$s_6 = AC: \ Authentication$	$\hat{s}_4 = Knox$
S_E	$s_6 = AC: \ Authorization$	$\hat{s}_5 = Ranger$

(c) Definitions of the pipeline template (7.3) and instance (7.4).

$$\Pi = \langle p, e \rangle \text{ with } \begin{array}{l} p = \langle t_1 \oplus t_2 \oplus t_3 \oplus t_4 \rangle \\ e = \langle s_1, [s_2, s_3], s_4, s_5, s_6 \rangle \end{array} \tag{7.3}$$

$$I = \langle \hat{p}, \hat{e} \rangle \text{ with } \begin{array}{l} \hat{p} = \langle \hat{t}_1 \oplus \hat{t}_2 \oplus \hat{t}_3 \oplus \hat{t}_4 \rangle \\ \hat{e} = \langle \hat{s}_1, [\hat{s}_2, \hat{s}_3], \hat{s}_1, \epsilon, [\hat{s}_4, \hat{s}_5] \rangle \end{array} \tag{7.4}$$

Table 7.1 summarizes the Example 7.3.1. The pipeline instance $\hat{p}$ defines an ingestion task (T_I) to ingest batches of data from HDFS ($\hat{t}_1 = loadFromHDFS()$) using the Hadoop service $\hat{s}_1$, performs a preparation task (T_P) normalizing all the data fields ($\hat{t}_2 = normalization(all)$) to fit into a modelling task (T_C) for k-means model creation ($\hat{t}_3 = kmeansModeling(k)$) based on Spark service $\hat{s}_2$.

The model is then saved on HDFS ($\hat{t}_4 = saveToHDFS(model)$) using the

Hadoop service $\hat{s}_1$ for later usage. The entire pipeline is orchestrated with Airflow service $\hat{s}_3$. To execute such pipeline an access control is requested using *Knox* (authentication) and *Ranger* (authorization), $\hat{s}_4$ and $\hat{s}_5$ respectively.

7.4 Assurance Evaluation Methodology for Big Data Analytics Pipeline

We propose a methodology for assurance evaluation of Big Data analytics pipeline that implements our assurance process in Section 7.1 and targets the analytics pipeline modelling in Section 7.2. More specifically, it is based on three sequential steps: i) *template annotation* that annotates the Big Data analytics pipeline Template (template in the following) with generic requirements, ii) *instance annotation* that annotates the Big Data analytics pipeline instance (instance in the following) with specific requirements, and iii) *assurance evaluation* that evaluates the overall assurance according to the assurance process in Section 7.1.

7.4.1 Template annotation

The template annotation is a process that annotates template $\Pi = \langle p, e \rangle$ with a set of non-functional requirements $r \in \mathcal{R}$ taken from two vocabularies $\mathcal{R}_p, \mathcal{R}_e \subset \mathcal{R}$, such that $\mathcal{R}_p \cup \mathcal{R}_e = \mathcal{R}$. We define two labelling functions i) $\lambda : T \rightarrow \mathcal{R}_p$ that associates each tasks $t \in T$ in pipeline p with a set of non-functional requirements $r \in \mathcal{R}_p$, ii) a labelling function $\gamma : S \rightarrow \mathcal{R}_e$ that associates each services $s \in S$ in the pipeline ecosystem e with a set of non-functional requirements $r \in \mathcal{R}_e$. We formally define an annotated Big Data analytics pipeline template as follows.

Definition 7.4.1 ($\Pi^{\lambda,\gamma}$)**.** An annotated Big Data analytics pipeline template is defined as $\Pi^{\lambda,\gamma}$ where λ and γ are two labelling functions such that: i) λ assigns labels $\lambda(t_i)$ corresponding to pipeline requirements in $\mathcal{R}_p$ to be satisfied by task t_i; ii) γ assigns a label $\gamma(s_i)$ corresponding to service requirements in $\mathcal{R}_e$ to be satisfied by service s_i.

We note that labelling function $\lambda(\gamma, resp.)$ can assign label $\lambda(p)(\gamma(e), resp.)$ corresponding to pipeline (ecosystem, resp.) requirements in $\mathcal{R}_p(\mathcal{R}_e, resp.)$ to be satisfied by pipeline p (ecosystem e). We also note $\lambda(p)$ refers to requirements on the pipeline structure (e.g., the sequence of tasks), while $\gamma(e)$ refers to the environment where the services are deployed (e.g., the operating system or the service container).

Example 7.4.1 (Annotated template)**.** Following Example 7.3.1, a client specifies requirement *Confidentiality in transit* in $\mathcal{R}_p$ at template level. A complete annotation for requirement *Confidentiality* is discussed in Section 11.4. Template Π can be annotated as follows:

- All tasks $(*)$ in $p \in \Pi$ are annotated with $\lambda(*) =$ *Confidentiality in transit*, and the entire pipeline $p \in \Pi$ is annotated with $\lambda(p) =$ *Pipeline integrity*
- All services $(*)$ in $e \in \Pi$ are annotated with $\gamma(*) =$ *Confidentiality in transit*.

We note that tasks, services, the pipeline and the service ecosystem can be annotated with zero or more requirements according to Definition 7.4.1. For instance, $\gamma(e)$ was empty in Example 7.4.1.

7.4.2 Instance annotation

The instance annotation is a process that annotates instance $I = \langle \hat{p}, \hat{e} \rangle$ with a set of concrete requirements $\hat{r} \in \hat{\mathcal{R}}$ taken from two vocabularies $\mathcal{R}_{\hat{p}}, \mathcal{R}_{\hat{e}} \subset \hat{\mathcal{R}}$, where $\hat{\mathcal{R}} = \mathcal{R}_{\hat{p}} \cup \mathcal{R}_{\hat{e}}$ is a specialization of $\mathcal{R} = \mathcal{R}_p \cup \mathcal{R}_e$.

We define two labelling functions i) $\theta : \hat{T} \rightarrow \mathcal{R}_{\hat{p}}$ that associates each invocation of $\hat{t} \in \hat{T}$ in the pipeline $\hat{p}$ with a set of non-functional requirements $\hat{r} \in \mathcal{R}_{\hat{p}}$, ii) labelling function $\psi : \hat{S} \rightarrow \mathcal{R}_{\hat{e}}$ that associates each invocation of $\hat{s} \in \hat{S}$ in the ecosystem model $\hat{e}$ with a set of non-functional requirements $\hat{r} \in \mathcal{R}_{\hat{e}}$. Similarly to template annotation, we formally define an annotated Big Data analytics pipeline instance as follows.

Definition 7.4.2 ($I^{\theta,\psi}$)**.** An annotated Big Data analytics pipeline instance is defined as $I^{\theta,\psi}$ where θ and ψ are two labelling functions such that: i) θ assigns a label $\theta(\hat{t}_i)$ corresponding to pipeline requirements in $\mathcal{R}_{\hat{p}}$ to be satisfied

by task $\hat{t}_i$; ii) ψ assigns a label $\psi(\hat{s}_i)$ corresponding to the service requirements in $\mathcal{R}_{\hat{e}}$ to be satisfied by service $\hat{s}_i$.

We note that labelling function $\theta(\psi, resp.)$ can assign label $\lambda(\hat{p})(\gamma(\hat{e}), resp.)$ corresponding to pipeline (ecosystem, resp.) requirements in $\mathcal{R}_{\hat{p}}(\mathcal{R}_{\hat{e}}, resp.)$ to be satisfied by pipeline $\hat{p}$ (ecosystem $\hat{e}$).

An annotated instance $I^{\theta,\psi}$ is obtained by an annotated template $\Pi^{\lambda,\gamma}$ according to transformation function $\xrightarrow{R}$ as follows.

Definition 7.4.3 ($\xrightarrow{R}$)**.** $\xrightarrow{R}$ is a transformation function that receives as input the annotated template $\Pi^{\lambda,\gamma}$ and the pipeline instance I, and generates as output an annotated pipeline instance $I^{\theta,\psi}$, where: i) $\Pi \xrightarrow{I} I$ and ii) the generic pipeline requirements annotated with $\lambda \in \Pi$ are specialized into instance-specific requirements annotated with $\theta \in I$ and the generic ecosystem requirements annotated with $\gamma \in \Pi$ are specialized into instance-specific requirements annotated with $\psi \in I$.

We note $\xrightarrow{R}$ can be either a manual or an automatic transformation. Any generic requirements $r \in \mathcal{R}$ can be specialized in one or more instance-specific requirement $\hat{r} \in \hat{\mathcal{R}}$, thus leading to one or more annotated pipeline instances $I^{\theta,\psi}$. For conciseness, we consider the instance annotation function $\xrightarrow{R}$ as given and carried out by the service provider on their instances a priori, according to the available templates. In case of multiple possible instantiations, the service provider selects the best one according to its deployment strategy.

Example 7.4.2 (Annotated Instance)**.** Let us consider the annotated template $\Pi^{\lambda,\gamma}$ in Example 7.4.1. One possible annotated analytics pipeline instance $I^{\theta,\psi}$ can be as follows.

- The pipeline instance $\hat{p} \in I$ is annotated with
- The ecosystem instance $\hat{e} \in I$ is annotated with $\psi(\hat{s}_1) =$ *Encrypted HDFS* and *Inter-node communication security*, $\psi(\hat{s}_2) =$ *Inter-node communication security*, $\psi(\hat{s}_3) =$ *Orchestrator Confidentiality* and $\psi(\hat{s}_4) =$ *Communication channel security*.

We note that tasks $\hat{t}_1 = loadFromHDFS()$ and $\hat{t}_4 = saveToHDFS(model)$ are not affected by the request of confidentiality in transit expressed at template level in Example 7.4.1. This requirement is in fact considered by the ecosystem level only in the specific instance in Example 7.4.2 and addressed by $\hat{s}_1 = Hadoop$. In other words, annotation $\lambda(t_1) =$ *Confidentiality in transit* is transformed by the instance annotation function $\xrightarrow{R}$ to $\theta(\hat{t}_1) = \varnothing$ and $\theta(\hat{t}_4) = \varnothing$. We also note that confidentiality in transit is instantiated as $\theta(*) =$ *Avoid connection to external services* and $\theta(*) =$ *Avoid use of vulnerable libraries* for all tasks $(*)$ in the pipeline. We finally note that $\theta(\hat{p}) =$ *Pipeline Integrity* refers to the verification of the pipeline structure and is associated with the corresponding requirement at service level $\psi(\hat{s}_3) =$ *Orchestrator Confidentiality*, while $\psi(\hat{s}_4) =$ *Communication channel security* refers to the authentication channel used by $\hat{s}_4$ (i.e., Knox).

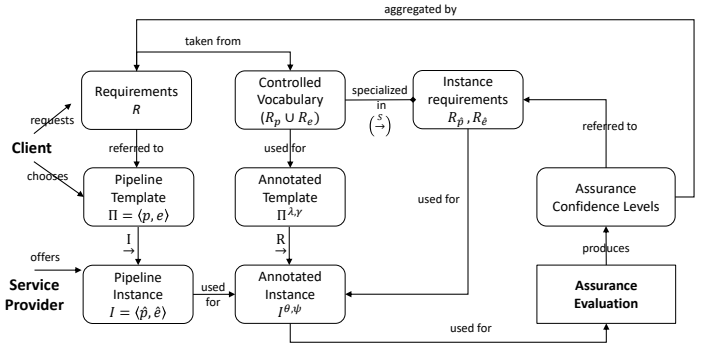


Figure 7.3: The Assurance methodology for Big Data Analytics Pipeline.

7.4.3 Assurance evaluation

The assurance evaluation executes one or more assurance processes A in Definition 7.1.3 (using the architecture in Figure 7.2) on $I^{\theta,\psi}$ such that $\Pi^{\lambda,\gamma} \xrightarrow{R} I^{\theta,\psi}$. It can be formally defined as follows.

Definition 7.4.4 (Assurance Evaluation)**.** Let us consider template $\Pi^{\lambda,\gamma}$ and corresponding instance $I^{\theta,\psi}$, such that $\Pi^{\lambda,\gamma} \xrightarrow{R} I^{\theta,\psi}$. For each pair $\langle \tau, r \rangle \in \Pi^{\lambda,\gamma}$, where τ is a task t, a service s, a pipeline p, or an ecosystem

e annotated with requirement $r \in \mathcal{R}$, the assurance evaluation first retrieves the corresponding set of pairs $\langle \hat{\tau}_i, \hat{r}_i \rangle \in I^{\theta,\psi}$, where $\hat{\tau}_i$ is an executable task $\hat{t}$, a deployed service $\hat{s}$, a concrete pipeline p, or a concrete ecosystem e annotated with concrete requirement $\hat{r} \in \mathcal{R}$. It then computes the overall assurance confidence level $L_{\hat{\tau}_i,\hat{r}_i} = \min(A_{\hat{\tau}_i,\hat{r}_i})$.

We note that the assurance evaluation calculates the assurance confidence level of each $\langle \tau, r \rangle$ expressed by the client on template $\Pi^{\lambda,\gamma}$. To this aim, it aggregates the results of multiple assurance processes A executed on the corresponding $\langle \hat{\tau}_i, \hat{r}_i \rangle \in I^{\theta,\psi}$, and computes the overall assurance confidence level $L_{\tau,r}$ as the minimum assurance confidence level. More advanced approaches will be investigated in our future work. Figure 7.3 shows a complete view of our methodology applied to the scenario in Section 7.3. We note that, the Service Provider can instantiate a given template ($\xrightarrow{I}$) or search for a suitable valid instance following our approaches in [79] and [79] respectively.

Following the assurance process A in Section 7.1.1, the assessment is carried out via a specific set of assessment probes P_i suitable for the target τ and the specific requirements $\hat{r}$ annotated on the corresponding targets by θ or ψ. Probes are parametric and use the hooks offered by τ to carry out the inspections. To accomplish the heterogeneity of assurance verification, we define different assessment probes (see Table 7.2): i) *pipeline probes* focusing on the verification of pipeline tasks and orchestration, ii) *service probes* focusing on the verification of the Big Data services executing the pipelines, iii) *ecosystem probes* focused on the lower layers involving infrastructure behind the Big Data ecosystem.

Example 7.4.3 (Probes)**.** Let us consider a given task $\hat{t}_2 =$ *normalization*(*all*) annotated by $\theta(\hat{t}_2)$ with a requirement $\hat{r}_1^{\theta} =$ *Avoid vulnerable libraries* and a given ecosystem service $\hat{s}_4 = Knox$ annotated by $\psi(\hat{s}_4)$ with a requirement $\hat{r}_1^{\psi} =$ *Authentication-enabled*. A Code Vulnerability Check Probe P_i can be used to check whether task $\hat{t}_2$ relies on vulnerable libraries reducing the confidentiality ($\hat{r}_1^{\theta}$). A Configuration Check Probe P_j can be used to check whether authentication is requested by service $\hat{s}_4$ to trigger an analytics via a well-configured Apache Knox service ($\hat{r}_1^{\psi}$). A Testing Probe P_k can be used to check whether authentication is working, while trying to execute the pipeline ($\hat{r}_1^{\psi}$).

Probe Name	Description
Task probes	
Code Inspection	search instructions/pattern
Dependency Check	find vulnerable dependencies
Code Vulnerability Check	search vulnerable code
Lineage	verify sequence of actions using logs
Pipeline probes	
Parameters Check	check tasks' actual parameters
Orchestration Check	check the workflow structure
Service probes	
Vulnerability Check	search for vulnerability
Configuration Check	parse and verify configuration
Ecosystem probes	
Infrastructure	targets lower layers such as OS (see [50])
General purposes probes	
Testing	perform specific test cases on a target
Monitoring	monitor a target ore a time frame

Table 7.2: Types of assessment probes.

The probes are structured according to our probe paradigm detailed in [81] where the probe receives parameters for connecting to the target (e.g., API hooks) and instruction on how to match the information retrieved by the target (e.g., test cases and expected outputs). It performs i) connection, ii) information retrieval and iii) matching against expectations. The information retrieved and the matching outcomes constitute the probe evidence.

Evidence EV retrieved by the probes is evaluated (see Definition 7.1.2) and used to compute the assurance level $A_{\tau,\hat{r}}$ (see Definition 7.1.3) of the target τ with respect to the given requirement $\hat{r}$.

Example 7.4.4 (Assurance Confidence Levels)**.** Let us consider Example 7.4.3 and the following evidence collected by corresponding assessment probes. Probe $P_1(\hat{t}_2, \hat{r}_1^{\theta})$ produces evidence $EV_1 = [CVE-2018-1334, 4.7]$, referring

to a weakly-vulnerable (severity 4.7) version of spark library used in normalization code; probe $P_j(\hat{s}_4, \hat{r}_1^{\psi})$ produces evidence $EV_2 = [enabled = true]$, referring to a correctly configured Apache Knox enabling authentication for pipeline execution; probe $P_k(\hat{s}_4, \hat{r}_1^{\psi})$ produces evidence $EV_3 = [(\langle valid - user, valid - pwd \rangle, allow), (\langle valid - user, wrong - pwd \rangle, deny), \ldots]$ referring to a set of positive testing results concerning authentication verification for the pipeline execution. Two evaluation processes analyse the collected evidence to check the support of the requirements $\hat{r}_1^{\theta}$ and $\hat{r}_1^{\psi}$. In this example, $E_1(EV_1, \hat{r}_1^{\theta}) = 0.5$, due to the presence of a vulnerable library with a low severity score (i.e., Common Vulnerability Scoring System (CVSS)), and $E_2(EV_2 \cup EV_3, \hat{r}_1^{\psi}) = 1$, due to the positive checks on the Knox authentication. The corresponding assurance confidence levels (Definition 7.1.3) are $A_{\hat{t}_2, \hat{r}_1^{\theta}} = 0.5$ and $A_{\hat{s}_4, \hat{r}_1^{\psi}} = 1$, respectively.

We not that, the outcome of the assurance process is a set of assurance confidence levels on the annotated pipeline instance $I^{\theta,\psi}$. Following Definition 7.4.4, the assurance confidence levels are aggregated in the overall assurance confidence level. Considering the assurance levels in Example 7.4.4, the overall assurance confidence level is equal to 0.5, the minimum between the retrieved assurance confidence levels.

We note that our assurance methodology immediately react against events impacting on the assurance evaluations, retrieving new evidence and computing the new assurance level. Events include new discovered vulnerabilities, new versions of ecosystem services, updates on the task or pipeline code, or new more effective probe deployed on the probe repository. In case the assurance confidence level is no more satisfactory, the instance can be replaced with a new one if the requested changes are major. The instance can be adapted in terms of annotations if the requested changes are minor (e.g., new service versions, new probes or new vulnerabilities). While we consider adaptation for our future work, we provide a preliminary discussion in the experiments in Section 11.4.

7.5 Discussion

The increasing trend towards Big Data process outsourcing and the lack of Big Data trustworthiness and transparency represent the major barriers against high-quality Big Data computations. Users, in fact, (fully) anonymise data and results to reduce their liability in case of data breach. The assurance verification in Section 7.1 permits to unleash the full power of Big Data, fostering its adoption in critical scenarios where sensitive data are used and promoting the "*Big Data-as-a-Service*" paradigm where trustworthiness is of paramount importance. It is important to remark that Big Data transparency is also mandatory when anonymized data are used, for instance, to ensure that the level of anonymity reached with anonymized data is not violated during processing (i.e., within the pipeline) or while presenting the final results due to correlation with different data sources [82].

Chapter 8

Assurance Aware Deployment infrastructures

Deployment solutions manage the life cycle of applications, networks and resources within large and heterogeneous environments. In order to achieve this, multiple levels of abstractions are adopted to manage the underlying hardware and software. This approach delegates the responsibility of defining the expected outcome to the user, while the system takes care of (most) implementation details. As previously described in Section 2.3, resources, deployments and configurations are represented in code following common and publicly available resource schema. These configurations enable the definition of SLOs for several features, such as application scheduling, network priority, storage availability and retention, and performance budgets. Application scheduling is affected by these SLOs, introducing hard or soft limits that must be met by the chosen deployment solution provided by the deployment engine. Most SLOs focus on performance and resource usage of applications, and are therefore neglecting other interesting aspects of NFPs, such as security, privacy and automation.

We aim to provide certified NFPs in application deployments in accordance with user requirements. To accomplish this objective, we must: i) expand the resource system to include the ability to specify NFPs ii) implement

NFPs assurance in the deployed applications and deployment targets, creating contracts to verify the requested NFPs and suitable Assurance Agents (AAs) iii) the assurance results must be integrated into the application scheduling system. Furthermore, when deploying applications, the hosting provider may offer services that can be utilised to meet he SLAs, such as authentication and permission management solutions. Alternatively, supplementary applications may be required to provide these services.

In this scenario, continuous verification of the NFPs enables the automation and adaptive deployment of applications in response to changes in the deployment infrastructure, thereby ensuring the dependability of both the platform and the applications deployed on top of it. This is particularly significant in an E2C environment due to its heterogeneous and distributed nature and reliance on communication between data centres.

8.1 Continuous Non-Functional Property assurance in deployment infrastructure

Continuous verification of NFP is crucial for automating the deployment lifecycle in Cloud and E2C infrastructures. The user's expectation is to provide a recipe for the intended deployment, including software packages, configurations and requirements, and to receive a fully operational system. The recipe is received by the service provider, which schedules the deployment using the data centre resources. The scheduling process attempts to solve an optimisation problem, looking for viable configurations that meet the user's requirements and minimise some combination of parameters, usually cost, power consumption and machine under-utilisation [83, 84]. Generally, only a small part of NFPs are considered for these requirements, falling under the umbrella of the resources performance and availability families, and only more recently Cloud providers started to adopt security and privacy NFPs. The scheduling system employs a simplified continuous assurance process for assess the system's condition and determine whether the user requirements can be fulfilled. Specifically, it utilizes system metrics to gauge the available resources and the status of the deployment target, calculating general availability and behaviour of the deployed applications.

8.1.1 Extending deployment infrastructures

Scheduling systems have the potential to expand their coverage to encompass more complex NFPs as well as multiple layers of deployment. The system's state can be measured using transparent monitoring techniques or by ad hoc probes and made available through monitoring endpoints, as discussed in Section 3.2.3. This allows to have a more complete picture of the lower layers of the deployment target, thus allowing us to infer stronger properties over the system.

The present definitions of SLAs primarily concentrate on the performance-based NFPs and are restricted to soft and hard thresholds. Consequently, they lack the ability to articulate more sophisticated necessities, like data privacy and confidentiality, network security policies, and availability. To address these limitations, we have introduced the concept of function-based contracts as SLAs. Advanced SLAs are formalised through contracts, granting users a formal and transparent method to ensure requirements feasibility and NFP guarantees that works across multiple providers, addressing Gap G_2 *Strong reliance on infrastructure*. These contracts depend on a unified set of metrics for collecting data on the system's state and incorporate arbitrarily complex logic statements to describe how the intended property should be verified. This results in a solution that is both more efficient and accurate. Both contracts and metrics are specified and released by a CAs and can be adopted by Cloud providers and users as a common basis for NFP verification. This enables citing of a specific contract directly from the deployment definition, achieving greater transparency and adherence to standardisation efforts.

8.2 Introduction

According to IDC's Data Age 2025 white-paper, the worldwide data will grow to 175 zettabytes by 2025. This growth will be driven by the increment in speed (average speed of 110 Mb/s by 2023 according to Ericsson[1]) and in a number of connected data sources (almost 30 billion of IP devices and half of

[1]Mobile data traffic outlook available at `https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts`

them will be M2M devices by 2023 according to CISCO[2]). Ericsson predicts to reach 288 exabytes per month of traffic exchanged in the mobile network by 2027, most of them due to the adoption of high-speed 5G connections.

The current Cloud architectures cannot effectively handle such data growth across the network, becoming a blocking factor to unleashing the real potential of data eager technologies such as IoT platforms and AI. Edge and Fog computing constitute part of the missing link between Cloud and the data sources, moving computation and storing capabilities as well as communication and decisions closer to the network periphery, trying to address the data handling gap. While Fog is meant to work with Cloud involving a potentially extensive number of layers in between, Edge typically has fewer layers and not always requires a Cloud central point. In addition, Fog potentially includes the pervasive handling of resources such as networking and storage. In this paper, we focused on Edge architectures as means to deploy a data-intensive pipeline that also requires Cloud facilities to handle the analytics executions and present the final results. More specifically we focused on critical pipeline acquiring and processing sensitive data and requiring advanced security and privacy protection mechanisms [85]. Such protections have to be selectively activated depending on the user and context where the pipeline is executed. In order to build the continuum, in addition to multiple layers of on-premises Edge nodes, we also consider MEC nodes offered by telco operators using the 5G core network. The 5G core network is a large composition of services with complex interactions and processes, including the self-organization of network functionalities, collaboration with other networks and management of user data and traffic. 5G is considered an enabling technology for integrating resource-intensive E2C computing with IoT and mobile computing, bridging the gap between powerful data centres and low-power devices. With the increment of 5G adoption, there will be an increment in the need for micro data centre capillary distribution as well as 5G facilities to support MEC. These facilities will be the natural landing Edge platform for acquisition and preprocessing tasks of next-generation data-intensive pipelines requiring advanced QoS such as low latency security and privacy [86]. In this scenario of Edge continuum includes telco operator facilities being capable of orchestrating and deploying data-intensive pipelines while maintaining specific QoS is of paramount importance. In this paper, we present an Edge continuum orchestrator builder that is capable of generating executable orchestrations

[2]White-paper available at `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`

for the different continuum layers (including 5G Edge) guaranteeing a specific QoS across the continuum.

8.3 Reference Scenario

Our reference scenario considers a client that wants to deploy its data-intensive pipeline on a given 5G-enabled Edge continuum architecture made of i) Edge nodes on-premises ii) mobile Edge nodes in the core network of a given telco operator and iii) Cloud nodes. The client also expresses specific QoS requirements in terms of security/privacy and performance to be met by the deployment. The pipeline is made of tasks that can be dynamically selected (a priori) and orchestrated to be deployed in the 5G-enabled Edge continuum with a specific analytics goal. More specifically the pipelines of our scenario are focused on critical data-intensive analytics including ingestion and complex privacy-preserving data preprocessing activities that need to be effectively orchestrated in the continuum in order to avoid as much as possible traffic congestion.

8.3.1 Requirements

Considering the above scenario a number of crucial requirements, for pipeline deployment and orchestration in 5G-enabled Edge continuum, has to be considered as follows:

- [**R1**] **Interoperability**. It refers to the possibility of orchestrating tasks at multiple layers of the Edge continuum.
- [**R2**] **Transparency/auditability**. It refers to the need of inspecting the status of each task and of the entire orchestration with regards to functional and QoS requests.
- [**R3**] **Reliability, Availability, and Serviceability (RAS)**. This triad is focused on ensuring reliable and available application pipeline in the continuum. More in details reliable and available deployment and orchestration. Serviceability refers to the need of ensuring the

maintainability of the orchestration meaning the possibility to modify the orchestration dynamically if needed. This requirement is also clearly related to R2 requiring the ability to inspect and re-deploy orchestrations if requested.

- **[R4] Openess**. In our scenario openness can be seen as the composability of tasks in the pipeline and of orchestrations in the continuum via open standards e.g., Fast Health Interoperable Resources (FHIR). It is related also to R2 and R3 (Serviceability) requirements.

- **[R5] Agility**. It refers to the ability to transform a great volume of data into actionable insights. More specifically being capable of making decisions as soon as the data can lead to meaningful context and as close as possible to the data sources. It can be also seen as the need to avoid network dependencies.

- **[R6] Programmability**. It refers to the dynamicity of deployment and the adaptability of the orchestrations. In our scenario programmability refers primarily to the coding of adaptive orchestrations ensuring optimal deployment addressing QoS needs.

- **[R7] Security**. Security is a quite transversal requirement for distributed systems. In the context of our scenario it assumes a more holistic fashion including the security of Edge nodes, 5G core, communications, deployed tasks and deployment and orchestration mechanisms.

- **[R8] Privacy**. Similarly to security, privacy in distributed systems requires addressing a number of challenges. In our scenario, we consider the privacy of the data potentially spreading in different forms across the continuum. For instance it means that the Edge nodes have to provide i) contextual integrity and isolation, ii) control access and aggregation of privacy-sensitive data before it leaves the Edge, depending also on the access right of the actors executing the pipeline [71], iii) segregation capabilities.

We note that most of the above requirements share the same generic needs expressed by the Fog pillars of OpenFog [87] but they are declined for critical pipeline orchestration in 5G-enabled Edge continuum.

8.4 Building Blocks

In the following, we present our building blocks. More specifically we describe the notion of Edge computing, previously described in Section 4.1, with specific reference to the peculiarities that are of interest for our Big Data pipeline scenario, shown in Section 8.3. We then provide our notion of a data-intensive pipeline, where data is collected, processed, ingested and analysed across the Edge continuum. Concluding we describe the 5G architecture to show the basic components enabling the 5G MEC.

8.4.1 Data-intensive Pipeline

A data-intensive pipeline is a data acquisition and analytics procedure adopted for instance in the context of massive IoT and for the acquisition of distributed streaming data sources. Nowadays these types of acquisition procedures are becoming central for a number of applications including the AI-empowered services, where data are fundamental to generating and updating AI models. In this paper, we consider the data-intensive pipeline as made of tasks that can be deployed in a distributed fashion and orchestrated in order to fulfil the acquisition goal. The pipeline tasks include analytics tasks such as clustering, acquisition oriented procedures like data source connectors, or data gathering/digitalisation, but also data preprocessing procedures aimed for instance to aggregate data, perform space reduction, anonymisation, sparsity reduction, early fusion, latent data space transformations, to name but a few. In the framework of the Edge continuum, considering the impact of data flows in terms of networking saturation and therefore latency, the effectiveness of the acquisition pipeline in filtering and transferring the minimum set of meaningful data only, is fundamental. On the other hand, the pipelines cannot be easily generalized and they are applications dependent. Therefore they have to be deployed dynamically to address different application needs. The pipeline orchestrator should be capable of selecting suitable tasks and deploying them in suitable edge continuum locations in order to fulfil a specific functional goal and QoS requirements (e.g., latency, privacy).

8.5 Orchestration Builder

In general, the process of orchestration refers both to services, workload, and resources. In the Cloud context, orchestration automates the coordination and management of distributed services, computing resources and middleware. It is also at the basis of the Cloud scaling, elasticity and the self-service model. Nowadays there is an ongoing research effort in the direction of including QoS compliance in orchestration where also NFPs considered [50,51]. However, the current approaches are not suitable for addressing the level of dynamicity and heterogeneity introduced by the 5G-enabled Edge continuum and by the needs of new-generation data-intensive pipelines. An advanced orchestration approach is needed to support the dynamic deployment of services in such an Edge continuum, where applications can be geo-distributed and storage decentralized posing new challenges in terms of integrity and reliability. In addition, in this paper, we considered 5G-enabled Edge scenarios, making the orchestration even more complex requiring to have a more deep understanding of the peculiarities of the landing Edge node. Currently, one of the most adopted approaches to distribute applications addressing the heterogeneity of the landing platform is via containerization [88], however, there is still the need to detail topology and orchestration planning and it is quite complex to satisfy QoS requirements. To address these limitations, orchestration standards have been defined to describe the expected configuration of a deployment, while leaving the task of deciding the concrete setup to an orchestrator. The concept of containers and virtualisation is largely used to address distributed systems needs even in telco sector which is becoming more and more softwarised. For instance, the 5G architecture (see Section 4.2) adopts NFV to address the need for flexibility and elasticity but requires a more complex orchestration approach leading to the ETSI MANO framework adopted by the 5G Network.

In this paper, we propose an orchestrator builder capable of generating prescriptive orchestrations for the different continuum layers addressing their specificity and linking them together to build the entire meta orchestration. A meta orchestration is a Directed Acyclic Graph (DAG) made of tasks defined as follows.

Definition 8.5.1 (Meta Orchestration)**.** A meta orchestration $O(V, E)$ is a DAG made of a vertex $v_i \in V$ one of each task t_i building the orchestration.

It includes special vertices as follows: i) root vertex v_r, ii) two flow splitting vertices $v_{\otimes}$, $v_{\oplus}$ to structure alternative ($\otimes$) and parallel ($\oplus$) executions of tasks. Splitting vertices are used to begin the splitting and merge the results for tasks that need to be executed in an alternative or parallel fashion. E is the set of edges e_i connecting the vertices forming the DAG.

The meta orchestration is annotated with QoS requirements $r_i \in R$ (e.g., latency, privacy) and constraints $c_i \in C$ (e.g., deployment on Cloud) forming an annotated meta orchestration $O^{R,C}$. More specifically, both vertices and edges in the meta orchestration $O(V, E)$ can be annotated with specific r_i or c_i. For instance a security requirements $r_i = \textit{Confidentiality}$ can be associated to an edge (i.e., communication channel) e_j denoted as $e_j^{r_i}$ and a constraint $c_i = \textit{On-premises}$ can be associated to a vertex (i.e., a task of the pipeline) v_j denoted as $v_j^{c_i}$

Definition 8.5.2 (Orchestration Builder)**.** Orchestration builder is a function of the form B: $(O^{R,C}, A) \rightarrow \hat{O}_i \cdots \hat{O}_n$. It takes in input the annotated meta orchestration $O^{R,C}$, and the description of the landing continuum architecture A, and produces as output one or more orchestrations $\hat{O}_i$ partitioning the original meta orchestration $O^{R,C}$. Each orchestration $\hat{O}_i$ is generated in order to be suitable for the deployment on the landing Edge layers and to address both QoS and constraints expressed in $O^{R,C}$. The orchestrations $\hat{O}_i$ are merged by special input and output vertices ($v_{\Leftarrow}$ and $v_{\Rightarrow}$ respectively) in order to form a complete pipeline where: i) $V \in O \subset \cup_{i=1\cdots n}(\hat{V}_i \in \hat{O}_i)$; ii) $\forall$ $e \in E$ between two vertices v_i and v_j either $\exists$ $\hat{e} \in \hat{E}$ between $\hat{v}_i$ and $\hat{v}_j$ or $\exists$ $\hat{e}_i$ between $\hat{v}_i$ and $v_{\Rightarrow}$ and $\exists$ $\hat{e}_j$ between $v_{\Rightarrow}$ and $v_{\Leftarrow}$ and $\exists$ $\hat{e}_k$ between $v_{\Leftarrow}$ and $\hat{v}_j$; iii) vertices $v_{\Leftarrow}$ and $v_{\Rightarrow}$ are replaced with ad hoc tasks capable of setting up a data communication channel using the peculiarities of the landing edge node; iv) vertices $v_{\otimes}$, $v_{\oplus}$ are replaced with tasks capable of splitting the execution flows potentially to different node instances; v) normal vertices are replaced with tasks implementing the specific functionality requested.

Figure 8.1 shows the deployment of a given meta orchestration $O(V, E)$ on our reference architecture made of i) on-premises Edge nodes, ii) 5G Edge nodes and iii) a central Cloud node. We note that the meta orchestration $O(V, E)$ model a pipeline made of six different tasks ($t_1 \cdots t_6$) to be executed consecutively. We also note that tasks t_1, t_2 and t_3 are selected by the orchestration builder to be part of the Edge orchestration, t_4 is the only

task selected for 5G Edge orchestration while t_5 and t_6 are selected to be orchestrated on the Cloud. The meta orchestration is annotated with QoS r_1 on the communication edges and constraint c_1 on the tasks requiring on-premises deployment. The orchestrator builder replaced the input and output vertices ($v_{\Rightarrow}$ and $v_{\Leftarrow}$) with communication-oriented tasks $t_3^{\Rightarrow}$, $t_4^{\Leftarrow}$, $t_4^{\Rightarrow}$ and $t_5^{\Leftarrow}$ in order to establish a communication flow between the orchestrations.

Figure 8.1 also shows that Edge and Cloud nodes share the same orchestration technology based on Kubernetes, while the 5G Edge node uses the MEAO and NFV-based orchestration of MEC services. We note that our reference architecture allows deployments on multiple Edge and 5G Edge nodes while considering for the sake of simplicity a single centralized Cloud node.

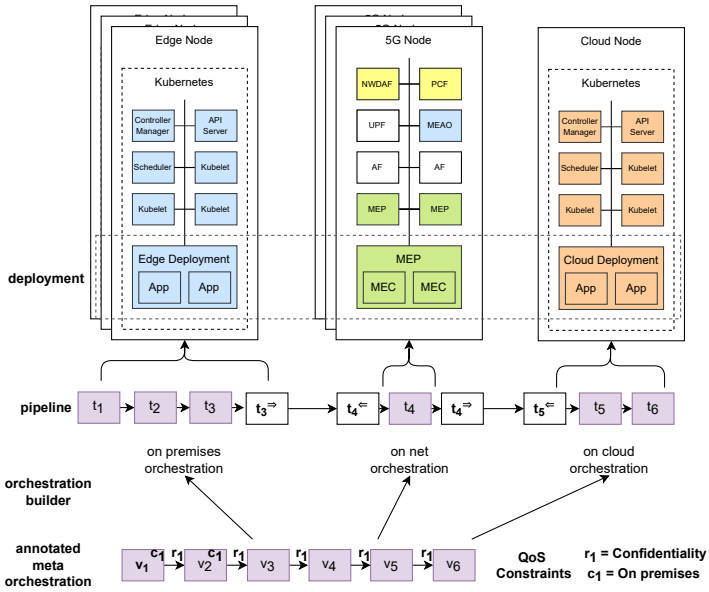


Figure 8.1: Our methodology applied to a given meta orchestration deployed on a given 5G-enabled Edge continuum architecture.

8.5.1 QoS on Continuum Edge

In addition to the functionality, our orchestration builders have to generate orchestrations that are compliant with specific QoS requirements. Our orchestration builder is capable of considering QoS requests while building the orchestrations considering also the different types of nodes of the continuum. Based on the deployment technologies we can differentiate between i) on-premises Edge and Cloud nodes, ii) 5G Edge nodes. For the sake of conciseness let us consider security/privacy QoS related only.

Security aspects of on-premises Edge and Cloud Orchestration should use the security/privacy mechanisms provided by the node and add additional ones in case of necessity within the pipeline. For instance edge between v_4 and v_5 in Figure 8.1 is subject to r_i. It means that the communication task $t_3^{\Rrightarrow}$ and $t_4^{\Lleftarrow}$ generated by the orchestration builder have to provide a confidential communication channel. If the landing node is not capable of providing the means to establish such a secure communication channel, the data have to be encrypted prior to the be sent by $t_3^{\Rrightarrow}$ and decrypted by $t_4^{\Lleftarrow}$ prior to being passed to the receiver.

We note that at this level the orchestration should be i) aware of the security/privacy features of the landing node and ii) capable of enabling/using them on request, iii) capable of adding security/privacy mechanisms between different orchestrations.

Security aspects of Edge in 5G In the current 5GS architecture the use cases related to the application of user plane security are limited to the security policies towards the NG-RAN, based on the activation of the integrity protection and/or confidentiality in the air interface between the UE and the Base Station. These security policies can be part of the subscriber information stored in the UDR and retrieved by the UDM function or, configured locally. Since currently UP security policies are globally based, control by local policies in the SMF may seem a priori sufficient. However, a new approach is necessary to address QoS related to critical security features required by pipelines addressing mission-critical scenarios and requiring different and customized security levels. For instance, it can be needed to selectively activate

different integrity protections or specific security measures for specific users/pipelines or network slices. The orchestrator should be capable of configuring the available 5G policies enabling selective security/privacy features needed. Note that in case the telco offers already services with adequate security features, user-specific policies are not requested to be asked.

8.6 Walkthrough Example

In the following, we describe a specific vertical of the generic scenario in Section 8.3 aimed at supporting a wet lab implementing a data analytics pipeline to identify novel therapeutic targets and highlight pharmacological networks in humanized experimental models.

Let's start describing the generic pipeline description to be fed into our orchestration building. The initial task of the pipeline is a digitalisation task focused on nucleotide sequencing from both mouse and human mRNA extracted from tissue (task t_1). These data have to be preprocessed and cleaned before being further analysed (task t_2). The collected data have to be then processed in different ways by i) spectral detection (task t_3), ii) peptide search (task t_4), iii) peptide indexing (task t_5) and iv) protein inference (task t_6) focused on protein quantification. The results of this pipeline should be provided back to the wet lab with the highest possible throughput and the entire pipeline have to be secured for data confidentiality (task t_7). The meta orchestration modelling the above pipeline can be defined as follows $O = \langle v_1, v_2, v_{\oplus}, v_3, v_4, v_5, v_6, v_{\oplus}, v_7 \rangle$. Let's consider QoS requirements $r_1 =$ *Confidentiality* and $r_2 =$ *High throughput* and the constraint $c_1 =$ *on premises* annotated on the digitalisation vertex ($v_1^{c_1}$) of the meta orchestration modelling the fact that the acquisition is based on a physical machine (Nextgen 550 Illumina). Let's also consider as a landing continuum architecture the one in Figure 8.1.

Our orchestrator builder in Definition 8.5.2 generates four orchestrations made of executable tasks implementing the pipeline as depicted in Figure 8.2 addressing r_1, r_2 and c_1. More specifically the first orchestration is for the on-premises edge node, according to the constraint c_1, and is based on digitalisation task t_1 only. The second orchestration resides in the MEC and is based on cleaning task t_2 only. The third orchestration is made of a set of

parallel tasks (tasks t_3, t_4, t_5, t_6) that are deployed on multiple cloud nodes to support performance requirement r_1. The fourth orchestration resides again on the MEC and is based on visualization task t_7 only.

These orchestrations are merged by some communication-oriented tasks: i) $t_1^{\Leftarrow}$ to ingest data from the physical machine; ii) $t_1^{\Rightarrow}$ and $t_2^{\Leftarrow}$ to establish the communication between the Edge premises orchestration and the 5G MEC one; iii) $t_2^{\Rightarrow}$ and $t_{\oplus}^{\Leftarrow}$ to establish the communication between the 5G orchestration and the Cloud one where tasks (from t_3 to t_6) are executed in parallel; iv) $t_{\oplus}^{\Rightarrow}$ and $t_7^{\Leftarrow}$ to establish the communication providing back to the wet lab the results of the analysis via 5G; v) $t_7^{\Rightarrow}$ to provide the hook to access the results visualization. Such communication-oriented tasks are configured to provide secure communication channel to address r_2.

Figure 8.2 shows the wet lab scenario deployed on our reference architecture. We note that the 5G node has two MEC applications that refer to the two different orchestration deployed one for t_2 and one for t_7.

We also note that orchestration t_7 is deployed on MEC to support requirement r_1 providing a better response time.

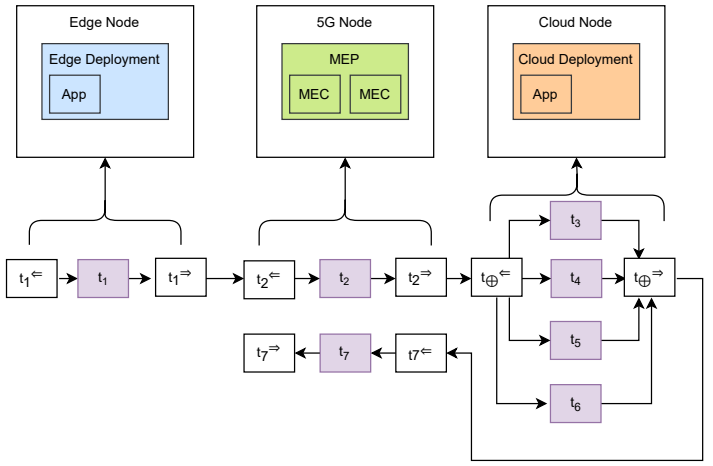


Figure 8.2: The wet lab pipeline deployed on our 5G-enabled Edge continuum architecture.

8.6.1 5G Orchestration Deployment

```
tosca_definitions_version: tosca_simple_yaml_1_3
description: An example VNFD template.
metadata:
  template_name: example_vnfd_template
topology_template:
  node_template:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        image: cirros-0.5.2-x86_64-disk
        availability_zone: central
        flavor: m1.tiny
      actions:
        failure: restart
    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        mac_address: fa:40:08:a0:de:0a
        ip_address: 10.10.1.14
        type: vnic
        anti_spoofing_protection: false
        management: true
        order: 0
        security_groups:
          - secgroup1
          - secgroup2
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1
    VL1:
      type: tosca.nodes.nfv.VL
      properties:
        vendor: Tacker
        network_name: net-01
```

Figure 8.3: VNFD file example (an extract) for a simple deployment on MEC suitable for the orchestration involving t_2.

In the following we describe the deployment of the t_2 orchestration (i.e., the orchestration involving t_2) on the 5G MEC. The 5G system supports deployment techniques such as VNF to host services in its core network, extending its functionalities. VNF uses VNFD to define the configuration of a deploy-

```
virtualComputeDescriptor:
  virtualComputeDescId: orechestrator.node.comp
  logicalNode:
    - id: orechestrator.node.comp.req
      logicalNodeRequirementDetail:
        cpu: 4
        memory: 10G
  mcioConstraintParams:
    - localAntiAffinityZone
virtualStorageDescriptor:
  - id: orechestrator-storage
    typeOfStorage: FILE
    fileStorageData:
      sizeOfStorage: 10 # GB
      fileSystemProtocol: NFS
      intVirtualLinkDes: orechestrator-nfs
  - id: cirros-image
    typeOfStorage: BLOCK
    blockStorageData:
      sizeOfStorage: 2 # GB
      swImageDesc: cirros-0.5.2-x86_64-disk
selectedMECCHostInfo:
  - hostName: nodes.mec.1
    hostId:
locationConstraints:
  - countryCode: IT
vimConnectionInfo:
  id: orechestrator.vim.1
  vimType: BRIDGE
```

Figure 8.4: AppD file example (an extract) suitable for the orchestration involving t_2.

ment, describing the resources to allocate and the system image to use. VNFs handles a complete virtualised system, automatizing the production cycle of applications and services running in a controlled environment. Figure 8.3 shows an example VNFD file to deploy a CirrOS virtual machine as a VNF to work as MEP for the t_2 orchestrator.

The process of deployment t_2 orchestration is made of the following steps involving the 5G services as in Figure 4.3:

- The OSS requests the instantiation of an application (i.e., our t_2 orchestration) to the MEAO including a configuration in the AppD format

(an example of the AppD configuration is shown in Figure 8.4);

- The MEAO checks the configuration and authorization of the request. Then it selects an edge host that matches the requirements and forwards the request to its Mobile Edge Platform Manager (MEPM);

- The MEPM concretizes the configuration, introducing the necessary traffic, DNS and service exposure, and dependency rules, and requests the VIM to allocate the necessary resources and start the application;

- Once the VIM has confirmed that the application is running, the MEP activates the network configuration rules by allowing it to communicate with the data network through *Mp2*. When the running services are confirmed to be healthy, their availability is reported to the rest of the cluster and exposed through *Mp1* to the rest of MEC services.

- The chain of requests returns from the MEP to the MEPM and then to the MEAO, reporting information about the allocated resources;

- Finally, the MEAO returns the results of the instantiation process to the OSS, including an identifier to the application instance.

8.7 Discussion

There is an increasing need for solutions capable of deploying analytics pipelines in continuum Edge scenarios unlocking the full potential of Edge computation. The adoption of next-generation 5G networks will allow having Edge computation capabilities in the periphery leading to broader applications and fostering the distribution of micro Edge nodes embedded in the telco network. Edge computation can provide several benefits, such as low latency, high bandwidth, context awareness, and privacy preservation. In this chapter, we present an orchestration-based solution for handling advanced data-intensive pipeline deployment in the 5G-enabled Edge continuum. Our solution aims to provide an efficient and effective way of deploying analytics pipelines in the 5G-enabled Edge continuum, taking into account the requirements and preferences of the users, the characteristics and constraints of the resources, and the dynamics and uncertainties of the environment.

Chapter 9

Assurance Aware Deployment in E2C Continuum

Cloud-native technologies are drastically improving the way infrastructure resources are allocated and applications are deployed. For developers, the ability to encapsulate applications in containers or virtual machines simplifies the development process, making it independent of and transparent to deployment. Containerization allows also to build applications following the distributed paradigm, where components, mainly services, are chained into complex compositions.

The advent of Edge Computing has changed the way a service composition can be deployed, allowing for instance to better support real-time applications via low latency [89] and in general a better placement of services to fulfil specific QoS requests [90, 91]. Despite being physically separated from the Cloud, the Edge heavily depends on it to execute resource-hungry tasks, since computational power available on Edge nodes decreases the farthest they are from the Cloud. This dependency has been emphasized in recent times, when the momentum gained by ML pointed out the computational limitations of Edge Computing and consequently stimulated the development of distributed approaches relying on both Cloud and Edge, such as Federated Learning (FL) [92].

In such context, a new computational paradigm is taking hold, merging both Cloud and Edge in a Continuum. E2C Continuum combines the strengths from both platforms in terms of scalability, flexibility, and mobility, to name but a few, and delivers a wide infrastructure where applications can be seamlessly deployed across the entire network [93]. The aim of the Continuum is to decouple the management of heterogeneous resources and the deployment configuration from the execution of the specific application. This decoupling can be achieved by abstracting the deployment infrastructure, allowing the developer to simply define applications in terms of functional capabilities (i.e., a workflow of services) and QoS properties to hold (e.g., confidentiality, availability, latency, etc.). In our approach we assume that most of such NFPs be guaranteed thanks to suitable deployment configurations, without the need of modifying the application logic. For instance, considering an application on the continuum made of a pipeline of services focused on collecting sensible data to build a specific model. Low latency and privacy properties can be granted for the data gathering services via on premises or edge deployment, while high computational power can be guaranteed for the data modelling services via cloud deployment. We note that in a Cloud-only or Edge-only scenario, we may not benefit from all the properties without requiring modification at application logic, whereas in the Continuum, the same requirements may be fulfilled by specific deployment recipes.

Notwithstanding the potentialities offered by the E2C Continuum, the research into how to fully exploit its potential is still in its infancy. Some solutions have been proposed in the field of serverless computation, where stateless applications are executed in modern FaaS platforms [94–96]. In FaaS, frameworks are developed to select the best platform according to some metrics, such as cost. However, stateless applications focus only on resource allocation (i.e., CPU, memory and bandwidth) and represent only a portion of a much wider and complex range of E2C Continuum applications requiring much more complex properties to hold. For instance, the metrics defined for the selection of the optimal FaaS platform cannot be used to cope with latency, confidentiality and authentication [94, 95, 97, 98]. In short, we still lack a NFP-aware approach to deploy applications in the E2C Continuum, since existing solutions work mainly for serverless frameworks and fail short to handle pipelines of services and complex properties.

This chapter aims to address the above gap by defining a new methodology for service deployment in the E2C Continuum. We extended the graph-based representation of service composition in [27] to model also the Continuum

deployment environment peculiarities. We then build a matching function capable to map each service in the composition to a facility of the Continuum, so that all the required properties can be fulfilled. This mapping is finally used to generate and deliver deployment recipes specific for each Continuum facility.

The contribution of this paper is fourfold: i) a new notion of E2C Continuum involving both 5G Telco Edge node and on-premises Edge node (Section 9.1), ii) a novel methodology for QoS-aware deployment of composed services in the E2C Continuum (Section 9.2), iii) a suitable Architecture implementing the methodology and generating executable recipes for the deployment, and iv) preliminary experimental evaluation showing the feasibility (Section 11.5).

9.1 Scenario, Requirements and Architecture

Our reference scenario considers i) a client that wants to deploy its workflow of services s_i specifying QoS requirements and constraints to be satisfied on the E2C Continuum; ii) CSPs offering deployment facilities f_i for third-party services on the E2C Continuum; iii) facilities offering specific capabilities c_i to satisfy QoS requirements and constraints. In this work, we consider an advanced E2C Continuum where Edge nodes can be: i) telco nodes (i.e., 5G MEC) based on an agreement between the CSP and a given telco operator offering their core network capabilities to be part of the CSPs Continuum (e.g., AWS Wavelength); ii) on-premises nodes based on the deployment facilities on the client's premises enabled by the CSP for services deployment (e.g., using AWS Greengrass). When the client asks to deploy a given workflow of services, the CSP matches the service workflow, QoS and constraints with the capabilities and peculiarities of its E2C Continuum deployment facilities associated to the specific client in order to find all the feasible deployment configurations. Among them the CSP selects the configuration to be deployed according to internal policies, such as considering Cloud as the preferred service deployment due to lower operating costs. For instance, in case of two candidate configurations, one using 5G and Cloud facilities and another using on-premises and Cloud facilities, the latter would be selected. The CSP then generates the deployment recipe for the selected configuration and executes the deployment on the continuum.

Example 9.1.1 shows the scenario used in the rest of the chapter to present our methodology.

Example 9.1.1 (The Scenario)**.** Let us assume that a client wants to deploy a ML workflow made of 4 sequential services: s_1 gathering data, s_2 normalizing data, s_3 generating a model out of the collected data (e.g., training a decision tree), and s_4 saving the model for further usage (e.g., for prediction). Let us assume that the client requires that data accessed by s_1 should be protected for confidentiality and the model generated by s_4 should be protected from tampering by controlling its integrity. Let us assume that the client expresses a constraint on the communication links between s_1 and s_2 and between s_2 and s_3, which must carry large volumes of row data, requiring a bandwidth of at least 200 Mbit/s. Let us also assume the client has a contract with the CSP for Continuum facilities including on-premises machine f_1, 5G Edge node f_2 in the proximity of its premises, and Cloud node f_3. The CSP ensures some capabilities (c) on the facilities and the network links connecting them. In particular, f_1 ensures confidentiality at rest via isolation of the storage from the direct control of CSP (c_1), and f_3 provides a service (s_I) ensuring data integrity at rest (c_2). In addition, all the internal links, that is the links between two services deployed on the same facility, have infinite bandwidth (c_3), while the 5G link between f_1 and f_2 provide at least a 500 Mbit/s bandwidth (c_4).

9.1.1 Edge-continuum deployment Requirements

In order to support the Scenario in Section 9.1, the CSP should be empowered with an advanced deployment architecture addressing the following requirements:

- **R1 Continuum-readiness**: it should seamlessly deploy services on all the different Continuum premises.
- **R2 Property-driven**: it should be driven by QoS properties and constrains expressed by the client.
- **R3 Technology agnostic**: it should be capable to handle heterogeneous deployment facilities regardless the underlying virtualisation technology.

- **R4 Comprehensive model**: it should provide a way as general as possible to represent pipelines and facilities, without limiting the choice in topology and data flow.

- **R5 Interoperability**: it should be able to interact with CSP facilities through software hooks.

- **R6 Context adaptability**: it should perform deployment life-cycle management by re-deploying services when changes in the environment occur.

9.1.2 Deployment Architecture

To support the scenario in Section 9.1 and ensure the requirements in Section 9.1.1, we propose a deployment architecture capable of guaranteeing a seamless QoS and constraints preserving execution of a given service workflow in the Continuum.

Figure 9.1 schematizes our system architecture made of i) a *Deployment Engine* service that targets the deployment facilities through *Deployment Agents*, and ii) a set of *Deployment Facilities* of different nature, including Cloud, Telco-Edge and on-premises nodes R1. The client interfaces with the deployment engine through *Deployment API* providing the service workflow to be deployed and the QoS/constraints specification in a machine-readable format R2 and R4. The *Deployer Solver* chooses the most appropriate deployment configuration interacting with the *Deployment Controller* which is in charge of i) interrogating Deployment Facilities on their capabilities and ii) delivering the deployment recipes, using the *Deployment Agents*.

The Deployment Agents are responsible to build the deployment continuum across the facilities by driving the service deployment R3. In order to support QoS and constraint-aware integration, the CSP exposes to the Deployment Agents suitable hooks to relevant resource (e.g., resource manager for deployment) or services (e.g., services offering security features) constituting their capabilities R5. For instance, the CSP can offer a hook to access non-functional certificates (i.e., using certification scheme in [6, 99]) proving some capabilities or to invoke authentication services to support authentication requirements.

We note that in case of necessity (e.g., changes in the QoS or budget constraints) the given workflow can be re-deployed via re-executing the deployment matching with modified QoS and constraints R6. The re-deployment can be also used to handle service migrations and in general changes occurring post-deployment.

In the following we describe the peculiarities of the Continuum *Deployment Facilities* in terms of architecture and capabilities.

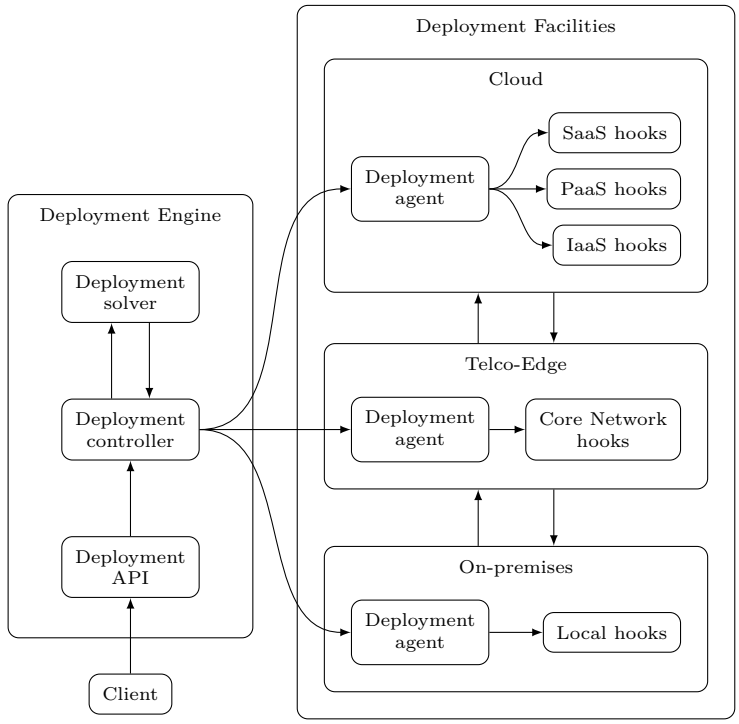


Figure 9.1: Deployment Architecture for E2C Continuum.

9.1.3 Cloud

Cloud solutions offer to their users managed services and infrastructure with scalable amount of resources. This allows users to deploy and integrate their

services and products, relieving them from most management tasks. Generally, the control plane of the system is handled by a cloud provider, managing the services life-cycle and ensuring their availability. The most common type of deployment used in this context are based on containerized services or on virtualised hosts.

The Cloud is the most flexible and powerful facility in the Continuum in terms of offered capabilities. It supports non-functional QoS via PaaS or IaaS services enabling them for the deployed services. It also partially supports constraints, for instance via resource scalability, network latency, high bandwidth and connectivity supported by virtualised network and local and distributed data centres. However, this support is normally very limited due to the public internet connectivity and the shared environment used in standard Cloud configurations. The Cloud currently suffers from the lack of transparency impacting NFPs as privacy [79].

To support our deployment architecture the Cloud facility has to provide hooks for i) resource management offering deployment and configuration of containerized and/or virtualised services; ii) configurations of services relevant to support given NFPs and iii) workflow-level networking management.

9.1.4 Telco-Edge

Telco-Edge is an innovative Edge scenario enabled by 5G via integrating mobile networking capabilities. The state-of-the-art solution for automated service management in such systems is based on MEC [13,100–102]. MEC allows integration of container and VM based services with the 5G core network, mutually exposing their functionality. Differently from traditional Cloud facilities, i) the edge nodes are connected to 5G radio antennas, allowing fast communication to and from mobile devices; ii) the service provisioning to mobile devices is backed by unique device and user identification through IMEI and Subscriber Identity Module (SIM) identifiers, supporting strong authentication; iii) the telco edge nodes can be geographically closer to their users, enabling new notion of data privacy by proximity, similarly to the notion of privacy on-premises; and iv) the 5G standard allows users to allocate virtual network slices, ensuring bandwidth availability and network performance levels and latency.

In addition, by adopting the Telco-Edge facilities i) the bandwidth between the edge node and other network nodes is allocable; ii) the in-motion data can be contained within the boundary of the telco network; and iii) additional services (i.e., identification, network monitoring, authentication) for the users connected through mobile network are available.

To support our deployment solution, the Telco-Edge facility have to offer hooks on resource management and service configuration similarly to the Cloud but also additional peculiar hooks for i) network-level user management and authentication, and ii) network resource allocation (5G network slices) for bandwidth and latency management via MEC.

9.1.5 On-premises

With on-premises we consider deployment facilities that are fully under the control of the owner, but are equipped with CSP services to make them part of the Continuum. They can be realized via VMs/containers or physical machines. Such facilities have normally stronger resource limitations compared to Cloud or Edge environments, lacking scalability and elasticity or not providing specific hardware. On the contrary, they i) have strong properties of high confidentiality and privacy, for instance ensuring that data cannot physically leave the organization; ii) have the lowest latency to the devices within the organization; and iii) leave the owners full control of the execution environment.

In order to be part of the Continuum, on premises facilities should i) allow our Deployment Agents to be executed in a supported deployment environment; ii) access to resources to arrange the service deployment; and iii) offer hooks to handle resources for local deployment of services, connectivity to the continuum and access to the relevant local services if needed. Normally, on premises hooks have very restricted access to local services, data and deployment resources making their use in the continuum very challenging.

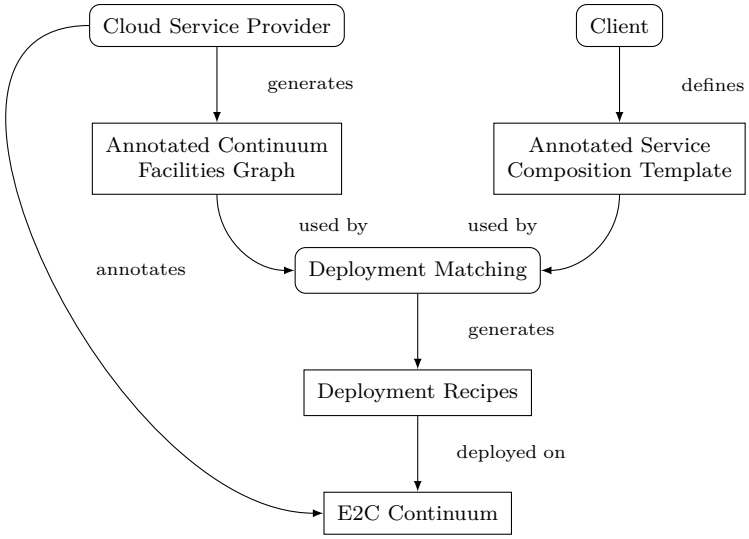


Figure 9.2: Our Methodology.

9.2 Methodology

Figure 9.2 shows our methodology for the E2C Continuum scenario in Section 9.1 based on our architecture in Figure 9.1. The client defines the service composition workflow to be deployed and annotates it with QoS requirements and constraints forming an *Annotated Service Composition Template*. The service provider annotates its facilities with the relevant properties, indicates hooks reachable by Deployment Agents, and produces the *Annotated Deployment Facilities Graph*. The client submits the request for deployment by submitting the Service Composition Template via Deployment API to our Deployment Engine. The Service Composition Template and the Annotated Deployment Facilities Graph triggers the *Deployment Matching* process executed by our Deployer Solver in order to generate the *Deployment Recipes* to be used to deploy the given workflow of services on the E2C Continuum.

9.2.1 Annotated Service Composition Template

A Service Composition Template is an abstract representation of the workflow of services that a client wants to deploy. It describes the services, their execution parameters and their interconnectivity configuration.

Definition 9.2.1 (Service Composition Template)**.** A Service Composition Template is a directed graph $T = (S, E)$ where $s_i \in S$ are services constituting the graph vertexes, and $e_i \in E$ are graph edges modelling the interaction between two services on both control and data plane. We note that, since $T = (S, E)$ is a directed graph, each edge represents a one-way interaction, while a two-way communication requires a cycle made by two edges.

The Service Composition Template can be annotated with specific non-functional requirements $r_i \in R$ and constraints $k_i \in K$ on resources. Following the notation in [103], a requirement r_i is a pair $(\hat{r}, Attr)$, where $r_i.\hat{r}$ is an abstract requirement from a shared vocabulary of properties (e.g., confidentiality, integrity) and $r_i.Attr$ is a set of attributes specifying the low-level characteristics that should be provided. The attribute values induce a hierarchy H_R of requirements $(R, \preceq_R)$, where R is the set of requirements and $\preceq_R$ is the partial order. Similarly, a constraint $k_i \in K$ is a tuple $(\hat{k}, Val, Op)$, where $k_i.\hat{k}$ is a resource (e.g., bandwidth, on-premises), $k_i.Val$ is the desired value and $k_i.Op$ is the operation on that value. The resource determines what type of value can be specified, such as integers, booleans, lists, and what operations can be applied (e.g., $=, <, \geqslant, \in$).

Definition 9.2.2 (Annotated Service Composition Template)**.** Let $T = (S, E)$ be a Service Composition Template. The Annotated Service Composition Template $T^{R,K}$ is generated annotating vertexes and edges in the template $T = (S, E)$ with non-functional requirements $r_i \in R$ and constraints $k_i \in K$.

For instance, a security requirement $r_i = (\textit{Confidentiality}, \textit{AES256})$ can be associated to an edge (i.e., communication channel) e_i, denoted as $e_i^{r_i}$, while a constraint $k_j = (\textit{On-premises}, \texttt{true}, =)$ can be associated to a vertex (i.e., a service of the workflow) s_j denoted as $s_j^{k_j}$ indicating that the deployment must be performed on an on-premises facility.

Example 9.2.1 (Annotated Service Composition Template)**.** Considering the Example 9.1.1, we can represent the client service workflow as the annotated template $T^{R,K}$ with the set of services $S = [s_1^{r_1}, s_2, s_3, s_4^{r_2}]$ and the set of edges $E = [e_1^{k_1}, e_2^{k_1}, e_3]$. The expressed requirements are $r_1 = (\textit{Confidentiality,Isolation})$ and $r_2 = (\textit{Integrity,Rest})$, while the posed constraint is $k_1 = (\textit{Bandwidth}, 200, \geqslant)$.

9.2.2 Annotated Continuum Facilities Graph

The CSP models its Continuum facilities for a given client as a Continuum Facilities Graph.

Definition 9.2.3 (Continuum Facilities Graph)**.** The Continuum Facilities Graph is a directed graph $G = (F, L)$ made of a vertex $f_i \in F$ for each facility provided by the CSP. L is the set of edges (here called links) l_i connecting the vertices of the graph. Two vertices f_i and f_j can be connected by a link l_i if facility f_j is reachable from facility f_i either through the open Internet or a dedicated private channel.

Note that the Continuum Facilities Graph is directed and therefore each link models a one-way connection. We can however express both two-way and intra-node communication capabilities by defining cycles. In particular, intra-node communication is represented as a cycle of length 1, i.e., a loop.

The Continuum Facilities Graph generated is then annotated with non-functional capabilities $c_i \in C$. Capabilities represent a mechanism to support both NFPs constraints such as confidentiality, latency, performance to name but a few. A capability is defined as a tuple $(\hat{c}, Spec, Op, Impl)$, where $c_i.\hat{c}$ is either a property or a resource, $c_i.Spec$ is an attribute if $c_i.\hat{c}$ is a property, a value otherwise, $c_i.Op$ is an operation on $c_i.Spec$ (if it is an attribute, Val is always a $=$), and $c_i.Impl$ is a set, even empty, of key-value pairs describing how the facility implements the capability. The annotated data life-cycle is managed by the service providers, possibly using certification-based solutions. The selection and implementation of the associated methodology is out of the scope of this work.

Definition 9.2.4 (Annotated Continuum Facilities Graph)**.** Let $G = (F, L)$

be a Continuum Facilities Graph. The Annotated Continuum Facilities Graph G^C is generated annotating vertexes and edges in the Continuum Facilities Graph $G = (F, L)$ with the capabilities $c_i \in C$ (e.g., latency, bandwidth, resources).

For instance, a security capability $c_i = (\textit{Channel\_encryption}, \textit{AES256}, =)$ can be associated to a link l_i denoted as $l_i^{c_i}$, while a capability $c_j = (\textit{Edge}, \texttt{true}, =)$ can be associated to a vertex (i.e., a facility of the ISP) f_j denoted as $f_j^{r_j}$.

Together, Annotated Service Composition Template and Annotated Continuum Facilities Graph provide a general framework to describe most kind of pipeline topologies in the Continuum, addressing R4.

Example 9.2.2 (Annotated Continuum Facilities Graph)**.** Considering the Example 9.1.1, we can represent the CSP facilities as the annotated graph G^C with the set of facilities $F = [f_1^{c_1}, f_2, f_3^{c_2}]$ and the set of links $L = [l_1^{c_3}, l_2^{c_4}, l_3^{c_3}, l_4, l_5, l_6, l_7^{c_3}]$. The provided capabilities are $c_1 = (\textit{Confidentiality}, \textit{Isolation}, =, [])$, $c_2 = (\textit{Integrity}, \textit{Rest}, =, [\textit{service}: \textit{sI}; \textit{mode}: \textit{interception}])$, $c_3 = (\textit{Bandwidth}, +\infty, =, [])$ and $c_4 = (\textit{Bandwidth}, 500, \geqslant, [])$.

9.2.3 Deployment matching

The deployment matching process searches for the most suitable solution for the QoS-aware deployment of a given service workflow on the Continuum. It takes as input the Annotated Continuum Facilities Graph G^C and the Annotated Service Composition Template $T^{R,K}$, generates a set of suitable deployment solutions M and among them finds the one $\hat{M}$ that better satisfies a given CSP policy (e.g., the lowest operational cost).

The set of suitable deployment solutions M is defined on $(S \in T^{R,K}) \times (F \in G^C)$ so that: i) for every edge $e_i \in E$ between any two vertices s_i and $s_j \in S$ there is a link $l_i \in L$ between the matching vertices f_i and $f_j \in F$; ii) for every pair (s_i, f_i) in M, the capabilities c_i of f_i satisfy the requirements r_i and the constraints k_i of s_i; iii) for every edge e_i between any two vertices s_i and $s_j \in S$, the capabilities c_i of l_i between the matching vertices f_i and $f_j \in F$ satisfy the requirements r_i and the constraints k_i of e_i. If the set

Algorithm 3 Pseudocode of our deployment matching exhaustive algorithm.

```
procedure MATCHING(S, F)
    matches = ∅
     ▷ Iterate services permutations
    for service_perm in perm(S) do
         ▷ Services partitioning in |F| facilities
        for part in partitions(service_perm, |F|) do
             ▷ Test if all requirements are met
            valid ← true
            for r_i in R do
                if r_i(S, F, part) == false then
                    valid ← false
                    break
            if valid then
                 ▷ Match found
                matches = matches ∪ {part}
    return matches
```

of suitable deployment solutions M is empty, it means that the deployment of the service workflow cannot take place on the given continuum, given the QoS requirements and constraints. If the set of suitable deployment solutions M is not empty, the deployment matching orders them according to the CSP policy and selects the first one in the order.

Given the set of deployment solutions M, if the CSP policy refers to operational cost reduction only, a solution like the one in [104] can be adopted. We will investigate the impact of more articulated CSP policies as well as the adoption of an optimization approach for finding the suitable deployment solution and contemporaneously addressing such CSP policy in our future works.

Algorithm 3 shows the pseudocode of our matching function. The algorithm i) iterates over all the permutations of services altering the elements starting from the beginning of the list; ii) for each permutation it iterates on its partitions starting from the beginning of the list, thus leaving the largest partitions containing the most preferred facilities; iii) for each permutation it checks whether all requirements are met, if that is the case it adds it to the results partition otherwise it breaks to the inner for loop; iv) finally it

returns the set of results.

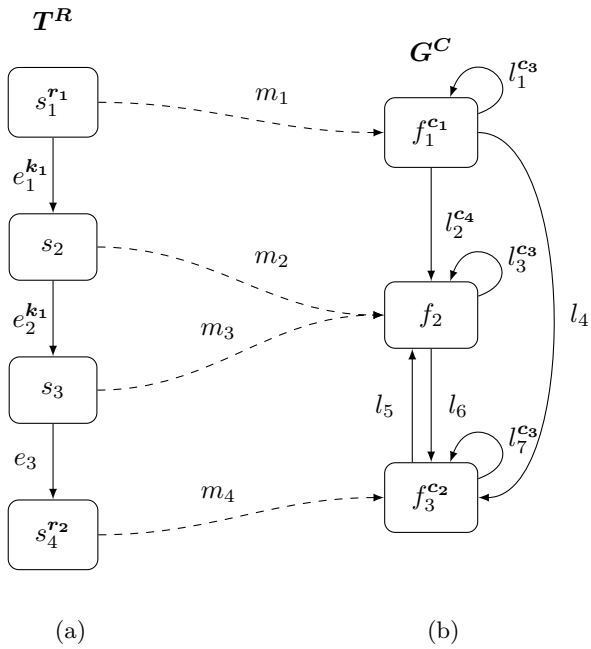


Figure 9.3: Annotated Deployment Graphs including (a) Annotated Service Composition Template and (b) Annotated Continuum Facilities Graph, with the resulting matching.

Example 9.2.3 (Deployment matching)**.** Let us consider the scenario in Example 9.1.1, the Annotated Template $T^{R,K}$ and the Annotated Graph G^C, in Example 9.2.1 and Example 9.2.2 respectively. Let us now apply our matching function in Algorithm 3 to the services in $T^{R,K}$ and the facilities in G^C retrieving a set of possible matching M. Figure 9.3 shows one of the possible solution in $M = \{m_1, m_2, m_3, m_4\}$, where $m_1 = (s_1, f_1)$, $m_2 = (s_2, f_2)$, $m_3 = (s_3, f_2)$, and $m_4 = (s_4, f_3)$. Here, s_1 is matched with f_1 since it is the only facility ensuring confidentiality. Similarly, s_4 is matched with f_3 since it is the only facility providing a way to check integrity. Lastly, s_2 and s_3 are matched with f_2 to provide large bandwidth between the first three services.

9.2.4 Deployment Recipes

The purpose of the above deployment matching is to assign each service to a facility in such a way that all requirements (i.e., QoS and constraints) can be fulfilled and the CSP internal policy satisfied. However, a mere assignment is not always enough to enforce the desired properties. For instance, low latency can be achieved by simply assigning services to the physically closest facility, while, on the contrary, communication channel confidentiality requires configuring a component or service to handle message encryption and decryption. To address the above deployment challenges, our methodology enriches traditional deployment recipes with hooks metadata. This metadata is consumed by our Deployment Agents to enable relevant properties configuring or embedding facility's services in the service workflow to be deployed.

Recipes are generated for each service of the workflow in $\hat{M}$ and contain the operational instructions for the deployment of both the service and the supporting facility's components/services. In particular, a recipe consists of three parts: i) the deployment configuration of the service, including the image to be used and resources to be allocated; ii) a description of the support components/services to be integrated and their parameters (e.g., for an authentication component it includes the list of user credentials); iii) modality of integration (i.e., none, interception or wrapper).

Example 9.2.4 (Deployment Recipe)**.** Considering the selected matching $\hat{M}$ in Example 9.2.3, according to our methodology the relative deployment recipes are generated for all the services $s_i \in \hat{M}$. For simplicity let us focus on s_4 recipe only. Figure 9.4 shows an excerpt of the s_4 recipe provided to the deployment agent in f_3, along with the final deployment graph for $\hat{M}$. According to the recipe, the agent, in order to guarantee integrity, has to deploy an additional facility's service (*service:* s_I) that intercepts (*mode: interception*) incoming data from s_3, computes and adds to data a cryptographic checksum, and delivers it to s_4 for storing.

We note that details on how the recipes are generated out of the given selected matching $\hat{M}$ is out of the scope for this work. We will further investigate this topic in our future works.

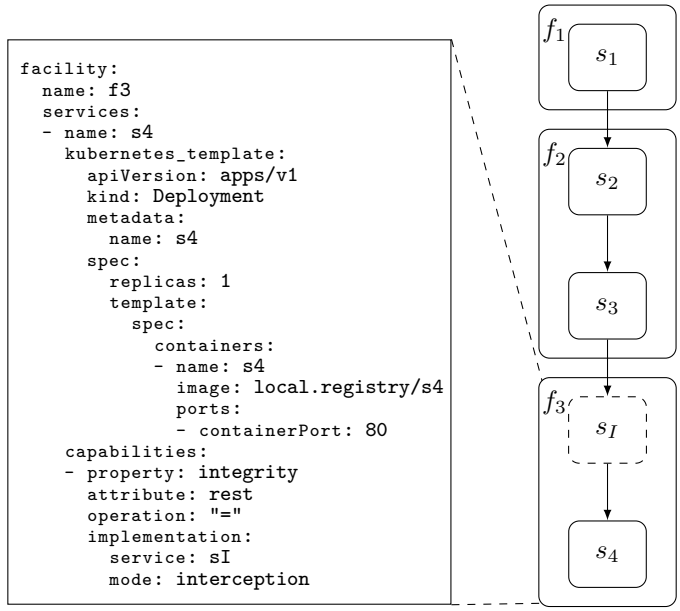


Figure 9.4: Deployment Recipe for facility f_3.

Chapter 10

Experimental scenario: MIND Foods HUB

In order to enhance crop plant varieties and generate favourable traits, such as increased yield, improved resistance to environmental stress, and heightened levels of health-promoting compounds, controlled (indoor) and field-based phenotyping experiments are conducted on both small and large scales. Phenotyping involves measuring various plant parameters and features through different methods. The ultimate goal is to identify and describe individuals that showcase desirable traits, which arise from the positive interplay between their genetic make-up and environmental conditions, as well as management factors applied during their growth. The significant advancements of recent decades in sensing, automation, and information technologies have revealed the era of "high-throughput" phenotyping through the creation of automated platforms, which are based on indoor growth-chambers or climate-controlled greenhouses. These platforms allow the monitoring of hundreds or thousands of potted plants with specific sensors. In recent years, field phenotyping has expanded significantly, incorporating larger scale experiments using mobile platforms to conduct non-destructive, traceable, high-spatial/time resolution measurements of plant features throughout the entire growth season, employing multiple sensors. The requirements for accuracy, frequency, repeatability and in-depth analysis of the vast amount of data measured make field phenotyping an ideal domain for the application of robotics, enabling highly efficient (semi-) autonomous data collection. Big Data engines handle data

management, integration with external sources, ground truth annotation, automated analysis and job pipelining, while providing intuitive interfaces to retrieve the extracted information [105]. Integration with 5G-based networks and services enables efficient handling of massive amounts of data from IoT devices [27], quickly delivering contextual information to the user in the field. It is important to note that the ongoing impact of these technologies is not limited to research infrastructures, but is becoming increasingly relevant to crop management practice. Indeed, with the adoption of Precision Agriculture (PA) technologies to monitor the spatial and temporal variability of crop development within fields, to model the site- and time-specific needs of the crop, and to precisely adjust the variable distribution of inputs (seeds, water, fertilisers, protective treatments, etc.), this data and automation-driven implementation is currently taking agriculture to a new level, often referred to as Smart Agriculture.

The contribution of this work is threefold. First, we propose a novel 5G-enabled Big Data platform for IoT. Second, we present a vertical implementation for addressing Smart Agricultural needs. Third, we present preliminary experimental evaluation in a real scenario of the MIND Foods Hub project. The rest of the chapter is organized as follows: Section 10.1 summarizes the background knowledge for machine-assisted phenotyping and big data platforms. Section 10.2 describes the primary components of our system. Section 10.3 details our practical scenario and its specifications. Section 11.6 reports the preliminary results achieved by our implementation. Section 10.4 gives our conclusions.

10.1 Background and motivation

The availability of reliable data collected along the growing season has always been considered of paramount importance for efficient and sustainable agricultural production. Over the last two decades, PA [106] has been gradually established as a reference approach to optimally manage the intra-field variability of crop needs and to modulate accordingly the farming practice and the distribution of inputs in different field zones. Today, PA is a set of sensing and actuation technologies that equip machinery and farm equipment, modelling tools linked to remote data sources, digital decision support tools, etc., used as an integrated system for planning the use of resources and managing

farm operations. At the basis of PA are sensors (mostly optical) mounted on agricultural machinery that travels the fields during normal operations or for scouting purposes at specific times, or mounted on Unmanned Aerial Vehicless (UAVs) or satellites to monitor the spatial and temporal variability of crop growth, water and nutrient status, health and competition from weeds. The data collected is fed into modelling and computational tools to produce prescription maps for subsequent management operations, which can be modulated site-specifically by PA equipment. Plant phenotyping uses similar or more advanced sensor technology to measure relevant characteristics with greater temporal and spatial resolution, even at the level of individual plants or organs. By far the most commonly used measurement techniques are based on optical sensing because they are non-destructive and can therefore be repeated without disturbing plant growth, and because many physical and physiological characteristics of plants strongly influence their optical properties, which can then be used as reliable proxy parameters. For instance, RGB cameras mounted in a top-view configuration along with associated segmentation algorithms can characterise rates and synchronicity of seed germination or measure growth rates of seedlings in the first developmental stage [107]. Similarly, specific phenological stages can be identified by counting or detecting outer organs such as ears, fruits, and flowers. Plant dimensions and outer foliage size and structure can be assessed using 3D imagery, such as stereo or time-of-flight cameras, or via Laser Imaging Detection and Ranging (LIDAR) systems [108]. By taking repeated measurements at various intervals, variations in plant growth rates may be determined. From 3D point clouds of a plant, some physiological morpho-geometric parameters can be estimated, such as total leaf surface, plant elongation, and number of leaves [109–113]. Thermal infrared images can be used to assess water availability and stress in plants, as tissue temperature is controlled by the transpiration rate through the stomata of leaves. Photosynthetic efficiency can be used as an indicator of plant health and vitality. Handheld specialized devices are commonly used to measure chlorophyll fluorescence, although automation for field phenotyping still poses technical challenges. Further insight can be gained through multi-spectral and hyper-spectral cameras, which are capable of obtaining images within narrow, specific spectral regions associated with nutrient status (in particular nitrogen) and pathogen infections [114]. Interestingly, this method can also estimate the levels of desirable compounds (such as those with high nutritional or health-protecting qualities) in plant tissues following specific management treatments. It may also be employed to evaluate the effectiveness of strategies aimed at reducing the accumulation of undesired compounds in the plant tissues.

Big data systems are designed to store and manage large amounts of data in a variety of forms, such as text, structured datasets or blobs of binary data such as images and videos. While the raw data remains critical, the extracted information represents the true value to an organisation or business. Consequently, the implementation of a fitting infrastructure to analyse, convert and leverage the data becomes necessary. Therefore, the designed system aims to not only store and retrieve use cases, but also to consistently transform incoming data through analysis pipelines. The system also necessitates a dat a distribution solution whereby subscribers interested in obtaining the most recent collected samples can opt for particular distribution channels and efficiently obtain dataset updates as soon as they are available.

The need for quality assurance in Big Data solutions is driven by the complexity and variety of analysis requirements, as well as the demand for easier debugging. This applies to both user and infrastructural levels. Techniques for Big Data assurance are used to ensure the accuracy of pipeline jobs and the suitability of the underlying infrastructure, as explained in previous Chapters 2, 3, 4, 5, 6, 8, 9 and 7.

10.2 System service components

In this section, we describe the architecture of our Big Data Engine platform. Figure 10.1 is a representation of the engine components and their interaction with external systems and users

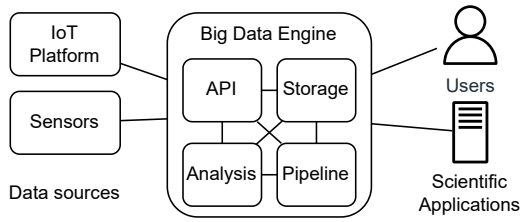


Figure 10.1: Big Data Engine components and interactions.

The primary use cases for a Big Data Engine are the following: i) storing large quantities of data in a distributed and searchable manner; ii) providing

an efficient execution environment for analysis and transformation of the data; iii) enabling users to automate and schedule repetitive tasks through a pipelining solution. In our microservice-oriented architecture, we use use specialized components for each of these scenarios in order to decouple their functionalities.

Storage The storage feature is handled by a data lake solution. In particular, we are interested in horizontal scalability of the storage space, data replication capabilities and ease of access through a restful API.

Analysis The execution feature is provided by a parallel execution engine. This solution automatically handles the tasks in input, providing them the necessary libraries, configurations and secure access to the data. Such execution platforms also provide frameworks to easily develop applications that can benefit from parallel execution, splitting the load on multiple execution nodes in the same cluster.

Pipeline The automation and scheduling of tasks is a feature integrated in a pipeline execution platform. These platforms provide frameworks and web services to create, configure and execute pipelines of tasks, managing their environment and possible failures. It is paramount that the chosen platform integrates with the execution engine, leveraging its parallel execution capabilities. The pipeline platform also provides an API to schedule and monitor the execution of jobs.

API Data ingestion and query is managed by a unified API that integrates with the previously mentioned components. Fluxes of sensor data, such as those provided by IoT platforms or smart sensors, are collected through the API and stored in the data lake. Depending on the type of data, the API may schedule the execution of analysis tasks though the pipeline service. The API also provides users and external applications querying capabilities in order to extract information form the processed data.

Data sources The Big Data Engine needs to support multiple data sources, like IoT platforms and smart sensors. Depending on the source type and

capabilities, the ingestion is implemented in a *push* or *pull* configuration. In the first case, the data source sends the data to the engine through its API, completely handling the update process. In the pull configuration, a specialized collector agent is implemented to retrieve updates from the data source, acting as an adapter between the two components.

Data consumers The consumers of the Big Data Engine are primarily specialized users and scientific applications, that require access to the stored data or to analysis results. They interface with the engine API in order to query the storage component for the required information. Trusted users may have access to the pipelining service, in order to develop the analysis or transformation processes.

10.3 System implementation

The following sections present our practical scenario implementation, based on MIND Foods Hub project[1]. In such project the Big Data Engine we proposed acts as the central hub for collection, elaboration, analysis and query of the agricultural data produced by on field sensors.

Figure 10.2 presents the major actors in the data cycle and their interactions. Follows a description of each component of the system.

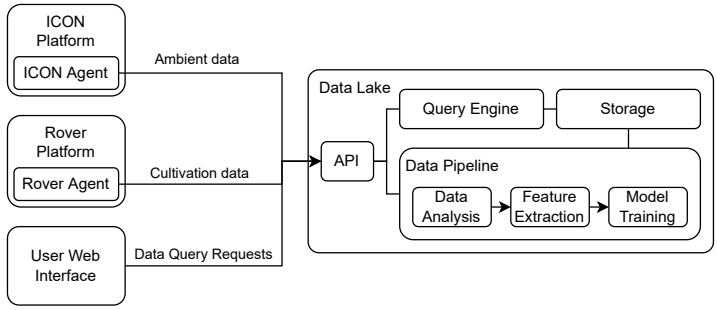


Figure 10.2: Robotic platform during a phenotyping mission.

[1] `https://www.mindfoodshub.com`

10.3.1 Data lake services

The data lake is a composition of software services, hardware and configurations aimed at ingesting, storing, analysing and querying large quantities of data.

The identified use cases of the data lake deployment can be resumed in the following points: i) continuous ingestion of structured data from the ICON platform; ii) ingestion of raw sensor data from the rover platform; iii) analysis and transformation of the ingested data through a unified and scalable solution; iv) user and script friendly interface for querying the stored data; v) data query service low response time.

The adopted solution had to meet the following additional requirements: i) the system has to constantly provide a high level of availability; ii) the compute and storage capabilities of the data lake have to be easily scalable.

The implemented solution will be used for future big-data research projects, therefore stability and replicability of the deployment are paramount. The system has been deployed on commodity hardware provided and managed by the SESAR lab of the Department of Computer Science of the University of Milan.

Figure 10.3 contains an overview of the components adopted in the big-data architecture and their interactions. A more in-depth description of each component follows.

Storage service

The storage functionality has been achieved with the adoption of the Apache Hadoop ecosystem, which offers a uniform and simple interface to a distributed filesystem. This solution allows for automatic replicas of data in a distributed and scalable manner, ensuring availability, consistency and privacy. The service exposes its filesystem through multiple protocols, including HDFS and a restful API, while providing access control functionalities.

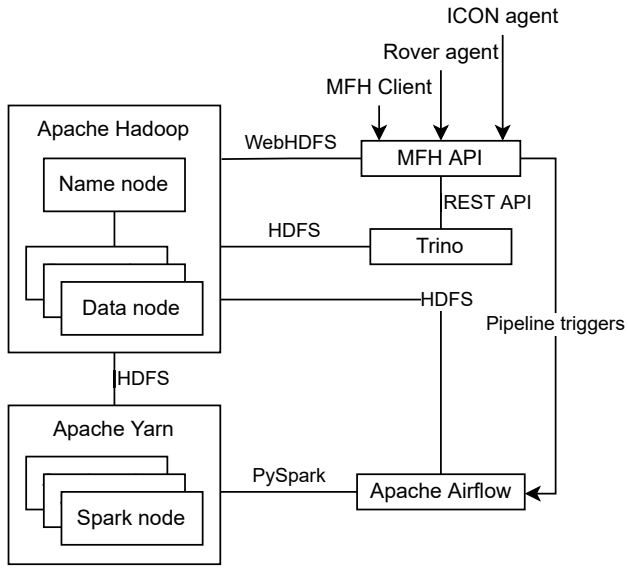


Figure 10.3: Software components of the data lake.

Data analysis framework

A data analysis framework provides useful tools to efficiently handle the data lake's stored data. These include (de-) serialization adapters, statistical and ML models implementations and data visualization primitives. The chosen solution is Apache Spark, for its effective and easy to use framework for parallel big-data computing. Spark is a platform commonly used for efficient and distributed computing in a large number of fields, including ML, statistics and data analysis. The framework has first-class integration with Hadoop and other storage solutions. It can be combined with several programming languages, including Python, Scala and Java.

Data query service

The data query service acts as an interface to the stored data, exposing it through an SQL interface. This component simplifies the integration of the

data lake with external services and tools, acting as an intermediary layer decoupling the query logic from the data lake implementation. The chosen service is Trino, a distributed SQL query engine supporting Hadoop and a variety of storage services and data formats. Trino exposes its features through a restful API and a SQL dialect, easing the integration with other frameworks. Trino's speed is sufficient for the quasi-real-time use case identified for the scenario, significantly outperforming other similar services.

Data pipelines service

The data pipeline service is the component that manages the execution of transformation tasks on the stored data, providing scheduling, logging and dependency management features. This service listens for triggering events, such as web-hooks, timers or filesystem events, to execute a set of preconfigured scripts describing the tasks. Each task can be partially ordered by a dependency DAG and is monitored until its completion. One of the data pipeline service primary functions is to provide a consistent environment of libraries and configurations to ensure repeatability.

Data query API

The data lake functionalities are exposed through a GraphQL API, providing a unified, self commenting and easy to integrate interface for other applications. The API is designed as a microservice, with scalability and ease of deployment in mind, and acts as the primary endpoint for both data ingestion and retrieval. The service interacts with both the storage service and the data query service. It can be used to trigger data pipelines, i.e. to elaborate newly ingested data. The choice of using a GraphQL based implementation permitted to have clear documentation tightly coupled with the source code and to provide a consistent interface over the stored information.

10.3.2 Sensor platforms

Sensor platforms manage physical sensors, handling their measure processes and exposing their functionalities to other software components. In our sce-

nario, we consider two sensor platforms: i) a rover platform that measures individual cultivation properties and ii) the TIM ICON platform that aggregates several environmental sensors.

Rover platform

Figure 10.4: Robotic platform during a phenotyping mission.

The field carrier, shown in Fig. 10.4, is a customized robotic platform built upon a commercial rover base (Infosolution Heavy Duty). The rover is equipped with hyper-spectral, multi-spectral and depth cameras, supporting the measurement of a wide range of cultivation characteristics. An autonomous guidance system allows the platform to automatically move to predefined locations and automate the sampling operation through waypoint based missions. Once all the data is collected, the rover sends it to the Big Data engine API through a specialized adapter application.

ICON platform

The TIM ICON (IoT CONnectivity) platform is an integration middleware aimed at the world of the Internet of Things, compliant with the oneM2M

standard. ICON is a horizontal platform and does not address a specific application area, but can be used in any vertical context in the IoT field, such as the different scenarios of Smart City, Smart Home, Industry 4.0, etc. It allows storing the data originated by the sensors and to share them with the applications (store & share paradigm). To store data, ICON uses MariaDB and MongoDB. ICON implements a common service layer (Figure 10.5) to uniformly manage the different IoT devices, typically characterized by a plethora of heterogeneous technologies. Applications that interact with devices can thus have a standard interface, which abstracts the complexity of the underlying world.

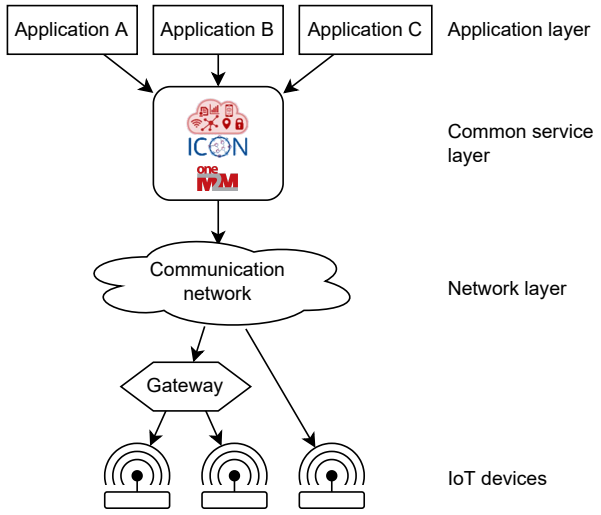


Figure 10.5: ICON Architecture.

ICON makes available a south-bound interface to the world of sensors (to publish data on the platform) and a northbound interface to the world of applications (to access archived data). The ICON platform used in this work is a lab instance available for integration activities involving external partners. It is connected with the ICON production environment to get data from deployed IoT devices connected to the commercial TIM NB-IoT network, as described in Figure 10.6 where are also visible the different possibilities to connect sensors to the south-bound interfaces. The oneM2M interfaces are exposed as an HTTP restful API.

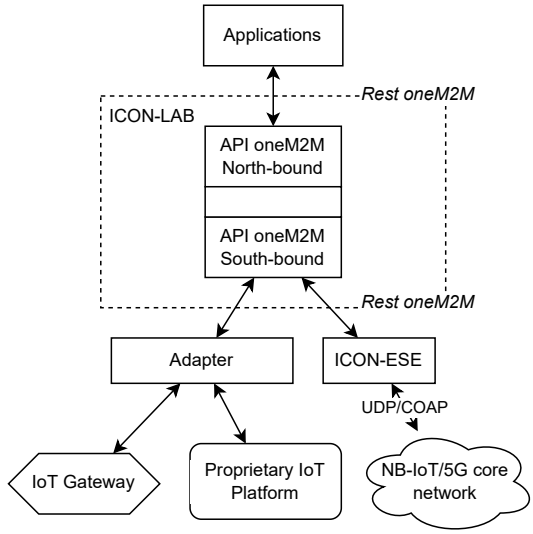


Figure 10.6: Architecture with ICON-Lab and ICON-ESE (production) instances.

The underling data model is based on the *Container* concept, a basic data structure oneM2M-compliant that can contain data referred to any kind of device. Containers can be nested and are made of Content Instances, a minimum set of meaningful data for a specific container.

IoT Sensors To monitor some environmental characteristics and climatic conditions that affect growth and health of crops, IoT (Internet of Things) sensors have been installed in cultivation areas. These sensors measure the quantities to be monitored and are connected by cable to data-loggers that collect the measured data at regular intervals and send them via the NB-IoT network to ICON platform to be stored and then made available to applications. The data-loggers are equipped with autonomous power supply (via battery and solar panel) and can be easily positioned on field. The sensors were split up in three different data-loggers: i) in the greenhouse tunnel to measure air temperature and humidity and soil temperature and tensiometric values at two depth levels; ii) in the open field sensing soil temperature and tensiometric values at two depth levels and iii) on field border to measure

weather-related metrics (rain level, wind direction and speed, air humidity and temperature, solar radiation and leaf wetness).

10.3.3 Data flow

In our system, we identify three types of data flows: i) data ingestion, where information from external sources is collected, enriched and stored in the data lake; ii) data transformation, where data that is stored in the data lake is used as material for producing more refined or specialized information; iii) data query, where the engine exports information on user request. In the following sections we expand on the data-related tasks our system performs.

ICON data collection

A network service, named *ICON agent*, is tasked with the goal of collecting the information exposed by the ICON platform. To do so, the application implements a ONEM2M client and schedules queries to ICON at a fixed rate, requesting recent updates on the exposed sensors. If a new state is found, the data from ICON is parsed and checked for consistency. Each measurement is enriched by including metadata about the sensor and a timestamp of measure. Finally, the service records the measurements in the data lake by sending a registration request to the API. Once the data is stored, data pipelines may be triggered depending on the service configuration.

Rover data collection

The rover platform uses waypoint-based missions to gather data from the cultivations: starting from a known position, it moves towards the next geo-marker with a sufficiently small error and triggers the acquisition of data from the on-board sensors. Such sensors produce two main type of data: plants' 2D images obtained from the multi-spectral and hyper-spectral cameras and plants' 3D point clouds. Images and point clouds are respectively encoded in the PNG and PCD file formats. Once all the required data is collected, the rover uses a network client, named *Rover Agent*, to send the files to the data lake through its API.

10.3.4 Data pipeline

The data pipeline is composed by a sequence of operations on the ingested or stored data. An example pipeline may include an initial analysis step that filters the data and checks for consistency; a feature extraction step that enriches the data by inferring additional information, and a final step where the data is fed as a training dataset to a regression model. Each of these steps is monitored to ensure availability of the resources and integrity of the obtained results. The produced data can then be published in the data lake and made accessible through the data query API.

User data queries

The data lake provides a user level web interface and GraphQL API to allow both humans and applications to send structured requests and retrieve information. The API converts the requests received into queries which are submitted to the query engine, retrieves the results and returns them to the users. The API can also be integrated with the ACL of the data lake, providing to each user a different level of access to the stored resources.

10.4 Discussion

The Big Data solution presented in this paper met the requirements exposed in Section 10.3.1. The storage and transformation system proved to be stable and easy to deploy on multiple machines, while being effective and versatile. The data query component, Trino, has been successfully deployed and integrated with the rest of the services. Moreover, it has demonstrated high and consistent performance in the relevant use cases. The integration between the data lake components has been implemented in the *MFH API* and pipeline services by using the available restful APIs. The Spark framework provided easy access to the Apache services ecosystem, while the GraphQL technology significantly simplified the resources' accessibility.

Chapter 11

Experimental results

This chapter presents the experiments conducted and findings obtained throughout the various stages of this research. The subsequent sections outline the experimental procedures implemented to authenticate the experimental setups and methodologies presented in the previous chapters.

11.1 5G Simulator setup

Table 11.1: Setup of the virtual machines composing the Testbed.

Resource	**5gcnl-oran**	**5gcnl-osm**
Virtual CPUs	4 virtual cores	12 virtual cores
RAM	20 GB	32 GB
Storage (SSD)	250 GB	300 GB
OS	Ubuntu 18.04 LTS	Ubuntu 20.04 LTS
Juju	v2.9.42	v2.9.42
K8S	MicroK8s v1.26.4	MicroK8s v1.26.4
MicroStack	None	Ussuri

Our Testbed is composed by two different virtual machines. One contains Aether and all its components (*5gcnl-oran*) while the second contains the NFV orchestrator and the intent extension (*5gcnl-osm*). Table 11.1 describes

the characteristics of the two virtual machines. On the *5gcnl-oran* machine we conducted infrastructure experiments to understand how Aether affects the system in terms of CPU cores usage and memory allocation. We applied the same test on the *5gcnl-osm* machine, but we also decided to check if the deployment of several network services degrades the performances of OSM over time. Both Aether and OSM are deployed in Kubernetes clusters and to be precise, the components are deployed in different Kubernetes namespaces. On *5gcnl-oran* we have three namespaces:

- omec: even if the name suggests the utilization of OMEC's 4G EPC, in this namespace are deployed all the 5G Core Network elements developed by Free5GC.
- sdran: in this namespace are deployed the RIC components described in Section 4.3.1.
- aether-roc: in this namespace are deployed the ROC components described in Section 4.3.1.

On *5gcnl-osm-20* we evaluated the performances of one namespace:

- osm: in this namespace are deployed the modules of the architecture of ETSI OSM as described in Section 4.3.2.

Table 11.2 shows the network services deployed during the test on *5gcnl-osm-20*.

11.1.1 5gcnl-oran experiments

Methodology

To evaluate the performances of Aether, we developed a simple Python program that collects metrics from Kubernetes *metrics-server*, an add-on that provides resource utilization metrics for monitoring the cluster and its workloads. We opted to measure the number of CPU cores and the bytes of memory allocated by the Pods in the three namespaces mentioned above. The program performs the following operations:

Table 11.2: List of all the network services deployed on OSM during the test.

Network Service	Helm Chart	Description
CoreDNS	stable/coredns	A DNS server
Joomla	stable/joomla	A Content Management System
Keycloak	stable/keycloak	An Open Source IAM solution
LAMP	stable/lamp	A bundle to develop web applications
Minecraft	stable/minecraft	The server of a videogame
MySQL	stable/mysql	A Data Base Management System
OAuth2 Proxy	stable/oauth2-proxy	A proxy to provide authentication
Open LDAP	stable/openldap	Lightweight Directory Access Protocol
PHP My Admin	stable/phpmyadmin	A tool to manage MySQL
PostgreSQL	stable/postgresql	A Data Base Management System

1. It collects CPU and memory metrics every five seconds for ten minutes.

2. At the end of the test, the results are stored in a `.csv` file.

5gcnl-oran CPU performances

In this subsection we will show the results obtained from the analysis of the plots regarding the CPU core consumption of Aether's namespaces.

Namespace omec Figure 11.1 shows the number of CPU cores that every 5G Core Network component consumes once deployed. All 5G Core Network components use around 0.23/0.24 CPU cores except for the UPF that consumes 0.21 CPU cores. The two *mongodb* instances consume 0.26 CPU cores and occupy the first positions. The other components consume less than 0.01 cores.

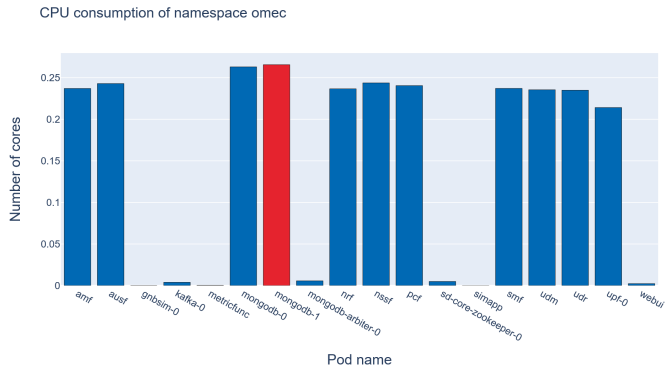


Figure 11.1: Number of CPU cores utilized by the 5G Core Network components.

Namespace sdran Figure 11.2 shows the number of CPU cores that every 5G RAN component utilizes once deployed. The CPU consumption of the 5G RAN components is very low in the idle state; Pod *onos-e2t* and the *Atomix controller*'s elements consume less than 0.002 CPU cores while Pod *onos-consensus-store* consumes 0.035 CPU cores. Other *micro-onos* components use an amount of CPU cores that is around 0.001.

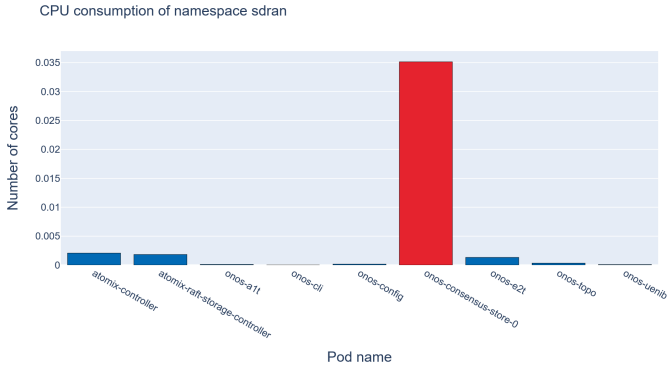


Figure 11.2: Number of CPU cores utilized by the 5G RAN components.

Namespace aether-roc The consumption of CPU cores by the ROC controller Pods is particularly low, as shown in Figure 11.3. The Pod *onos-consensus*, consumes barely 0.0042 cores and occupies the first position. It is followed by *onos-consensus-store* with 0.003 cores, by *onos-topo* with 0.0014 cores and *aether-mock-exporter* with 0.0013 cores. The remaining Pods consume less than 0.001 cores.

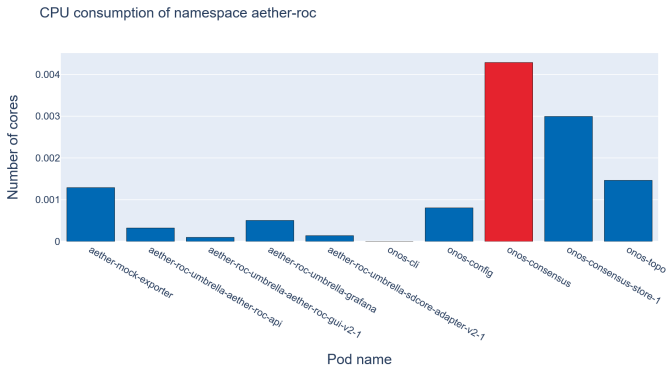


Figure 11.3: Number of CPU cores utilized by the ROC.

5gcnl-oran memory performances

In this subsection we will show the results obtained from the analysis of the plots regarding the memory allocation of Aether's namespaces.

Namespace omec In Figure 11.4 we show the quantity of bytes that every 5G Core Network component allocates once deployed. In this case, it is worth noting that most 5G Core Network components allocate a low quantity of bytes of memory. They allocate approximately 10/20 MB of memory except the UPF that allocates 206 MB of memory. Pod *kafka* occupies 425 MB of memory while the two instances of *mongodb* 350/378 MB respectively. Pods *sd-core-zookeeper* and *mongodb-arbiter* instead occupies 292 and 193 MB. The other components allocate less than 10 MB each.

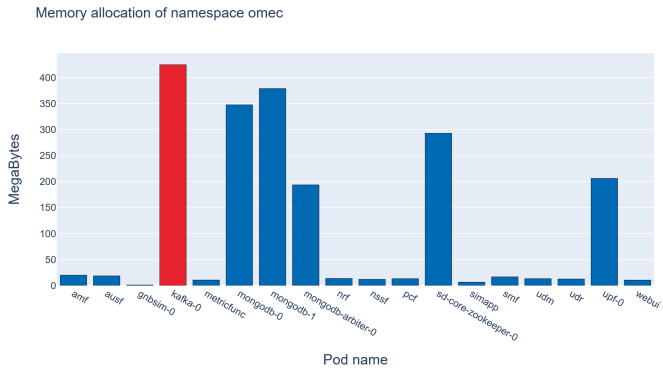


Figure 11.4: Memory allocated by the 5G Core Network components.

Namespace sdran The memory allocation of the 5G RAN components is shown in Figure 11.5. Again the Pod *onos-consensus-store* consumes the most resources. It allocates 65 MB of memory followed by *atomix-controller*, *atomix-raft-storage-controller* and *onos-e2t* Pods that consume 21.5, 19.8 and 25.8 MB respectively. The remaining components allocate less than 5 MB each.

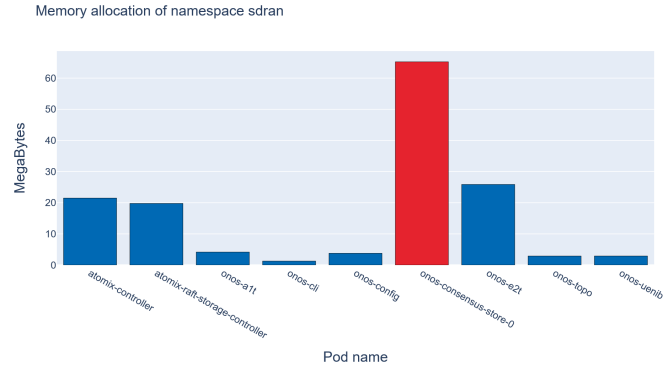


Figure 11.5: Memory allocated by the 5G RAN components.

Namespace aether-roc The analysis of the memory occupation by the ROC controller Pods in Figure 11.6 shows that *onos-topo* is the component that allocates the most memory, with a value that is around 96 MB. Another evident peak is generated by the Pod *aether-roc-umbrella-grafana* which allocates 66 MB. The other components present values below 40 MB and in some cases even below 10 MB.

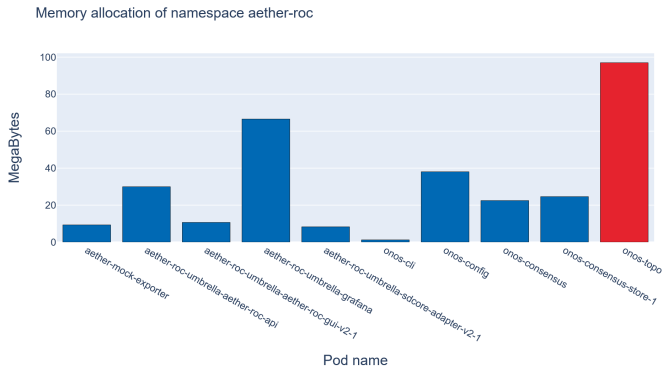


Figure 11.6: Memory allocated by the ROC.

11.1.2 5gcnl-osm-20 experiments

Methodology

To evaluate the performances of OSM, we re-designed the Python program used for the collection of the metrics on the *5gcnl-oran* machine. It always collects metrics using the *metrics-server* add-on, but it also performs other operations. Below is described its workflow:

1. It collects CPU and memory metrics every five seconds for ten minutes (in this phase no network services are deployed).

2. After these first ten minutes, the program starts a timer to calculate the time needed by OSM to on-board a network service (during the

test we deployed ten different network services). It then uploads the VNFD and the NSD of the network service and finally deploys it. The timer stops when the network service is fully functional.

3. It then recollects CPU and memory metrics every five seconds for ten minutes.

4. It performs point 2 and point 3 other nine times with different network services. It should be noted that during point 3 the collection of the metrics involves an increasing number of network services.

5. At the end of the test, the results are stored in a `.csv` file.

To upload network service's packages and perform the deployment, the program runs OSM commands via *osmclient*. The `.csv` files are parsed to produce graphs to show the relevant results.

5gcnl-osm-20 CPU performances In this subsection we will show the results obtained from the analysis of the plots regarding the CPU core consumption of OSM with no network services instantiated and with ten network services instantiated.

From the graph in Figure 11.7 it can be seen that during the idle state, with zero NS instantiated, all the Pods have a consumption of CPU cores below the value 0.02. In this phase, the only Pod to have a much higher consumption of CPU cores is the *keystone* Pod which reaches the value 0.08. When the number of instanced NS increases to ten, the situation changes slightly for all Pods except for *mon* and *mongodb*. These two Pods drastically increase their core consumption until they reach 0.47 and 0.1 respectively. The *mon* increases its consumption by a factor of 23.5 and the *mongodb* by a factor of 10.

The Violin Plot in Figure 11.8, shows how the CPU cores consumption values are distributed as the number of network services instantiated increases on the *mon* Pod. The graph is composed by a Box Plot every two network services instantiated and the distribution of all CPU core consumption measurements made on the Pod with that number of network services. From this graph it can be seen that the increase in CPU cores consumption is similar to a logarithmic trend, with a large increase with few network services instantiated and with

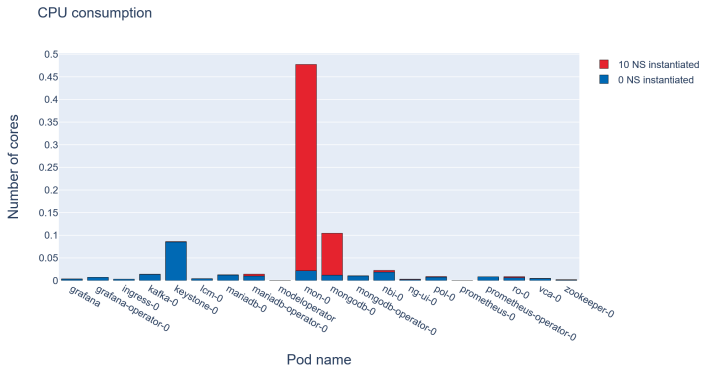


Figure 11.7: Comparison of CPU cores required by OSM with zero and ten network services deployed.

a tendency to stabilize as the number increases. We highlight the presence of outlier values that reduce CPU core consumption to almost zero.

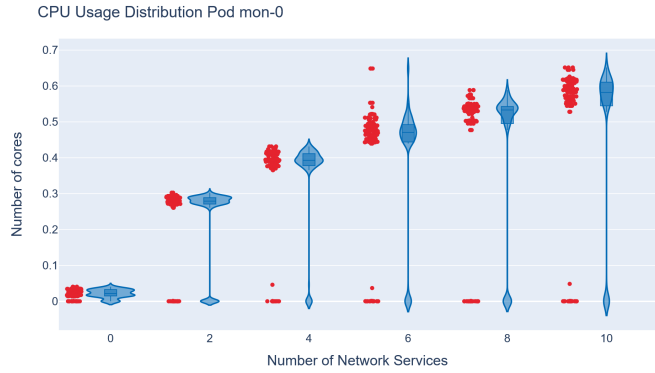


Figure 11.8: Distribution of CPU cores consumption values with increasing number of network services instantiated of the MON module.

Figure 11.9 contains another Violin Plot but referring the *mongodb* Pod. Unlike the previous graph, the outliers in this case peak upwards and not downwards. In this case the distribution of the values over time seems linear

and not logarithmic.

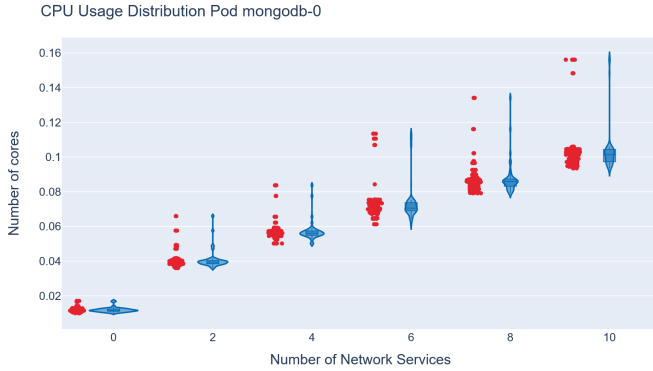


Figure 11.9: Distribution of CPU cores consumption values with increasing number of network services instantiated of the mongodb Pod.

5gcnl-osm-20 memory performances In this subsection we will show the results obtained from the analysis of the plots regarding the memory allocation of OSM with no network services instantiated and with ten network services instantiated.

In Figure 11.10 the differences between zero or ten NSs instantiated are low for most the Pods. The highest peak is generated by the Pod *mongodb* which reaches 1.4 GB with zero instantiated NSs and then up to 1.45 GB with ten NSs instantiated. All other Pods have values less than or equal to 250 MB except for *kafka* (500 MB) and *mon*. Also in this case the Pod *mon* shows a more significant increase compared to all the other modules going from 260 MB to 340 MB.

11.2 Assurance in 5G networks

In this section we show how several 5G infrastructure properties of different types can be verified using the methodology described in Section 3.3.

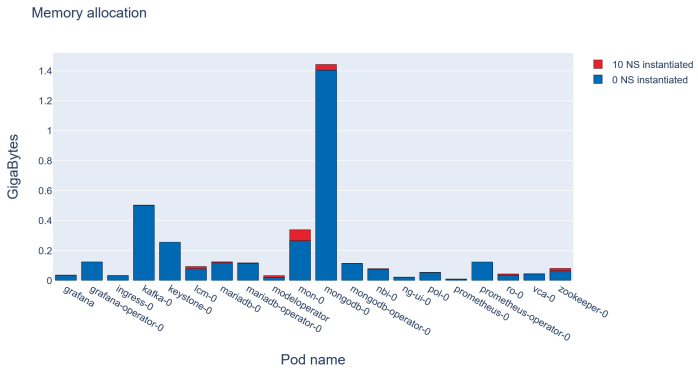


Figure 11.10: Comparison of memory allocated by OSM with zero and ten network services deployed.

11.2.1 Network connection availability

Property p_1 describes the ability of the infrastructure to provide network functionalities to the connected devices and the services being hosted. This property is fundamental for the correct operation of the 5G infrastructure and its continuous verification is of particular interest to both the users and the edge provider.

Metrics As the property (p_1) references multiple functionalities (f_1, f_2, f_3, f_4, f_5), one for each level of connectivity, we have to consider various metrics. In this case, we assume to have a metric $net\_avail(t, from, to)$ specific for each functionality, allowing us to test whether the communication is possible between $from$ and to and returning upload/download speeds and latency measurements for the time instant t.

Contract This property can be verified using a parametric contract that takes in consideration the values obtained during the metrics evaluation phase and compares them to user-expected parameters. E.g., if we only consider the expectancy of a user for mobile connection to Internet, we can hypothesize the user to want a minimum link speed of 25 mb/s from its UE to the Internet in both directions and a maximum introduced latency of 10 ms. Therefore,

the contract takes the form of Equation 11.1.

$$
\begin{aligned}
&net\_avail\_network(t) = let \\
&\quad ue\_dn = net\_avail(t, ue, dn); \\
&\quad ue\_nf = net\_avail(t, ue, nf); \\
&\quad nf\_dn = net\_avail(t, nf, dn); \\
&\quad sp = 100mb/s; \\
&\quad la = 100ms; \\
&in \bigwedge_{res \in \{ue\_dn, ue\_nf, nf\_dn\}} \\
&\quad res.downlink > sp \ \wedge res.uplink > sp \ \wedge res.latency < la
\end{aligned}
\tag{11.1}
$$

A more advanced version of this contract can include checks on network performance of the infrastructure from the point of view of the deployed services, giving users the ability to specify the $from$ and to parameters. Furthermore, we can extend this contract to time ranges, requiring all checks to be valid for each measurement collected at fixed intervals in the interval $[t_1, t_2]$.

11.2.2 Network latency performance

A network latency performance property (p_4) is awarded when all network communications can be achieved within a given time threshold (f_7, f_9). The property is, therefore, parametric, leaving the user the ability to define the maximum expected latency.

Metrics The property in question requires all network communications to be achievable under a certain time threshold. Is therefore necessary to have a measure of latency for each point in the network, regardless of which input and output. We can reuse the same $net\_avail(t, from, to)$ function used in Section 11.2.1, extracting the latency argument of the results.

Contract We need to check whether the worst case latency measurement is under the expected value provided by the user. To do this, we can compare the parameter with double the worst scoring measurement for the instant t

as shown in Equation 11.2.

$$
\begin{aligned}
&net\_avail\_network(t, latency, hops) = let \\
&\quad ue\_dn = net\_avail(t, ue, dn); \\
&\quad ue\_nf = net\_avail(t, ue, nf); \\
&\quad nf\_dn = net\_avail(t, nf, dn); \\
&\quad worst\_lat = \max(ue\_dn.latency, ue\_nf.latency, nf\_dn.latency); \\
&\quad worst\_hops = \max(ue\_dn.hops, ue\_nf.hops, nf\_dn.hops); \\
&in \\
&\quad worst\_lat \times 2 \leqslant latency \ \wedge worst\_hops \times 2 \leqslant hops
\end{aligned}
\tag{11.2}
$$

11.2.3 Network management automation

The network management automation (p_8) is the property of being able to self-manage the network resources, reacting to network intents (f_{18}, f_{19}). We can expect this system to have this property if its resource management control cycle uses a fair allocation of resources.

Metrics To evaluate this property we need a metric that lists the current hard-requirements for network resources $net\_hreq(t, req)$ and a metric that maps each network request with the actually provided resources $net\_sres(t, req)$. Here we assume that network resource requests have a unique identifier $req \in REQ$.

Contract To evaluate the network management automation quality of the system, we check whether all hard requirements are met at instant t as shown in Equation 11.3.

$$
net\_mgmt\_auto(t) = \bigwedge_{req \in REQ} net\_hreq(t, req) \leqslant net\_sres(t, req) \tag{11.3}
$$

11.2.4 Storage confidentiality

The storage confidentiality property (p_{10}) indicates the impossibility of unauthorized access to a data resource. There are many ways of implementing this type of checks depending on the level of detail required [115,116], but for ease of explanation we consider basic user-level access to resource.

Metrics We adopt a metric $has\_access(t, u, r)$ that returns $true$ or $false$ depending on whether the user u has access to the resource r at the instant t.

Contract We verify the property by checking that for each resource r nobody other than the owner of r has access like in Equation 11.4.

$$storage\_conf(t) = \bigwedge_{r \in resources,\ u' \in (users \setminus \{u\})} has\_access(t, u', r) == false \tag{11.4}$$

11.2.5 Experimental evaluation

The experimental evaluation of 5G assurance contracts is a complex process that involves numerous layers of abstraction. We use a 5G simulator based on the Open Networking Foundation's Aether[1], which is deployed in a Kubernetes environment. The simulator consists of the 5GS core network microservices, SD-Core, which is based on the Free5GC[2] project, a virtualised O-RAN compliant gNB, SD-RAN, which simulates the radio part of the mobile network, and Aether ROC, which coordinates the network components and automates their configuration and resource management.

While some KPIs can be extracted from the execution environment, such as those related to network connections and traffic or compute resource usage, this treats the 5G components as black boxes and lacks transparency into their internal state. One of the next research steps on this topic is to extend the Free5GC microservices with monitoring capabilities using OpenTelemetry.

[1] `https://opennetworking.org/aether/`
[2] `free5gc.org/`

This will allow us to expose meaningful monitoring endpoints on the internal state of the microservices.

The monitoring data is fetched by an OpenTelemetry collector, such as Jaeger[3], which provides long-term storage of the measurements. An AA, the assurance component managed by an AL which executes the verification process, receives a verification task. It then begins to assess the relevant contract by retrieving data from the collector and generating a Boolean value from the contract function. The outcome is then incorporated into a verification report and returned as the task output.

The monitoring endpoints are currently unavailable in the implementation, thus the experimentation is temporarily halted. We aim to expand the software to verify this approach in our forthcoming research.

11.3 Assurance and Certification for CDN networks

We experimentally evaluated our certification methodology in a simulated ICN for NFPs CS availability, host availability, and network availability. To evaluate the performance of our certification process, we first defined the certification contracts for the target properties (Section 11.3.1); we then evaluated the performance of the contract verification process (Section 11.3.2) and the network bandwidth consumed by the entire certification process execution (Section 11.3.3). We executed our experiments using the *Criterion* framework for the *Rust* programming language, repeating all tests at least 100 times and until the confidence on the measure is higher than 95%. All tests have been run on Linux with kernel 5.10.78 using an AMD 5900x processor and 32 GB of RAM.

11.3.1 Certification Contracts

We defined a set of contracts modelling NFPs: CS availability, host availability, and network availability in a ICN network node as follows.

[3] `https://www.jaegertracing.io`

CS optimality It verifies the correct configuration of the CS and its operational status. It contains the following rules:

- *CSMemoryusageUB* checks whether the CS is lower than 60% to prevent memory starvation;
- *CSContract* checks whether the CS selected contract is Least Recently Used (LRU);
- *CSUsageLB* checks whether the CS has at at least 1% utilization to verify that it is enabled and operational.

Execution optimality It verifies whether the network node has enough resources to operate correctly and avoid starvation. It contains the following rules:

- *FreeMemory* verifies whether the node has at least 200 MB of memory as a minimal system requirement;
- *NodeLoadUB* checks whether the node CPU delayed load is higher than 90% as an indicator of over-utilization.

Regular network traffic It analyses the recent network traffic looking for anomalies regarding the packet size and name components. It contains the following rules:

- *PITDataMinSizeLB* defines a lower bound of 10 B to the forwarded data packets to detect possible pollution attacks attempts;
- *PITInterestMinComponentsLB* and *PITInterestMinComponentsUB* identify a range of valid values between 3 and 12 for the average number of name components in the forwarded interest packets;
- *PITDataAvgComponentsLB* and *PITDataAvgComponentsUB* define a range of valid values between 3 and 12 for the average number of name components in the forwarded data packets;
- *PITPendingInterestUB* sets an upper bound of 100 pending interest

packets stored in the Pending Interest Table (PIT);

- *PITInterestMinSizeLB* defines a lower bound of 5 B to the minimum size of the forwarded interest packets.

Healthy node It combines using a Boolean AND the three previous contracts.

11.3.2 Contract Verification Process Performance

Given our implementation of the contract verification process, its asymptotic complexity is estimated as $\mathcal{O}(|\mathbf{n}| \cdot |\mathbf{r}|)$ with $|\mathbf{n}|$ being the number of nodes and $\mathbf{r}$ the number of rules included in the contract. We empirically verified this behaviour by measuring its execution time varying the complexity of the target contract and the number of target network nodes. To exclude any delay or interference due to network interactions, we simulated local executions only, providing precached measurements for each network node.

Figure 11.11 shows the execution time varying the number of target network nodes from one to 1000 in the 4 contracts. The execution time grew linearly with the number of nodes for all the contracts. Contract healthy node has lowest performance being a combination of the other three contracts. Difference between contracts CS optimality, execution optimality, regular network traffic depend on the performance of the specific rules composing them.

We then measured the impact of the contract complexity on the execution time in terms of rules by comparing the results obtained with a fixed number of nodes and a given time interval. We repeated the tests comparing the execution time of the 4 contracts using 500 target nodes. The average execution time for the 4 contracts are 16 ms for *CS optimality*, 27 ms for *Execution optimality*, 11 ms for *Regular network traffic* and 54 ms for *Healthy node*. Our results show how contract Healthy node has an evaluation time close to the sum of the execution time of the single contracts. We observe that the growth is linear with the number of evaluated rules, as expected.

Although the asymptotic complexity seems to be reflected in our experiments, it refers to the worst case, where i) the measurements cannot be

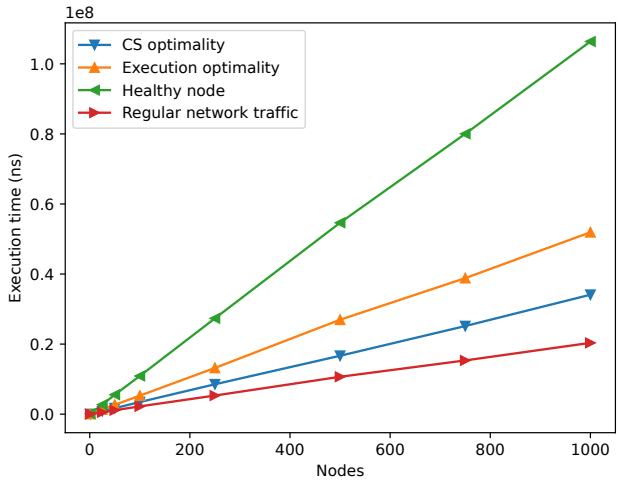


Figure 11.11: Execution time for the 4 contracts varying the number of verified nodes.

shared between one or multiple rules, ii) no caching of the previously evaluated contracts is allowed. If we consider sharing and caching, the asymptotic complexity is reduced to a logarithmic growth. Considering the caching abilities of both the AL and the network, a more fair asymptotic complexity estimation is $\mathcal{O}(\log(|\mathbf{n}|) \cdot \log(|\mathbf{r}|))$.[4]

11.3.3 Network Usage

We evaluated the network bandwidth used by the services during the certification process. The maximum size of each network packet is generally limited by the ICN protocol, 4 KB packets in NDN. We then used an upper bound to the number of network requests to estimate the total traffic. We expect the total number of requests to complete a single certification to be lower than $\mathcal{O}(|\mathbf{n}| \cdot |\mathbf{r}| \cdot |\mathbf{m}|)$, where $|\mathbf{n}|$ is the number of network nodes, $|\mathbf{r}|$ is the number of rules in the contract, and $|\mathbf{m}|$ is the number of metrics. Assuming the total

[4] Additional improvements can be obtained by parallelizing the execution of metric and rule evaluation, first inspecting their dependencies.

number of metrics is limited and lower than the number of rules, with rules reusing the same measurements, we can simplify the previous asymptotic complexity to $\mathcal{O}(|\mathbf{n}| \cdot |\mathbf{m}|)$.

Figure 11.12 shows the relationship between the number of evaluations and the number of metrics evaluation requests sent during a certification process in the 4 contracts in Section 6.3.4. Our results show a linear correlation between evaluations and requests. The evaluation to requests ratio is specific to the contract definition and varies from 2:1 ratio, with contract *Execution optimality*, up to 10:1 ratio, with contract *Healthy node*.

The contract verification service in the decentralized certification process produces additional traffic in the form of contract verification requests, contract query requests, and certificates retrieval. While in the first two cases their number is constant, the number of certificates retrieval requests grows linearly with the number of nodes and rules. A single certificate contains information about multiple nodes and multiple rules, therefore the actual ratio follows a logarithmic growth.

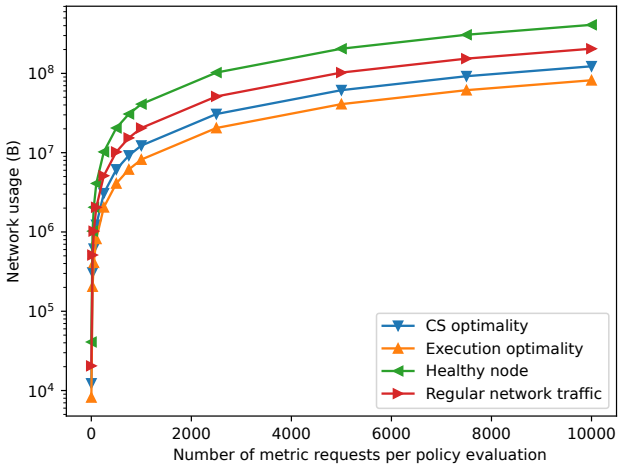


Figure 11.12: Number of metrics requests per contract evaluation.

Again, this experiment had been carried out in the worst case scenario, where

none of the requests are locally cached in the AL. When caching at both AL and ICN network nodes are considered, asymptotic complexity is reduced to $\mathcal{O}(\log(n) \cdot \log(m))$. In a more efficient implementation, measurements and rule evaluations can be retrieved once and shared by multiple rules, resulting in a drastic decrease in the total number of requests. We note that thanks to the ICN network caching capabilities, these request are more likely to be resolved by the caches of one of the network nodes in the request path before reaching their target node.

11.4 Assurance in Big Data Analysis Platforms

We experimentally evaluated the utility of our approach with a complete detailed walkthrough applied to our real scenario in Section 7.3 with the scope of evaluating its trustworthiness and improve its transparency. In this work we modelled trustworthiness using our overall assurance confidence level in Definition 7.4.4 and approximated the concept of transparency as the level of inspectability, that is, the number of probes executed to inspect the given system divided by the total number of available probes that can be applied. We note that we favoured simplicity over complexity in defining trustworthiness and transparency, to the aim of demonstrating the soundness of our assurance approach, the first applied to big data pipelines. We finally discuss the feasibility of our solution in terms of computational effort, considering different evaluation scenarios.

11.4.1 Experimental setup

Our experimental setup is a portion of the H2020 EVOTION platform hosted on our premises including more than 100 tasks, 20 analytics templates, and 30 analytics instances. Pipelines and tasks are implemented in Spark 2.3 and Spark 2.2 using Python or Java, and when needed using Spark mllib. Our H2020 EVOTION platform is grounded on the following ecosystem: Linux 5.16.19 (NixOS), Apache Hadoop 3.3.1, Apache Spark 3.2.1, Apache Airflow 2.2.4. The platform is deployed on a 'bare-metal' infrastructure using an AMD 5900x and 32 GB of RAM. The service configurations and the pipeline implementations are available at `bit.ly/3uy1Op4`. Our Assurance

architecture in Section 7.1.2 is deployed on top of the H2020 EVOTION Big Data engine, to exploit the parallelisation capabilities and synchronize the execution of assurance evaluation activities with the instance triggering.

11.4.2 Assurance Evaluation Walkthrough

We present a detailed walkthrough of our methodology based on an extended version of our running example in Section 7.3, where the client asks for both confidentiality in transit and at rest. Table 11.3 shows our walkthrough scenario including the template $\Pi^{\lambda,\gamma}$ and instance $I^{\theta,\psi}$ annotations.

The walkthrough is organized as follows. First, we present the execution of our assurance methodology (separating pipeline tasks from ecosystem services), including details on the assessment probes (P), the evaluation values (E), and the assurance confidence levels ($A_{\tau,r}$). Second, we present the process to compute the final overall assurance aggregating the assurance confidence levels at requirements level.

Pipeline Assurance

Let us start considering the three annotations r_1^θ, r_2^θ and r_3^θ that are common to all the tasks $\hat{t}_i \in \hat{p}$. The assessment processes are implemented with the same set of probes P_1, P_2, and P_3 for r_1^θ, P_4 for r_2^θ, and P_5 and P_6 for r_3^θ. We recall that probe types are defined in Table 7.2.

$P_1(r_1^\theta, *)$ is a Lineage probe focused on verifying that the spark job implementing the tasks $\hat{t}_i$ is writing data on the HDFS offered by the service $\hat{s}_1 \in \hat{e}$ only exploring the Spark DAG. It is focused on verifying the compliance to a given Spark writing pattern (i.e., *expected_dag*) and explores the spark DAG using the spark log.

$P_2(r_1^\theta, *)$ is a Code Inspection probe focused on verifying connection to external data storage observable within the source code. It is focused on finding specific path patterns (i.e., *expected_paths*) within a task source code.

$P_3(r_1^\theta, *)$ is a Parameters Check probe focused on verifying if parameters

Table 11.3: Walkthrough scenario derived from the running example in Section 7.3.

$\mathcal{R}$	**Description**
	Template requirements
r_1^{λ}	*Confidentiality at rest and in transit* for all the t in p
r_2^{λ}	*Authorization* for the $fileSystem$ task t in p
r_1^{γ}	*Confidentiality at rest and in transit* for all the s in e
	Pipeline Instance requirements derived from r_1^{λ}
r_1^{θ}	*No temporarily unprotected data storage* for all the $\hat{t} \in \hat{p}$
r_2^{θ}	*Avoid connection to external services* for all the $\hat{t} \in \hat{p}$
r_3^{θ}	*Avoid use of vulnerable code/libraries* for all the $\hat{t} \in \hat{p}$
r_4^{θ}	*Pipeline integrity* checking the correct ordering of tasks $\hat{t} \in \hat{p}$
	Pipeline Instance requirements derived from r_2^{λ}
r_5^{θ}	*Check Authorization* for ingestion task $\hat{t}_1 \in \hat{p}$
r_6^{θ}	*Check ownerships at rest* for visualization tasks $\hat{t}_4 \in \hat{p}$
	Ecosystem Instance requirements derived from r_1^{γ}
r_1^{ψ}	*Encrypted HDFS* for the $\hat{s}_1 \in \hat{e}$
r_2^{ψ}	*Inter-node communication security* for the $\hat{s}_1$ and $\hat{s}_2 \in \hat{e}$
r_3^{ψ}	*Orchestrator Confidentiality* for the $\hat{s}_3 \in \hat{e}$
r_4^{ψ}	*Communication channel security* for the $\hat{s}_4 \in \hat{e}$
r_5^{ψ}	*Authentication-enabled* for the $\hat{s}_4 \in \hat{e}$
r_6^{ψ}	*Authorization policies-enabled* for the $\hat{s}_5 \in \hat{e}$
r_7^{ψ}	*Vulnerability check* for all the services $\hat{s} \in \hat{e}$

for writing at rest (if any) are referring to the HDFS offered by the service $\hat{s}_1 \in \hat{e}$ or not. It is based on controlling the actual parameters of a given task and match them against expected parameters/patterns (in this case writing URLs)

$P_4(r_2^{\theta}, *)$ is a Code Inspection probe focused on verifying external network connections used for setting up data traffic. It is based on checks for connections to external sources (if any).

$P_5(r_3^{\theta}, *)$ is a Code Vulnerability Check probe focused on finding vulnerable

code used in the spark task $\hat{t}_i$. It is based on Pylint as tool for Python vulnerability check.

$P_6(r_3^\theta, *)$ is a Dependency Check probe (see Table 11.6) focused in finding vulnerable libraries used in the spark task $\hat{t}_i$. It is based on analysis of the "requirements.txt" file.

Let us consider r_4^θ, it requests to verify absence of hidden tasks or a wrong sequence of task in the pipeline compared to the one declared. It is implemented with an Orchestration Check $P_7(r_4^\theta, \hat{p})$ targeting the Python airflow description of pipeline DAG and verifying the ordering and presence of additional tasks compared to what is expected.

Let us consider r_5^θ it is annotated on the ingestion task $\hat{t}_1$ only. The assessment process is implemented with the probe $P_8(r_5^\theta, \hat{t}_1)$ realized via Testing Probe verifying that the users executing the pipeline against ownership of the source of data ingested in order to check that the pipeline has the right to carry out the ingestion.

Considering r_6^θ it is annotated on the visualization task $\hat{t}_4$ only. The assessment process is implemented with the probe $P_9(r_6^\theta, \hat{t}_4)$ realized via Testing Probe focused on the ownership at rest in order to verify that the pipeline write the data preserving the original ownership of the pipeline executor avoiding confidentiality infringement.[5]

Table 11.4 summarizes the results of the execution of our methodology in the walkthrough scenario in Table 11.3. It presents the pipeline tasks $\hat{t}$, annotated requirements $\mathcal{R}$, and the assessment probes (P) used to evaluate them. It also presents the retrieved results in terms of evaluation value (E) and assurance confidence level (A).

Probes P_1 to P_3 and P_6 to P_9 did not identify any significant issue in the pipeline, regardless the target, resulting in an optimal evaluation of 1.0. On the other hand, $P_4(r_2^\theta, \hat{t}_4)$, for target $\hat{t}_4$ and requirement r_2^θ, and P_5, for every target and requirement, identified the following issues. $P_4(r_2^\theta, \hat{t}_4)$ identified an external networking connection within $\hat{t}_4$ resulting in $E = 0$, meaning that r_2^θ is not supported by $\hat{t}_4$ according to P_4. P_5 identified several warnings and

[5] Current Big Data engines may lead to issues of data ownerships in a multi-tenant context [117, 118].

coding convention violations (e.g., too lengthy lines and naming convention violations), resulting in a set of evidence evaluated with $E = 0.75$ by our process. It means that some weaknesses are identified but none of them are major according to requirement r_3^θ. The assurance confidence levels (A), represented in the last column of Table 11.4, are calculated according to Definition 7.1.4. We note that the lowest assurance confidence level (0.66) was computed for task $\hat{t}_4$ according to requirement r_2^θ; it was due to the aforementioned issue discovered by the assessment probe $P_4(r_2^\theta, \hat{t}_4)$.

Table 11.6 shows the pseudocode of the pipeline probes from P_1 to P_9 aggregated according to their types as defined in Table 7.2.

Table 11.4: Assurance evaluation results for pipeline $\hat{p}$ and tasks $\hat{t}$ considering the requirements $\mathcal{R}$ of our walkthrough scenario in Table 11.3. Assurance probes $P(r, \tau)$, evaluations $E(EV, r)$ and assurance confidence levels $A_{\tau,r}$ are presented.

Pipeline tasks $\hat{t} \in \hat{T}$

$\hat{t}$	$\mathcal{R}$	$P(r,\tau)$	$E(EV,r)$	$A_{\tau,r}$
$\hat{t}_1$	r_1^θ	$P_1(r_1^\theta, \hat{t}_1)$	$[1.0]$	1.0
	r_2^θ	$P_2(r_2^\theta, \hat{t}_1), P_3(r_2^\theta, \hat{t}_1), P_4(r_2^\theta, \hat{t}_1)$	$[1.0, 1.0, 1.0]$	1.0
	r_3^θ	$P_5(r_3^\theta, \hat{t}_1), P_6(r_3^\theta, \hat{t}_1)$	$[0.75, 1.0]$	0.88
	r_5^θ	$P_8(r_5^\theta, \hat{t}_1)$	$[1.0]$	1.0
$\hat{t}_2$	r_1^θ	$P_1(r_1^\theta, \hat{t}_2)$	$[1.0]$	1.0
	r_2^θ	$P_2(r_2^\theta, \hat{t}_2), P_3(r_2^\theta, \hat{t}_2), P_4(r_2^\theta, \hat{t}_2)$	$[1.0, 1.0, 1.0]$	1.0
	r_3^θ	$P_5(r_3^\theta, \hat{t}_2), P_6(r_3^\theta, \hat{t}_1)$	$[0.75, 1.0]$	0.88
$\hat{t}_3$	r_1^θ	$P_1(r_1^\theta, \hat{t}_3)$	$[1.0]$	1.0
	r_2^θ	$P_2(r_2^\theta, \hat{t}_3), P_3(r_2^\theta, \hat{t}_3), P_4(r_2^\theta, \hat{t}_3)$	$[1.0, 1.0, 1.0]$	1.0
	r_3^θ	$P_5(r_3^\theta, \hat{t}_3), P_6(r_3^\theta, \hat{t}_1)$	$[0.75, 1.0]$	0.88
$\hat{t}_4$	r_1^θ	$P_1(r_1^\theta, \hat{t}_4)$	$[1.0]$	1.0
	r_2^θ	$P_2(r_2^\theta, \hat{t}_4), P_3(r_2^\theta, \hat{t}_4), P_4(r_2^\theta, \hat{t}_4)$	$[1.0, 1.0, 0.0]$	0.66
	r_3^θ	$P_5(r_3^\theta, \hat{t}_4), P_6(r_3^\theta, \hat{t}_1)$	$[0.75, 1.0]$	0.88
	r_6^θ	$P_9(r_6^\theta, \hat{t}_4)$	$[1.0]$	1.0
$\hat{p}$	r_4^θ	$P_7(r_4^\theta, \hat{p})$	$[1.0]$	1.0

Ecosystem Assurance

Table 11.5: Assurance evaluation results for ecosystem services $\hat{s} \in \hat{S}$ considering the requirements $\mathcal{R}$ of our scenario in Table 11.3. Assurance probes $P(r, \tau)$, evaluations $E(EV, r)$ and assurance levels $A_{\tau,r}$ are presented.

Ecosystem services $\hat{s} \in \hat{S}$

$\hat{s}$	$\mathcal{R}$	$P(r,\tau)$	$E(EV,r)$	$A_{\tau,r}$
$\hat{s}_1$	r_1^ψ	$P_{10}(r_1^\psi, \hat{s}_1)$	[0.1]	0.1
	r_2^ψ	$P_{11}(r_2^\psi, \hat{s}_1)$	[0.1]	0.1
$\hat{s}_2$	r_2^ψ	$P_{12}(r_2^\psi, \hat{s}_2)$	[0.1]	0.1
$\hat{s}_3$	r_4^ψ	$P_{13}(r_4^\psi, \hat{s}_3)$	[0.1]	0.1
$\hat{s}_4$	r_4^ψ	$P_{14}(r_4^\psi, \hat{s}_4)$	[1.0]	1.0
	r_5^ψ	$P_{15}(r_5^\psi, \hat{s}_4)$	[1.0]	1.0
$\hat{s}_5$	r_6^ψ	$P_{16}(r_6^\psi, \hat{s}_5)$	[1.0]	1.0
$\hat{s}_5$	r_7^ψ	$P_{17}(r_7^\psi, \hat{s}_5)$	[0.57]	0.57

Let us start considering $\hat{s}_1$ (i.e., Hadoop HDFS) used for ingestion and visualization. It is associated with requirements r_1^ψ and r_2^ψ. In both cases, to implement the relative assessment processes, Configuration Check probes $P_{10}(r_1^\psi, \hat{s}_1)$ and $P_{11}(r_2^\psi, \hat{s}_1)$ are used. Probe P_{10} is focused on verifying whether the HDFS is configured to be encrypted, while probe P_{11} is focused on verifying that secure channels are active between the different storing nodes of the HDFS. Considering $\hat{s}_2$ (i.e., Spark) and $\hat{s}_3$ (i.e., Airflow), which are services used to execute the pipeline, they are both associated with requirements r_2^ψ and r_3^ψ, respectively. The relative assessment processes are implemented using specific Configuration Check probes $P_{12}(r_2^\psi, \hat{s}_2)$ and $P_{13}(r_3^\psi, \hat{s}_3)$ aimed to verify specific security features of Spark for internode communication and Airflow security, respectively.

$\hat{s}_4$ (i.e., Knox), which is used to provide authentication mechanisms, it is associated with requirements r_4^ψ and r_5^ψ. The corresponding assessment processes are implemented using Testing probe $P_{14}(r_4^\psi, \hat{s}_4)$ to test the security of the channel used to carry out authentication (Kerberos) and a Configuration

Table 11.6: Pipeline probe scripts: Pseudocode.

Lineage

Probes 1: $P_1(r_1^\theta, *)$

```
def p1(expected_dag, app_id):
    evidence.logs = get_spark_logs(app_id)
    for job in evidence.logs.jobs:
        if job not in expected_dag:
            evidence.warnings.push(f"Unexpected job {job}")
    return evidence
```

Parameters Check

Probe 3: $P_3(r_1^\theta, *)$

```
def p3(expected_params, task_id):
    params = get_params(task_id)
    evidence.params = params
    for param, exp_param in zip(params, expected_params):
        if not param.matches(exp_params):
            evidence.warnings.push(
                f"Param {param} not matching"
            )
    return evidence
```

Code Vulnerability Check

Probe 5: $P_5(r_3^\theta, *)$

```
def p5(task_id):
    evidence.source_code = get_source(task_id)
    evidence.warnings = call_pylint(source_code)
    return evidence
```

Dependency Check

Probe 6: $P_6(r_3^\theta, *)$

```
def p6(task_id):
    evidence.requirements = get_requirements(task_id)
    output = call_safety(evidence.requirements)
    evidence.warnings = json.loads(output.decode().strip())
    return evidence
```

Orchestration Check

Probe 7: $P_7(r_4^\theta, \hat{p})$

```
def p7(expected_dag):
    dag = get_current_dag()
    evidence.dag = dag
    for task in dag:
        exp_task = expected_dag.find(task)
        if exp_task is None:
            evidence.warnings.push(
                f"Unexpected task {task}"
            )
        if task.deps != exp_task.deps:
            evidence.warnings.push(
                f"Unexpected {task} deps."
            )
    return evidence
```

Code Inspection

Probe 2: $P_2(r_1^\theta, *)$

```
def p2(expected_paths, task_id):
    evidence.source_code = get_source(task_id)
    evidence.detected_paths = get_hdfs_paths(
        evidence.source_code
    )
    for path in evidence.detected_paths:
        if path not in expected_paths:
            evidence.warnings.append(
                f"Unexpected path {path}"
            )
    return evidence
```

Probe 4: $P_4(r_1^\theta, *)$

```
def p4(task_id, expected_services):
    evidence.activity = get_network_activity(task_id)
    evidence.services = parse_services(evidence.activity)
    for service in evidence.services:
        if service not in expected_services:
            evidence.warnings.push(
                f"Unexpected service {service}"
            )
    return evidence
```

Testing

Probe 8: $P_8(r_5^\theta, \hat{t}_1)$

```
def p8(user, file_path):
    try:
        file = access_as_user(user, file_path)
        file.head()
        evidence.output = "File can be ingested"
    except PermissionError:
        evidence.output = "Permission Error"
    except MissingFile:
        evidence.output = "File not found"
    return evidence
```

Probe 9: $P_9(r_6^\theta, \hat{t}_4)$

```
def p9(expected_permissions, user, file_path):
    try:
        file = access_as_user(user, file_path)
        permissions = file.get_permissions()
        evidence.permissions = permissions
        if permissions == expected_permissions:
            evidence.output = "Permissions match"
        else:
            evidence.output = "Invalid permissions"
    except PermissionError:
        evidence.output = "Permission Error"
    except MissingFile:
        evidence.output = "File not found"
    return evidence
```

Check probe $P_{15}(r_5^\psi, \hat{s}_4)$ aimed to verify the authentication configuration.

$\hat{s}_5$ (i.e., Ranger), which is used to authorize users to carry out analytics, is associated with requirements r_6^ψ. The relative assessment processes are implemented using Monitoring probe $P_{16}(r_6^\psi, \hat{s}_5)$ aimed to verify policies triggered by Ranger.

According to requirement r_7^{ψ}, for every service $\hat{s} \in \hat{e}$, a vulnerability check needs to be executed. The assessment process is implemented as a Vulnerability Check probe $P_{17}(r_7^{\psi}, *)$ aimed to find vulnerabilities that can be relevant for requirement r_7^{ψ} on all services $\hat{s} \in \hat{e}$.

Table 11.5 summarizes the results of the execution of our methodology in the walkthrough scenario in Table 11.3. It presents the ecosystem services $\hat{s}$, annotated requirements $\mathcal{R}$, and the assessment probes (P) used to evaluate them. It also presents the retrieved results in terms of evaluation value (E) and assurance confidence level (A).

Each probe in the Table 11.5 produced a single value as evaluation E and thus the same value is also reported as the assurance confidence level A. More than half of them obtained a low assurance confidence level ($A = 0.1$) due to the suboptimal configurations adopted by the relative services $\hat{s}$. $P_{10}(r_1^{\psi}, \hat{s}_1)$ warned about a misconfiguration over the in-transit data encryption, while $P_{11}(r_2^{\psi}, \hat{s}_1)$ signalled that the registry security configuration was not enabled. $P_{12}(r_2^{\psi}, \hat{s}_2)$ detected that network encryption was not enabled for Spark, while $P_{13}(r_4^{\psi}, \hat{s}_3)$ highlighted the encryption fernet key being unset. $P_{17}(r_7^{\psi}, \hat{s}_5)$ found several registered CVEs of various severity, the worst of which had a score of 4.3, resulting in an assurance confidence level of $A = 0.57$. The remaining probes did not identify significant issues and returned an optimal value $A = 1$. Table 11.7 shows the pseudocode of the different service ecosystem probes dividing them into the different probe types as defined in Table 7.2.

Overall Assurance Evaluation

The final assurance evaluation was carried out according to Definition 7.4.4 leading to the following overall assurance confidence levels $L_{*,r_1^{\lambda}} = 0.66$, $L_{*,r_2^{\lambda}} = 1$ and $L_{*,r_1^{\gamma}} = 0.1$ aggregated on the relevant targets $*$. We note that the overall assurance confidence level for r_1^{λ} *Confidentiality at rest and in transit* is impacted negatively by the assurance evaluation on r_2^{θ} related to *Avoid connection to external services* carried out by P_4 probe on $\hat{t}_4 = saveToHDFS(model)$. We also note that the ecosystem of services clearly shows the lowest assurance confidence level due to severe configuration issues for $\hat{s}_1$, $\hat{s}_2$, and $\hat{s}_3$ retrieved by the relative probes.

Table 11.7: Service ecosystem probe scripts: Pseudocode.

Configuration check

Probe 10: $P_{10}(r_1^{\psi}, \hat{s}_1)$

```
def p10(config_endpoint):
    config = get_hadoop_configuration(config_endpoint)
    evidence.config = config
    warnings = []
    if config["dfs.encrypt.data.transfer"] == "false":
        warnings.append(
            "In-transit data encryption is disabled"
        )
    if (
        config["yarn.intermediate-data-encryption.enable"]
        == "false"
    ):
        warnings.append(
            "Intermediate data encryption is disabled"
        )
    # ...
    evidence.warnings = warnings
    return evidence
```

Probe 11: $P_{11}(r_2^{\psi}, \hat{s}_1)$

```
def p11(config_endpoint):
    config = get_hadoop_configuration(config_endpoint)
    evidence.config = config
    warnings = []
    if config["dfs.permissions.enabled"] == "false":
        warnings.append("FS access control is disabled")
    if (
        config["dfs.permissions.superusergroup"]
        == "supergroup"
    ):
        warnings.append("Task has unrestricted permissions")
    if config["hadoop.registry.secure"] == "false":
        warnings.append("Registry security is not enabled")
    if config["hadoop.security.authorization"] == "false":
        warnings.append("Authentication is disabled")
    # ...
    evidence.warnings = warnings
    return evidence
```

Probe 12: $P_{12}(r_2^{\psi}, \hat{s}_2)$

```
def p12(app_id, api_endpoint):
    config = get_spark_config(app_id, api_endpoint)
    evidence.config = config
    warnings = []
    if config["spark.network.crypto.enabled"] != true:
        warnings.append("Network encryption disabled")
    if config["spark.io.encryption.enabled"] != true:
        warnings.append("IO encryption disabled")
    # ...
    evidence.warnings = warnings
    return evidence
```

Probe 13: $P_{13}(r_3^{\psi}, \hat{s}_3)$

```
def p13(api_endpoint):
    config = get_airflow_config(api_endpoint)
    evidence.config
    warnings = []
    if config.core.get("fernet_key") is None:
        warnings.append("Fernet key is not set")
    if config.kubernetes.verify_ssl is False:
        warnings.append("SSL cert. check disabled")
    # ...
    evidence.warnings = warnings
    return evidence
```

Probe 15: $P_{15}(r_5^{\psi}, \hat{s}_4)$

```
def p15(exp_user_perms, api_endpoint):
    for user, exp_perms in exp_user_perms.items():
        perms = get_perms(user, api_endpoint)
        evidence.permissions[user] = perms
        if not exp_perms.matches(perms):
            evidence.warnings.append(
                f"unexpected perms. {perms} for {user}"
            )
    return evidence
```

Monitoring

Probe 16: $P_{16}(r_6^{\psi}, \hat{s}_5)$

```
def p16(expected_policies, api_endpoint):
    logs = get_ranger_logs()
    evidence.logs = logs
    warnings = []
    for event in logs:
        if not event.matches(expected_policies):
            warnings.append(f"Unexpected event {event}")
    evidence.warnings = warnings
    return evidence
```

Testing

Probe 14: $P_{14}(r_4^{\psi}, \hat{s}_4)$

```
def p14(auth_config, api_endpoint):
    result = authenticate(auth_config, api_endpoint)
    if result.failed():
        evidence.output = "Authentication failed"
    elif result.unauthorized():
        evidence.output = "Not authorized"
    # ...
    return evidence
```

Vulnerability Check

Probe 17: $P_{17}(r_7^{\psi}, *)$

```
def p17(service):
    evidence.warnings = openvas_analysis(service)
    return evidence
```

11.4.3 Performance Evaluation

We evaluate the computational effort requested by our assurance process for the walkthrough in Section 11.4.2 considering the different probe types adopted and different changing scenarios as follows. We note that our assurance solution reuses evidence from related assessment processes in the

framework of computing the final assurance confidence level.

1. Contextual Changes (CC): It includes changes due to the external factors like new version of the probe used in the repository or new discovered vulnerabilities. This scenario requires re-execution of the new probes as well as of the vulnerability-related probes.

2. Pipeline Changes (PC): It includes changes due to different parameters and changes at orchestration level. This scenario requires the re-execution of the task or orchestration probes showing changes only.

3. Ecosystem Changes (EC): It includes different versions/configurations of services at ecosystem level. This scenario requires the re-execution of the ecosystem level probes only.

4. Instance Changes (IC): It includes changes requesting complete assurance re-evaluation.

More specifically, in our walkthrough, we consider i) CC assuming new vulnerabilities discovered for all the services used at ecosystem level, ii) PC assuming new version of k-means modelling task only, iii) EC assuming new version of HDFS service, iv) IC assuming a new task for ingestion based on Hive instead of HDFS. We also consider a scenario where the assurance is re-evaluated without any changes.

Figure 11.13 shows a stacked histogram of the time requested to execute the entire assurance process in the different scenarios considering the different adopted probe types.

The pipeline taken as example is training a k-means model based on the dataset 'digits' from the *scikit-learn*[6] library. The model implementation is provided by the Spark ML library[7]. Finally, the trained model is serialized and saved in an Hadoop storage. The time necessary to execute the pipeline is 54 seconds.

The pipeline probes are executed in the CC, PC and IC scenarios requiring 0.13 seconds, while the service probe are executed in the EC and IC scenarios

[6] `https://scikit-learn.org`
[7] `https://spark.apache.org/mllib`

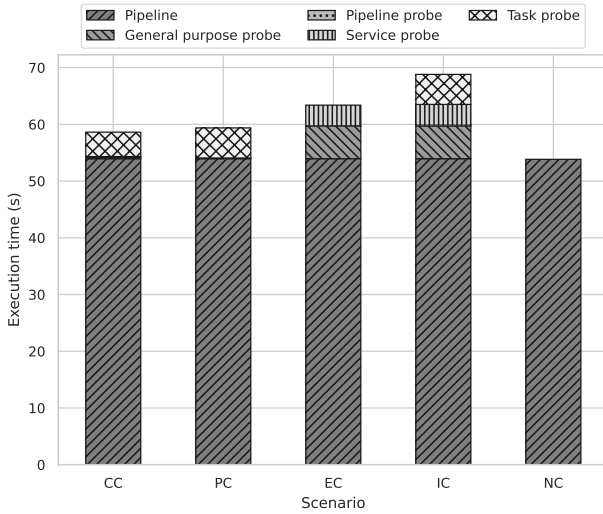


Figure 11.13: Performance on the walkthrough in different scenarios: contextual changes (CC), pipeline changes (PC), ecosystem changes (EC), instance changes (IC) and no changes (NC). Computational time requested by the different probes aggregated by probe types.

in 3.65 seconds. The general purpose probes have a total execution time of 5.79 seconds in scenarios EC and IC, while in CC their evaluation is completed is 0.25 seconds as only part of them has to be re-calculated. Similarly, the task probes total evaluation time is 5.29 seconds in scenarios PC and IC, dropping to 4.30 seconds in CC.

We note that, in this experiment, i) we are considering a sequential execution of the required probes, but parallelisation is possible in most cases, saving a significant amount of time; ii) the time requested for the IC assessment is obviously greater than the others, since it is executing all the probes evaluations; iii) the pipeline probe evaluations are particularly efficient as they only require information already stored in the configuration of the pipeline instance; iv) the total probes evaluation execution time is negligible compared to time required by the pipe in a Big Data context. Concerning the CC and IC scenarios, we note that the vulnerability analysis probe $P_{17}(r_7^{\psi}, *)$

is not included in Figure 11.13 due to its long execution time, approximately 3 minutes, resulting in a strongly unbalanced graph and unreadable bars. We also note that assurance levels coming from other assessment processes can be reused in case the targets and the requirements are the same. This is typically the case of ecosystem services used by different pipelines. Figure 11.13 shows this effect clearly for NC scenario where all the assessment processes are reused from the previous runs and the total requested time is almost the one needed for the pipeline only.

11.5 Assurance Aware Deployment in E2C Continuum

In this section we first describe our experimental settings realizing the scenario in Section 9.1 and then present a preliminary performance evaluation.

11.5.1 Experimental setup

We realized the E2C Continuum of our scenario in Section 9.1 as follows. The software stack used in the Cloud facility was based on MicroK8s. It allowed to simulate cloud-hosted VPS nodes in a cluster formation supporting auto-scaling as we expect by a Cloud provider. The Telco-Edge facility was implemented based on OSM[8], Free5GC[9] and MicroK8s[10], respectively implementing the MEC, a simulated 5G core network, and the resource manager. OSM was designed to integrate with the core network, exposing the services hosted in Kubernetes to the users and other nodes as Network Functions. The on-premises facility was based on MicroK8s, configured to handle limited resources. All the nodes were hosted in the same data centre, and were equipped with 4 virtual cores clocked at 2.09 GHz and 32 GB of RAM and running Linux 5.4.0 (Ubuntu 20.04 LTS). All the nodes of the continuum were empowered by our deployment agents communicating with our Deployment Engine using the same virtualised network used by the continuum node.

[8] `https://osm.etsi.org/`
[9] `https://www.free5gc.org`
[10] `https://microk8s.io`

Our Deployment Engine was implemented using Python 3.10.9 and has been executed on a machine running Linux 5.15.102 (NixOS) equipped with an AMD 5900x and 32 GB of RAM.

11.5.2 Performance evaluation

The computational effort needed to deploy a given service workflow using our Deployment Engine is clearly dominated by the effort required by our matching function that can be estimated as $\mathcal{O}(|F|^{|S|})$ where $|F|$ and $|S|$ are the cardinality of Continuum facilities and services in the workflow respectively.

Figure 11.14 shows the execution time requested by the matching function varying the number of services $s_i \in S$ in the workflow and the number of facilities $f_j \in F$ of the continuum using logarithmic scale. The performances were computed on the average of 5 executions for each combination of services and facilities. For each execution we randomly generated facilities' capabilities and services' requirements. As expected the number of services $|S|$ dominates the exponential growth of execution time.

The matching algorithm terminates in less than one second when evaluating the cases of 6 services in 5 facilities and 8 services in 3 facilities. As the number of services increases, the time of execution grows rapidly: the execution time with 12 services and 3 facilities is 8.11 s, which is the last experiment configuration to terminate under 10s. The cases 8 services in 5 facilities and 10 services in 4 facilities terminate respectively in 12.6 s and 22.1 s. With larger numbers of facilities and services the algorithm becomes too slow for practical use: the configuration 11 services in 4 facilities terminates in 91.7 s. The last configuration shown in Figure 11.14 considers 12 services in 5 facilities and has terminated in 13143 s.

In this work we described a novel methodology for deploying composed services in E2C Continuum focused on guaranteeing advanced QoS requirements. The main idea is to exploit the capabilities and the peculiarities of the different continuum landing platforms to ensure QoS at deployment time. Our deployment methodology is based on a machine-readable description of the service composition annotated with the QoS requirements and a meta-description of the capabilities of the landing platforms involved in the

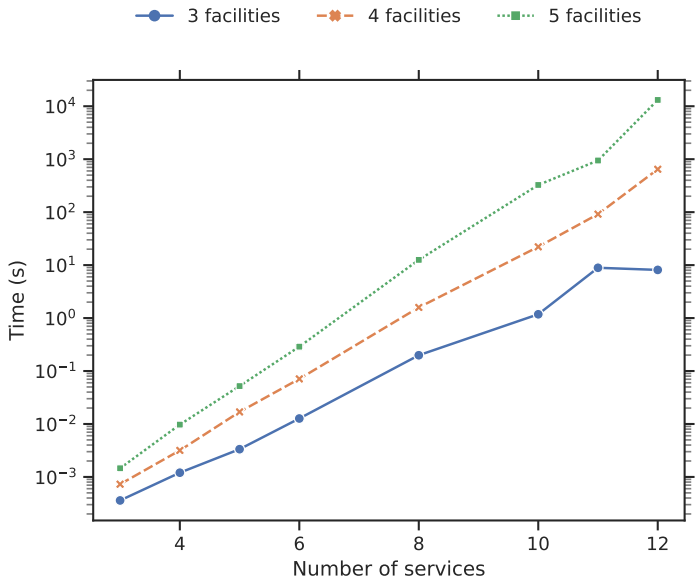


Figure 11.14: Execution time with increasing number of services s_j in the workflow and considering 3, 4 and 5 facilities f_j.

continuum. Based on these machine-readable descriptions, a set of feasible deployment solution are generated and the relative deployment recipes for the service composition are selected according to a given policy. We show the feasibility of our solution in our preliminary experimental evaluation. In our future works, we plan to investigate different deployment optimization solutions and consider a more complex experimental scenario involving migrations of service and data as well as changes at the network topology level.

11.6 MIND foods Hub Big Data Engine

In this section, we report the preliminary results obtained by our Big Data Engine implementation in some of its most common use cases. Each benchmark simulates a realistic scenario of execution, including data transfers via mobile network and production-like sized tasks. Figure 11.15 shows a com-

parison of the time necessary for an user to complete each interaction.

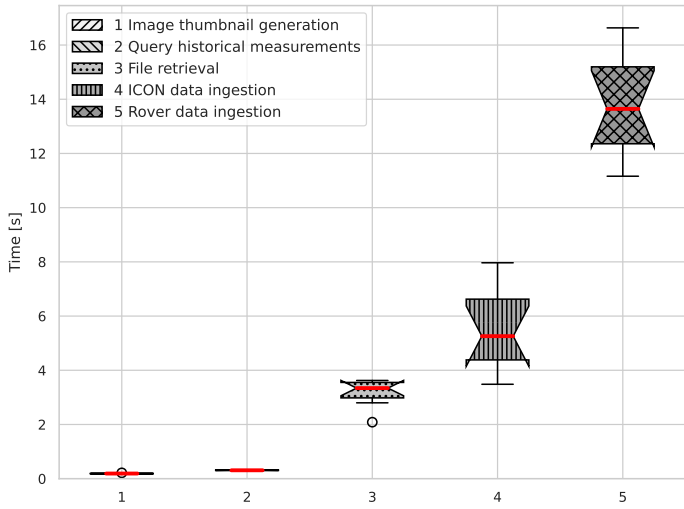


Figure 11.15: Performance comparison between common scenarios.

11.6.1 Image thumbnail generation

Image thumbnails are generated for each ingested PNG and PCD files in order to provide visual guidance to the users. When a supported file type is uploaded through the API, a thumbnail job is scheduled in the pipeline service and executed asynchronously. The measured average time of execution for such jobs is 0.19 seconds, with a standard deviation of 0.009.

11.6.2 Query history of measurements

In this scenario, the user queries the API to retrieve information of physical measures and the relative metadata. The API retrieves the requested data from the data lake and returns it in JSON format. The average time of response for a query collecting the last month measurements for a specific physical location is 0.31 seconds, with a standard deviation of 0.01.

11.6.3 File retrieval

The retrieval of a file consists in the process of querying and accessing a file contained in the storage service through the API. In this scenario, the API backend acts as a proxy, providing access control and logging capabilities. The measured average time of retrieval of a 4MB file is 3.18 seconds, with a standard deviation of 0.48.

11.6.4 ICON data ingestion

The ingestion of data from the ICON IoT platform is handled by a specialized adapter that monitors a subset of items in the system. When the status of the items changes, the adapter notifies the update through the engine API, including any relevant metadata. The measured average time of ingestion of 100 entries is 5.54 seconds, with a standard deviation of 1.59.

11.6.5 Rover data ingestion

This scenario covers the process of upload and ingestion of image files through the API. An adapter component deployed in the rover sends a request to the relevant API endpoint, providing a list of files and the relevant metadata. Then the API backend processes the inputs, saves them in the storage service and registers the new files' metadata in the data lake. The measured average time of ingestion of a 4MB image is 13.76 seconds, with a standard deviation of 1.79.

Chapter 12

Future work

In this thesis, we addressed various issues and gaps detected in E2C continuum deployment infrastructures and methodologies to ensure continuously provision of advanced NFPs. Our solutions have significantly enhanced the existing baseline in literature. However, certain new and intricate gaps surfaced during implementation. We present here some of the most interesting and challenging ones.

Transparent observability Current implementation of monitoring solutions for cloud applications are moving towards transparent observability, with a focus on a standardised and adaptable approach for gathering execution logs, traces and metrics. The prominent player within the field at present is Prometheus, an open-source service and protocol that boasts a high rate of adoption and performance. Our current methodology for incorporating metrics and probes in our assurance process involves altering the application source code to integrate our monitoring endpoints and enable the monitoring probes to access the application state. Adopting an open monitoring standard, such as Prometheus and OpenTelemetry, would facilitate the extraction of information from applications that already integrate such standards. In addition, we might enhance the application to reveal more internal data if required.

Multi-layer assurance Application and infrastructure layer assurance only focus on their respective properties and treat the rest of the execution environment as black boxes, assuming that you cannot trust other layers' properties cannot be trusted. This results in the need for more complex solutions to achieve the same levels of NFPs. For example, applications could introduce encryption at the application layer due to the lack of trust in the infrastructure provider's commitment to correctly implement data-at-rest encryption. This has a significant impact on performance, hence increasing costs. The methodology proposed in Chapter 3 is broad enough to be applied to both applications and the underlying infrastructures, offering a cohesive assurance solution for the entire stack. Unfortunately, implementing this solution in the E2C continuum necessitates cloud providers to furnish more transparent system information. Consequently, the adoption of transparent observability by all parties is a prerequisite for this approach.

Assurance-focused VNF The implementation and integration of a service specifically designed for assurance purposes would establish the basis for a transparently verifiable and certifiable network of services. In this scenario, the Internet Service Provider (ISP) would expose its network monitoring endpoints to an Assurance Layer (AL) that continuously verifies the network and the Non-Functional Properties (NFPs) of the services, using publicly available and standardized contracts. While a particular NFP holds, a CA that trusts the AL may issue a certificate containing the AL evaluation report to the ISP. Subsequently, a user who trusts the CA can acquire the certificate from the ISP and be confident that the property of interest is held. The 5G core network offers a standardised monitoring interface, which enables the collection of data that is conducive to the validation of NFPs. Examples of monitoring endpoints comprise the quantity of users connected, the allocated bandwidth, and service configuration details. Regretfully, certain sensitive information withholding worth for the verification process ought to remain within the network. As a resolution, we recommend the expansion of the 5G core network by integrating an assurance-specific function in its customary NFs. This function would be able to retrieve the sensitive information and run the AL verification software. This would provide more useful information for the verification process and ensure the privacy of sensitive data. To achieve this, the new VNF must be formally defined, specifying its interface and the data it can retrieve from the network.

Adaptive deployment infrastructures We demonstrated in chapters 8 and 9 how the deployment of application compositions in the E2C continuum requires a match between the user-defined SLAs of the deployment configuration and the available infrastructure. It is worth noting that our solution has the capacity to alter the deployment configuration through the adoption of deployment templates to achieve the required NFP where possible. In a more advanced deployment system, the service-interface definition provided in Chapter 3 could be adopted to specify the complete interface of the component and match the user requirements against several possible implementations. This approach is compelling in multi-tenant deployments, like the PaaS in the cloud, allowing the user to select between a CSP-supplied or self-managed variation of the same service in a transparent manner. The choice between multiple solutions is limited by the user's SLA, and the system optimizes according to user-specified parameters, such as cost. This approach would be particularly advantageous in the E2C system, as the deployment manager can identify the most suitable solution based on the available deployment infrastructures, similar to the proposal in Chapter 9.

Chapter 13

Related work

In this chapter, we have conducted a systematic literature review to identify and analyse the relevant publications related to the topics discussed in the previous chapters. We have summarised the main findings and insights from the publications and highlighted the gaps and challenges that require further investigation. Our aim is to provide a comprehensive overview of the state of the art on the topics of interest and to position our work within the existing literature.

13.1 Telco Edge Networks

In recent years, researchers are focusing their efforts on the realization of an efficient and secure cloud to edge continuum infrastructure, to improve latency, reliability, throughput, and efficiency of networks and distributed services [119]. More and more, we can see how several networks and the cloud computing are converging into a unified and coherent continuum, exploiting the host peculiarities, such as locality to the user, high availability and global coverage. Zhao et al. gave a clear representation of all the various aspects of edge computing and networking in [120], including their relationship with Fog computing and MEC, the various technologies available, the processes used for service provisioning and other security and optimisation aspects.

In such an infrastructure, the orchestration of services has the fundamental role of deciding which deployment configuration is best suited for the requested work, aiming to ensure the agreed SLOs, and monitoring and updating the state of the system if a better approach is possible. Researchers have proposed smarter orchestrators that better integrate their function with edge data-centres, moving part of the load to localized infrastructures [121–123], or even on mobile devices [124], based on the capabilities of the nodes and the dynamic load of the system, thus creating the base components of E2C continuum.

With the advent of 5G technologies, researchers pushed the idea of moving part of the services directly to the core network [125]. This kind of hybrid deployment allows for further distribution of the load on a geographical level and to apply transformations to the data before it leaves the network, substantially reducing the network usage and improving latency for locally hosted services.

Additional effort has been put into the standardization of mobile-edge service deployments, with the introduction of MEC, and the definition of a specification format for describing their SLOs. The additional complexity introduced pushed the researchers to generalize orchestrators to handle the multiple levels of the infrastructure, including optimizations of network communication and caching strategies [101, 126–128].

The field of continuum computing involving 5G is still in its infancy and its complexity leaves a large number of open questions for further research, including privacy and security concerns, scheduling optimizations, monitoring solutions, and how developers will embrace this technology in their solutions.

5G and other more specialised networks, such as satellite, are converging into cohesive and complete solutions that simplify the management and configuration of communications, especially in the case of private networks. In their study [129], Wang et al. highlight the increasing demand from users for effective communication and worldwide coverage, particularly in remote regions that are not serviced by commonly available networks. This integration offers numerous advantages, such as uninterrupted service provision and accessibility to network-enabled services, as well as the utilization of the well-established and commonly used MEC of 5G networks for network management. Additionally, it ensures dependable security for data, network and

computing.

Satellites are commonly viewed as a network of relay nodes, but they also have the potential to be equipped with sufficient computing capabilities to act as a target of interest for edge deployments of applications. In Xie et al.'s [130] study, they investigate the potential of satellites within the MEC architecture and highlight its significant benefits. The satellite serves as a storage node for a CDN, offering the ability to cache content. Satellite computation capabilities can be used for computation offloading, which reduces latency by executing applications that would otherwise have to be deployed in the cloud. Network services can dynamically adjust the network topology through SDN depending on connection availability to optimize performance.

In [131], the recent advances in edge computing for IoT networks are thoroughly described, highlighting how cloud and edge computing resources can be strategically utilised to decrease latency and network usage. This involves bringing general-purpose computing capabilities, which are not usually present in IoT devices, as close as possible to the devices for features such as ML-based capabilities. Abbreviations for technical terms are explained upon first use. Hamdam et al. suggest that proximity-based preprocessing of IoT data can significantly decrease the amount of data that needs to be transmitted to the remote cloud-centric destination, minimizing bandwidth consumption. They also identify several limitations of E2C platforms intended for IoT, which include security concerns regarding data confidentiality, integrity, and availability, as well as scalability, device updates, and interoperability issues.

In [132], the focus is on managing IoT networks using SDN, as it is particularly useful for controlling large fleets of Internet-connected devices with stringent security and performance requirements. The paper presents a comprehensive overview of the network layers that IoT traffic traverses, with a particular emphasis on the lower and more critical layers. The survey focuses on ensuring that network components deployed in E2C networks are secure from malicious adversaries, with the details of implementation and deployment left as abstraction for the E2C continuum solution.

In Rinaldi et al.'s study [133], additional types of network nodes are incorporated into the E2C continuum infrastructure, with UAVs and High-Altitude Platforms (HAPs) pinpointed as feasible edge platforms for temporary deployments that possess intermediary capabilities between terrestrial networks

(e.g. 5G) and satellite networks. These solutions would offer comparable computing capabilities to satellites, but without the restrictions imposed by the space environment. They also provide lower latency and extensive coverage of a large geographical area, ranging from 5 to 200 km. The authors present a comprehensive analysis of existing inter-network communication and management solutions. Particularly noteworthy is the examination of roaming behaviour between networks of different types and partially covered regions.

13.2 Assurance for CDN networks

The convergence of managing application deployments, computing infrastructure, networking and data caching in the E2C requires more than ever strong guarantees of performance, security and other advanced NFPs. Wang et al. [134] offer a very clear picture of the different typical infrastructures of the continuum and how data caching is affected by their peculiarities. Zolfaghari et al. [135] show similar results, describing the state of the art of CDNs and how data distribution is adapting to the newly available technologies and techniques enabled by the E2C continuum to improve performance and reduce costs. Both surveys highlight how the locality of edge networks provide the means for fine grained distribution of data based on local content popularity. This information can be exploited by E2C deployment systems enabling smarter management of resources, predictive preallocation of the data and in loco transformations, such as transcoding of video files and data filtering.

Certification methodologies have been successfully applied in many context including software and services. Anisetti et al. [50] proposed a formal certification scheme to validate NFPs cloud-based services. Ardagna et al. [136] described a lightweight certification methodology for cloud environments, supported by continuous monitoring of infrastructures, platforms and services. Stephanow at al. [52] described a test-based certificate solution to identify whether a CSP assured quality levels match the real measurements to prevent fraudulent and opportunistic behaviours. Felici at al. [53] proposed a multilayer security certification scheme based on testing and monitoring probes. The notion of certification have been rarely applied in the past to verify networking protocols and nodes. Wu at al. [137] and Bossert at al. [138] applied

certification to generic network security evaluation. They based their paper on Common Criteria certification model which has severe limits in dynamic environment. Network monitoring, is one of the prominent way to keep control of the networking traffic and behaviour and can be used for obtaining evidence for Certification. Monitoring in ICN networks has been extensively covered in literature, with particular emphasis on security of the network. In [44, 139] the authors proposed a monitoring plane for NDN with the goal of identifying network traffic anomalies and prevent content poisoning attacks. Another interesting solution is the one proposed by Van Adrichem at al. [140], which presented an implementation of an SDN layer for monitoring and traffic shaping in NDN. More recently, research has focused evaluating both networking nodes and protocols [141, 142]. Zhou et al. [141] presented a network-behaviour monitoring schema aimed to identify congestion. Bialas et al. [142] presented a monitoring technique focused on anomaly detection.

Even if monitoring of ICN and trust in general is receiving an increasing attention by researchers, the certification in ICN networks still a quite unexplored topic. In this paper we extend our previous work in [55] that from the best of our knowledge constitute the first attempt to apply certification framework for ICN nodes using a rule-based schema.

13.3 Assurance in Big Data Analysis Platforms

The last decade has seen the rapid growth in the usage of Big Data in a large number of fields, like Cybersecurity [64], Smart Cities [143], Smart Agriculture [144–146]. The complexity of computations and service deployments are growing rapidly with the increase in heterogeneity of the deployment environments, including E2C continuum [27, 119] and hybrid cloud infrastructures [147].

This trend is followed by the awareness of the necessity of protecting the data and its by-products with transparent and comprehensive techniques of security assurance. The most straightforward approach taken into consideration to address these necessities was based on the integration of security policies in the Big Data processes in order to mitigate or prevent risks originated by the mishandling of data [148]. Some initial security assurance techniques focused on Big Data targeted primarily on securing the stored data [149–152] and only

later expanded to a wider definition that includes its handling in the form of data pipelines [77], securing them according to the CIA (*Confidentiality*, *Integrity*, *Availability*) triad [153–155]. This includes security measures like access control policies on the resources and network isolation of the services. However, these security hardening solutions are not enough to cope with dynamicity of Big Data as a service scenario. The complex structure and variety of the stored data reflect an even more diverse usage of the information contained. Lack of a complete understanding of the security aspects of all the stored data may expose the users to significant risks [155]. Users personal data may be protected by a policy that prevents a malicious actor to acquire all their records, but it may still be possible to deanonymise them from leaky statistical data if, for instance the k-anonymity property is not properly enforced [156, 157]. The growth of high-sensitive data (i.e., financial, healthcare) and the introduction of privacy focused regulations like the GDPR and the California Consumer Privacy Act (CCPA) pushed towards the inclusion of *Privacy* in the primary aspects of the security assurance [152, 153, 158]. Emphasizing how breaches may occur at any step in the pipeline [159]. Researchers highlighted the necessity of automated processes to ensure compliance with privacy regulations [77] and high performance [158]. Due to the size and complexity of the systems, monitoring has been the primary approach applicable to defend Big Data applications according to Elsayed et al. [160]. More advanced techniques include continuous monitoring of the behaviour of the data handlers, in order to identify malicious or unexpected actions [64, 65]. Additional security mitigations solutions adopted anomaly detection techniques to detect model poisoning [161] or advanced access control and storage monitoring [162], although their focus is especially localized and small. Security assurance, on the other hand, enables its users to address privacy and security concerns by actively testing the whole Big Data pipeline environment, its configuration and its behaviour, verifying a target set of NFPs demonstrating if the applied countermeasures are effective [66]. The literature is lacking in this particular aspect of assurance, while more focus is given to the evaluation of quality of the systems, the models and the data, as in [70, 163, 164].

Table 13.1 shows a comparison with the most closely related works in terms of offered assurance capabilities (full and partial support are denoted with 3 and $\sim$ respectively). More specifically we consider: i) Pipeline evaluation (P), indicating the capability to inspect pipeline properties (e.g., applying code analysis techniques); ii) Ecosystem evaluation (E), indicating the capability to check services at ecosystem level (e.g., configuration checks); iii) Holistic

Table 13.1: Related work comparison. P: pipeline evaluation, E: ecosystem evaluation, H: holistic evaluation approach, A: automation of evaluation, I: inspectability.

Reference (year)	P	E	H	A	I
G. Manogaran et al. (2016) [64]		∼		3	3
P. P. Sharma et al. (2014) [65]		∼		3	3
M. Anisetti et al. (2021) [71]	3			3	
D. Yadav et al. (2019) [151]	∼	3			
M. Khan et al. (2019) [152]		3	∼		
L. Wang et al. (2019) [77]	∼	3	3	∼	
S. Kavitha et al. (2015) [156]			3		
A. Mehmood et al. (2016) [159]		3	∼		
M. Elsayed et al. (2018) [160]	∼		3	∼	∼
G. Ra et al. (2021) [162]		3	∼		
Ours	3	3	3	3	3

Evaluation (H), indicating the capability to combine multiple sources of information to present an overall evaluation of the entire Big Data solution; iv) Automation (A), indicating whether the system is fully automated and supports continuous verification of the target; v) Inspectability (I), indicating the capability to inspect the internal status of the system in depth. We note that most of the related solutions focus on the evaluation of services at ecosystem level [64, 65, 77, 151, 152, 159, 162], searching for misconfiguration and abnormal behaviours, while very few of them try to address pipeline and in most of the cases just partially [71, 77, 151, 160] which is custom and more complex to evaluate. We also note that most of the solutions proposed in literature [64, 65, 77, 151, 152, 159, 162] focus on detecting very specific problems and just some of them [77, 152, 156, 159, 160, 162] on providing an holistic evaluation on the entire system but just for some specific NFPs. Considering automation and inspectability, most of the actual solutions are fully customized on a given system and for specific properties missing the need of generality [64, 65, 71, 77, 160].

13.4 Assurance Aware Deployment in E2C Continuum

To the best of our knowledge, there is currently no deployment architecture capable of providing a comprehensive and adaptable approach to the persistent deployment of services in the continuum, taking into account customer-defined advanced properties and constraints. Few solutions exist for application deployment, which are compared in Table 13.2 with respect to the above requirements (full and partial support of a requirement is denoted with ✓ and ∼ respectively). Some considered works ([97, 121, 122, 165]) address the E2C Continuum scenario, but only the architecture presented in [165] can seamlessly deploy applications along the Continuum since it is fully independent of the specific technology and CSP involved. QoS requirements and constraints are taken in account, even partially, by most of the works ([94,98,104,121,165,166]), but only [121,166] provide a comprehensive way to model both applications and environment. Finally, there is no solution that performs a life-cycle management of the deployment, adapting to changes in context. In short, all solutions have drawbacks, whether they be the limited application scenario, the dependence on specific technologies, or the inadequate composition model. Our architecture, instead, addresses all the aforementioned requirements, providing a QoS-driven deployment system for the Continuum.

13.5 Experimental scenario: MIND Foods HUB

Big-data techniques have seen a significant increase in popularity in the last decade, especially in fields of research producing large quantities of experimental data. The literature has numerous examples of applications in agronomy. Continuous monitoring of multiple aspects of the cultivation life cycle allows the agronomist to produce clear and useful visualizations, like presented in [167, 168]. Cultivation data can also be used to experimentally asses risk, as in [146], enabling and supporting precision agriculture techniques. The solutions proposed in [145, 169, 170] use the collected data to take preventive action in a localized manner. Data collection and prevention operations can

Table 13.2: Related work comparison.

Author	**Ref.**	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
K. Fu et al.	[122]	✓			✓		∼
A. Orive et al.	[121]	✓	∼		✓	✓	
A. Brogi et al.	[166]		∼	✓	✓		
V. Casola et al.	[165]	✓	✓	✓		✓	
S. Nastic et al.	[97]	✓		✓			
N. Akhtar et al.	[94]		∼	✓		✓	∼
A. Das et al.	[96]			✓	✓	✓	
M. Anisetti et al.	[104]		✓	✓			
J. Quenum et al.	[98]		✓	✓		✓	
Our Work		✓	✓	✓	✓	✓	✓

be automated using unmanned aerial systems, as demonstrated in [170]. IoT based solutions are particularly effective data sources in the field, providing valuable information at a relatively low price [171, 172]. The integration of IoT devices in big-data solutions has been previously studied, like in [173] and [174] The infrastructure proposed can be successfully applied to most research fields with minimal changes. More recently, the trend of migration to cloud solutions provides benefits on the availability of both the data and the computation resources [174].

Chapter 14

Conclusions

In this final chapter we summarised the main findings and contribution described in the previous chapter, discuss the implications and limitations of this work with regards to the gaps identified in Section 2.4.

In Chapter 2 we provided a comprehensive analysis of E2C infrastructures, highlighting how they differ from the more traditional cloud deployments, enumerating the most important technologies in use and their position in the ecosystem. Followed a description of deployment management solutions, including intent-based solutions.

In Chapter 3 we showed how NFPs and their verification play a crucial role in offering robust assurances of QoS to end-users in intricate deployment circumstances, where diverse hosts, technologies, and standards must collaborate seamlessly. To achieve this, we developed an assurance methodology for E2C infrastructures that can model complex scenarios and provide evidence to validate advanced NFPs, such as security, privacy, and locality. By using service-level modelling and transparent monitoring, our methodology overcomes the limitations of other solutions in the literature, which model deployments at a coarser component level, preventing the reuse of metrics and contracts, and failing to compare the implementation of the same service. Furthermore, they concentrate on performance aspects, whereas ours is more generalizable. Our solution addresses the gaps 2.4, 2.4 and 2.4 defining a clear and complete methodology for continuous verification of NFP through transparently collected evidence and publicly available contracts.

The methodology also provides a clear solution for Gap 2.4 using a strict assurance process.

Chapter 4 analyses telco edge networks, their particularities and their application in the E2C continuum. The chapter investigates 5G, satellite, and IoT networks, with particular emphasis on mobile networks and their implementation, standards, and components. In this chapter, we provided a comprehensive account of constructing a fully operational 5G simulator that comes with high-end MEC abilities and improves current solutions for 5G edge deployments.

In Chapter 5, we explained how to implement our assurance methodology on edge networks, with emphasis on 5G due to its more stable and widespread usage. Our research unveiled unique non-functional properties of the standard 5G core network and currently missing functionalities for their verification. In Chapter 11, we provided a throughout example for the definition of metrics, monitoring endpoints, contracts for properties of interest in a 5G edge network with MEC capabilities.

One of the most prevalent uses of edge networks are CDNs. In Chapter 6, we described CDNs and showed how to build an efficient implementation in a NDN network using the ICN paradigm. Additionally, we demonstrated how our assurance methodology can be applied to this system, resulting in a decentralized and collaborative solution for assurance in NDN networks. In Chapter 11, the capabilities of this implementation were demonstrated through experimental evaluation of its performance in a simulated network.

In Chapter 7 we showed a real-world application of our approach in the framework of a security evaluation of the H2020 EVOTION Big-Data-Analytics-as-a-Service platform, later applied to the MIND Foods Hub project. Our experimental evaluation of the proposed solution, reported in Chapter 11 and implemented in the framework of the experimental scenario in Chapter 10 identified a number of gaps to be addressed. In particular, it showed i) how difficult is to keep control over NFPs in a system with custom parts (i.e., the pipeline code); ii) how complex is the configuration of a Big Data ecosystem increasing the possibility of misconfiguration, and iii) what is needed to have a real generic open platform for trusted Big Data computation.

Chapter 8 examines the issue of ensuring NFP in 5G edge deployment infrastructures, demonstrating that continuous assurance offers robust guarantees

to platform users. We demonstrated how an assurance-aware deployment is instantiated in a MEC-enabled 5G network. Our solution aimed to efficiently and effectively deploy analytics pipelines in the 5G-enabled Edge continuum, while considering users' requirements and preferences, resource characteristics and limitations, and environmental dynamics and uncertainties. We also detailed the specification of advanced user-defined SLAs and how they can be integrated with the deployment specification standards already available.

Chapter 9 explained the development of an assurance-aware deployment solution for distributed environments. This solution guarantees strong NFPs in any infrastructure, by matching available host properties with user-defined SLAs of the application composition. Chapter 11 presents experimental data on the performance of the proof-of-concept implementation of our deployment matching solution. The combination of assurance aware infrastructures and deployments provides a solution to gap 2.4, allowing automated deployment of applications while guaranteeing strong NFPs.

In Chapter 10, we presented the real-world application scenario that was used to experimentally evaluate our methodology in its different implementations. The project involved designing, implementing and deploying a big data engine with continuous analysis and CDN capabilities in an E2C setting. In Chapter 11, we detailed the performance results of our system in typical usage scenarios.

The combination of infrastructure assurance, described in Chapter 3, and deployment assurance, analysed in Chapter 8, addresses the gap 2.4 providing a coherent and complete methodology for the continuous verification of robust user-defined SLAs based on NFP, evidence collected through transparent monitoring and publicly available contracts.

The results obtained demonstrated the feasibility of the proposed assurance methodology, which offers strong assurance with low overhead. We have shown how the introduction of such techniques into the management of E2C infrastructures and deployments can bring significant benefits to their users. The application of assurance solutions in this area is still in its infancy, but the reception from the academic community and industry has been positive, and we expect to see solid progress in the coming years.

Appendix A

Publications

[175]

Title: Script Language Security
Authors: Ardagna, Claudio A. and Damiani, Ernesto and Berto, Filippo
Book: Encyclopedia of Cryptography, Security and Privacy, pp. 1–3
Year: 2023

[23]

Title: An assurance process for Big Data trustworthiness
Authors: Anisetti, Marco and Ardagna, Claudio A. and Berto, Filippo
Journal: Future Generation Computer Systems, pp. 34–46
Year: 2023
Abstract: Modern (industrial) domains are based on large digital ecosystems where huge amounts of data and information need to be collected, shared, and analyzed by multiple actors working within and across organizational boundaries. This data-driven ecosystem poses strong requirements on data management and data analysis, as well as on data protection and system trustworthiness. However, although Big Data has reached its functional maturity and represents a key enabler for enterprises to compete in the global market, the assurance and trustworthiness of Big Data computations (e.g., security, privacy) are still in their infancy. While functionally appealing, Big Data does not provide a transparent environment with clear non-functional properties, impairing the users' ability to evaluate its behavior and clashing with modern data-privacy regulations. In this paper, we present a novel assurance process for Big Data, which evaluates the Big Data pipelines, and

the Big Data ecosystem underneath, to provide a comprehensive measure of their trustworthiness. To the best of our knowledge, this approach is the first attempt to address the general problem of Big Data trustworthiness in an holistic way. We experimentally evaluate our solution in a real Big Data Analytics-as-a-Service environment, first presenting a detailed walkthrough evaluation, and then showing its feasibility and negligible performance overhead (i.e., approx 1 min).

[176]

Title: QoS-aware Deployment of Service Compositions in 5G-empowered Edge-Cloud Continuum
Authors: Anisetti, Marco and Berto, Filippo and Bondaruc, Ruslan
Book: 2023 IEEE International Conference on Cloud Computing (CLOUD), (to appear)
Year: 2023
Abstract: Nowadays, modern service compositions are increasingly adopted in critical scenarios where advanced Quality of Services (QoS) such as low latency, security, and privacy are fundamental. The landing platforms for the deployment of such compositions are progressively becoming capable to offer capabilities that support such advanced QoS requests (e.g., low latency via 5G network slice) spanning the Edge-Cloud Continuum. Actual deployment solutions focus mainly on resource allocation (i.e., CPU, memory, and storage), falling short of addressing advanced QoS and unleashing the true potential of the Edge-Cloud Continuum. In this paper, we present an automatic QoS-aware deployment solution for composed services in the Edge-Cloud Continuum. It compares QoS requests on the service composition with the capabilities of a given continuum in order to find, generate and execute suitable deployment recipes. Our preliminary experimental evaluation demonstrates the feasibility of our solution in a realistic scenario.

[144]

Title: A 5G-IoT enabled Big Data infrastructure for data-driven agronomy
Authors: Berto, Filippo and Ardagna, Claudio and Torrente, Marco and Manenti, Daniele and Ferrari, Enrico and Calcante, Aldo and Oberti, Roberto and Fra', Cristina and Ciani, Luca
Book: 2022 IEEE Globecom Workshops (GC Wkshps), pp. 588–594
Year: 2022
Abstract: The increasing necessity of efficient and effective agriculture

has pushed towards the development of computer aided techniques, where on-field measurements are used to take objective decisions to optimize the production, giving birth to data-driven agronomy. With the diffusion of 5G-based IoT devices it becomes possible to deploy a variety of sensors in large amounts, enabling continuous collection of monitoring data. Agronomists necessitate the adoption of Big Data techniques and technologies to handle such large amount of data. These solutions provide powerful tools to analyze and model the complexity of the field, i.e. applying statistics and Machine Learning based methods to the product processes. In this paper, we propose a Big Data infrastructure that integrates with 5G-enabled sensors, providing scalable data ingestion, pipelining and information querying capabilities. We also show a practical scenario where our infrastructure has been implemented and report preliminary results on its performance.

[6]

Title: A Security Certification Scheme for Information-Centric Networks
Authors: Anisetti, Marco and Ardagna, Claudio A. and Berto, Filippo and Damiani, Ernesto
Journal: IEEE Trans. Netw. Serv. Manage., pp. 2397–2408
Year: 2022
Abstract: Information-Centric Networking is an emerging alternative to host-centric networking designed for large-scale content distribution and stricter privacy requirements. Recent research on Information-Centric Networking focused on the protection of the network from attacks targeting the content delivery protocols, while assuming genuine content can always be retrieved from trustworthy nodes. In this paper, we depart from the assumption of the trustworthiness of network nodes and propose a novel certification methodology for informationcentric networks that supports continuous security verification of non-functional properties. Our methodology provides a complete and detailed view of the network security status, increasing the trustworthiness of the network and its services. The proposed approach builds on an enhanced certification model capturing the evolution of the system over time. It also defines certification services that fully integrate with existing networks to collect evidence on the target of certification and carry out the certification process. It finally proposes two certification processes, centralized and decentralized, balancing the impact on the network and the system performance. Efficiency, performance, and soundness of our approach are experimentally evaluated in a simulated Named Data Networking (NDN) network targeting property availability.

[27]

Title: Orchestration of data-intensive pipeline in 5G-enabled Edge Continuum
Authors: Anisetti, Marco and Berto, Filippo and Banzi, Massimo
Book: 2022 IEEE World Congress on Services (SERVICES), pp. 2–10
Year: 2022
Abstract: Nowadays there is an increasing trend in the volume and velocity of data, typically consumed by data-intensive AI/ML-based services, requiring a larger diffusion of more effective Edge computing approaches. In addition, we are experiencing an increment of critical applications using an increasing volume of sensitive data and requiring advanced security and privacy protections. 5G Edge technology can foster a more diffused Edge computing adoption but several challenges in terms of interoperability. Handling data-intensive pipelines on the 5Genabled Edge continuum, considering specific QoS requirements including security and privacy, is still in its infancy. In this paper, we propose an initial solution for deploying a data-intensive pipeline in a 5G-enabled Edge continuum satisfying specific QoS requirements. Our approach is based on a QoS-aware meta orchestration modeling of a given pipeline and an orchestration builder generating deployable Edge-specific orchestrations. In this paper, we also present an initial walkthrough scenario in the context of a wet lab analysis pipeline to be deployed on the 5G-enabled Edge continuum.

[24]

Title: A DevSecOps-based Assurance Process for Big Data Analytics
Authors: Anisetti, Marco and Bena, Nicola and Berto, Filippo and Jeon, Gwanggil
Book: 2022 IEEE International Conference on Web Services (ICWS), pp. 1–10
Year: 2022
Abstract: Today big data pipelines are increasingly adopted by service applications representing a key enabler for enterprises to compete in the global market. However, the management of non-functional aspects of the big data pipeline (e.g., security, privacy) is still in its infancy. As a consequence, while functionally appealing, the big data pipeline does not provide a transparent environment, impairing the users' ability to evaluate its behavior. In this paper, we propose a security assurance methodology for big data pipelines

grounded on the DevSecOps development paradigm to increase trustworthiness allowing reliable security and privacy by design. Our methodology models and annotates big data pipelines with non-functional requirements verified by assurance checks ensuring requirements to hold along with the pipeline lifecycle. The performance and quality of our methodology are evaluated in a real walkthrough analytics scenario.

[55]

Title: Security Certification Scheme for Content-centric Networks
Authors: Anisetti, Marco and Ardagna, Claudio A and Berto, Filippo and Damiani, Ernesto
Book: 2021 IEEE International Conference on Services Computing (SCC), pp. 203–212
Year: 2021
Abstract: Content-centric networking is emerging as a credible alternative to host-centric networking, especially in scenarios of large-scale content distribution and where privacy requirements are crucial. Recently, research on content-centric networking has focused on security aspects and proposed solutions aimed to protect the network from attacks targeting the content delivery protocols. Content-centric networks are based on the strong assumption of being able to access genuine content from genuine nodes, which is however unrealistic and could open the door to disruptive attacks. Network node misbehavior, either due to poisoning attacks or malfunctioning, can act as a persistent threat that goes unnoticed and causes dangerous consequences. In this paper, we propose a novel certification methodology for content-centric networks that improves transparency and increases trustworthiness of the network and its nodes. The proposed approach builds on behavioral analysis and implements a continuous certification process that collects evidence from the network nodes and verifies their non-functional properties using a rule-based inference model. Utility, performance, and soundness of our approach have been experimentally evaluated on a simulated Named Data Networking (NDN) network targeting properties availability, integrity, and non-repudiation.

[177]

Title: Spatial bloom filter in named data networking: a memory efficient solution

Authors: Berto, Filippo and Calderoni, Luca and Conti, Mauro and Losiouk, Eleonora

Book: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 274–277

Year: 2020

Abstract: Among the possible future Internet architectures, Information Centric Networking (ICN) is the most promising one and researchers working on the Named Data Networking (NDN) project are putting efforts towards its deployment in a real scenario. To properly handle content names, the different components of an NDN network need efficient and scalable data structures. In this paper, we propose a new data structure to support the NDN forwarding procedure by replacing the current Forwarding Information Base (FIB): the Spatial Bloom Filter (SBF), a probabilistic data structure that guarantees fast lookup and efficient memory consumption. Through a set of simulations run to compare the performance of FIB and SBF, we found that the latter uses less than 5 KB of data to handle 106 queried interests, with a (negligible) probability 10-4 of false positive events. Conversely, the FIB requires up to 2.5 GB of data in disadvantageous cases, e.g. when interests are composed of a considerable number of components.

Acronyms

5G-PPP 5G Infrastructure Public Private Partnership.
5GS 5G System.

AA Assurance Agent.
ACL Access Control List.
AF Application Function.
AI Artificial Intelligence.
AL Accredited Lab.
AMF Access and mobility Management Function.
API Application Programming Interface.
ARP Address Resolution Protocol.
AUSF Authentication Server Function.
AWS Amazon Web Services.

BNF Backus-Naur Form.
BSS Business Support System.

CA Certification Authority.
CCPA California Consumer Privacy Act.
CDN Content Distribution Network.
CN Core Network.
CNF Cloud-native Network Function.
CP Control Plane.
CPU Central Processor Unit.
CRUD Create, Read, Update, Delete.
CS Content Store.
CSMF Communication Service Management Function.
CSP Cloud Service Provider.
CVSS Common Vulnerability Scoring System.

DAG Directed Acyclic Graph.
DHCP Dynamic Host Configuration Protocol.
DN Data Network.
DNS Domain Name System.
DOS Denial Of Service.

E2C Edge-Cloud.
EAP Extensible Authentication Protocol.
ETSI European Telecommunications Standards Institute.

FaaS Function as a Service.
FHIR Fast Health Interoperable Resources.
FL Federated Learning.
FTTH Fibre to the Home.

GDPR General Data Privacy Regulation.
GEO Geostationary Earth Orbit.
GLB Greatest Lower Bound.
gNB gNodeB.
GSMA GSM Association.
GUI Graphical User Interface.

HAP High-Altitude Platform.
HTTP HyperText Transfer Protocol.

IaaS Infrastructure as a Service.
IBN Intent Based Networking.
IBS Intent Based System.
ICMP Internet Control Message Protocol.
ICN Information Centric Networking.
IETF Internet Engineering Task Force.
IM Information Model.
IMEI International Mobile Equipment Identity.
IoT Internet of Things.
IP Internet Protocol.
ISP Internet Service Provider.
ITU International Telecommunication Union.

KPI Key Performance indicator.

LADN Local Area Data Network.
LEO Low Earth Orbit.
LIDAR Laser Imaging Detection and Ranging.
LRU Least Recently Used.
LUB Least Upper Bound.

MANO MANagement and Orchestration.
MEAO Mobile Edge Application Orchestrator.
MEC Multi-access Edge Computing.
MEO Medium Earth Orbit.
MEP Mobile Edge Platform.
MEPM Mobile Edge Platform Manager.
ML Machine Learning.
MQTT Message Queuing Telemetry Transport.

NAS Non-access stratum.
NBI North Bound Interface.
NDN Named Data Networking.
NEF Network Exposure Function.
NF Network Function.
NFP Non-Functional Property.
NFV Network Function Virtualisation.
NGMN Next Generation Mobile Networks.
NIST National Institute of Standards and Technology.
NLSR Named Data Link State Routing.
NRF Network Repository Function.
NS Network Slicing.
NSSAI Network Slice Selection Assistance Information.
NSSF Network Slice Selection Function.
NSSMF Network Slice Subnet Management Function.
NWDAF Network Data Analytics Function.

O-RAN Open-Radio Access Network.

OAIC Open AI Cellular.
OSM Open Source MANO.
OSS Operational Support System.

PA Precision Agriculture.
PaaS Platform as a Service.
PCF Policy Control Function.
PDU Protocol Data Unit.
PIT Pending Interest Table.
PKI Public Key Infrastructure.
PoP Point of presence.

QoS Quality of Service.

RAN Radio Access Network.
RAS Reliability, Availability, and Serviceability.
RCS Rich Communication Services.
RIC Ran Intelligent Controller.
RTT Round Trip Time.

S3 Simple Storage Service.
SCP Service Communication Proxy.
SDN Software Defined Network.
SIM Subscriber Identity Module.
SLA Service Level Agreement.
SLO Service Level Objective.
SMF Session Management Function.
SMS Short Message Service.
SNMP Simple Network Management Protocol.
SoC System-on-a-Chip.

TCP Transmission Control Protocol.

UAV Unmanned Aerial Vehicles.
UDM Unified Data Management.
UDR Unified Data Repository.
UE User Equipment.
UP User Plane.
UPF User Plane Function.

VIM Virtualisation Infrastructure Manager.
VM Virtual Machine.
VN Virtualised Network.
VNF Virtualised Network Function.
VNFD Virtualised Network Function Descriptor.
VNFO Virtualised Network Function Orchestrator.
VPN Virtual Private Network.
VPS Virtual Private Server.

Bibliography

[1] G. He, X. Liao, and C. Liu, "A Security Survey of NFV: From Causes to Practices," in *2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, Jan. 2023, pp. 624–628.

[2] S. Sultan, I. Ahmad, and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," *IEEE Access*, vol. 7, pp. 52 976–52 996, 2019, conference Name: IEEE Access.

[3] K. E. Narayana and K. Jayashree, "Survey on cross virtual machine side channel attack detection and properties of cloud computing as sustainable material," *Materials Today: Proceedings*, vol. 45, pp. 6465–6470, Jan. 2021.

[4] W. Xiong and J. Szefer, "Survey of Transient Execution Attacks and Their Mitigations," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 54:1–54:36, May 2021.

[5] C. Shen, C. Chen, and J. Zhang, "Micro-architectural Cache Side-Channel Attacks and Countermeasures," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '21. New York, NY, USA: Association for Computing Machinery, Jan. 2021, pp. 441–448.

[6] M. Anisetti, C. A. Ardagna, F. Berto, and E. Damiani, "A Security Certification Scheme for Information-Centric Networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 3, pp. 2397–2408, Sep. 2022.

[7] "5G Observatory Biannual Report: April 2023," Directorate-General

for Communications Networks, Content and Technology, Tech. Rep. 18, Apr. 2023.

[8] "European FTTH/B Market Panorama 2023," FTTH Council Europe, Tech. Rep., Apr. 2023.

[9] ETSI, "System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.6.0 Release 16)," ETSI ISG, Tech. Rep. ETSI TS 123 501 V16.6.0, Oct. 2020.

[10] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018, publisher: IEEE.

[11] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[12] ETSI, *GS MEC 003 Multi-access Edge Computing (MEC); Framework and Reference Architecture*, V3.1.1, Mar. 2022.

[13] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, and others, "MEC in 5G networks," *ETSI white paper*, vol. 28, no. 2018, pp. 1–28, 2018.

[14] A. Ksentini and P. A. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5G," *IEEE Network*, vol. 34, no. 2, pp. 99–105, Mar. 2020, publisher: IEEE.

[15] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 196–248, 2020.

[16] T. He, E. N. Ciftcioglu, S. Wang, and K. S. Chan, "Location Privacy in Mobile Edge Clouds: A Chaff-Based Approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2625–2636, 2017, publisher: IEEE.

[17] G. Sahu and S. S. Pawar, "Security Challenges in 5G Network," in *Software Defined Networking for Ad Hoc Networks*. Springer, 2022, pp. 75–94.

[18] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," RFC Editor, Tech. Rep., Oct. 2022, issue: 9315 Num Pages: 23 Series: Request for Comments Published: RFC 9315.

[19] "Autonomous Networks: Empowering Digital Transformation For Smart Societies and Industries," TM Forum, Tech. Rep., Oct. 2020.

[20] J. Niemoller, R. Szabo, A. Zahemszky, and D. Roeland, "Creating autonomous networks with intent-based closed loops," *Ericsson review (English edition)*, vol. 2022, no. 4, pp. 2–11, 2022.

[21] W. Zhang, D. Yang, and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, Sep. 2019, conference Name: IEEE Systems Journal.

[22] C. Krupitzer, T. Wagenhals, M. Züfle, V. Lesch, D. Schäfer, A. Mozaffarin, J. Edinger, C. Becker, and S. Kounev, "A Survey on Predictive Maintenance for Industry 4.0," Feb. 2020, arXiv:2002.08224 [cs].

[23] M. Anisetti, C. A. Ardagna, and F. Berto, "An assurance process for Big Data trustworthiness," *Future Generation Computer Systems*, vol. 146, pp. 34–46, Sep. 2023.

[24] M. Anisetti, N. Bena, F. Berto, and G. Jeon, "A DevSecOps-based Assurance Process for Big Data Analytics," in *2022 IEEE International Conference on Web Services (ICWS)*. Barcelona, Spain: IEEE, Jul. 2022, pp. 1–10.

[25] S. Kukliński and L. Tomaszewski, "Key Performance Indicators for 5G network slicing," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, Jun. 2019, pp. 464–471.

[26] H. Yang, L. Lu, Y. Hu, R. Liu, K. Yao, X. Duan, and T. Sun, "A Technical Research towards 5G SLA: System Definition, Sense and Assurance

Solution," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*. Tianjin, China: IEEE, Oct. 2021, pp. 462–471.

[27] M. Anisetti, F. Berto, and M. Banzi, "Orchestration of data-intensive pipeline in 5G-enabled Edge Continuum," in *2022 IEEE World Congress on Services (SERVICES)*. Terassa, Spain: IEEE, Jul. 2022, pp. 2–10, iSSN: 2642-939X.

[28] P. Ranaweera, A. Jurcut, and M. Liyanage, "MEC-enabled 5G Use Cases: A Survey on Security Vulnerabilities and Countermeasures," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 186:1–186:37, Oct. 2021.

[29] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020, conference Name: IEEE Access.

[30] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge Computing in 5G: A Review," *IEEE Access*, vol. 7, pp. 127 276–127 289, 2019, conference Name: IEEE Access.

[31] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named Data Networking: A survey," *Computer Science Review*, vol. 19, pp. 15–55, Feb. 2016.

[32] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A Brief Introduction to Named Data Networking," in *Proc. of MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. Los Angeles, CA: IEEE, Oct. 2018, pp. 1–6, event-place: Los Angeles, CA.

[33] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named Data Networking in Vehicular Ad Hoc Networks: State-of-the-Art and Challenges," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 1, pp. 320–351, 2020.

[34] Z. Li, Y. Liu, Y. Chen, Y. Xu, and K. Liu, "Performance analysis

of a novel 5G architecture via Content-Centric Networking," *Physical Communication*, vol. 25, pp. 328–331, Dec. 2017.

[35] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named Data Networking of Things (Invited Paper)," in *Proc. of 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Berlin, Germany, Apr. 2016, pp. 117–128, event-place: Berlin, Germany.

[36] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *IEEE Network*, vol. 28, no. 6, pp. 91–96, Nov. 2014.

[37] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proc. of ACM SIGCOMM workshop on Information-centric networking*, ser. ICN '11. New York, NY, USA: ACM, Aug. 2011, pp. 13–18, event-place: New York, NY, USA.

[38] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in Named Data Networking," in *Proc. of 2013 IFIP Networking Conference*, Brooklyn, NY, USA, May 2013, pp. 1–9, event-place: Brooklyn, NY, USA.

[39] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in Named Data Networking," in *Proc. of 22nd ICCCN*, Nassau, Bahamas, Jul. 2013, pp. 1–7, event-place: Nassau, Bahamas.

[40] L. Yao, Y. Zeng, X. Wang, A. Chen, and G. Wu, "Detection and Defense of Cache Pollution Based on Popularity Prediction in Named Data Networking," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[41] H. Salah, M. Alfatafta, S. SayedAhmed, and T. Strufe, "CoMon++: Preventing Cache Pollution in NDN Efficiently and Effectively," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. Singapore: IEEE, Oct. 2017, pp. 43–51, place: Singapore.

[42] A. Karami and M. Guerrero-Zapata, "An ANFIS-based cache replace-

ment method for mitigating cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 80, pp. 51–65, Apr. 2015.

[43] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, Nov. 2013.

[44] T. Nguyen, H. Mai, G. Doyen, R. Cogranne, W. Mallouli, E. M. d. Oca, and O. Festor, "A Security Monitoring Plane for Named Data Networking Deployment," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 88–94, Nov. 2018.

[45] P. Tammana, R. Agarwal, and M. Lee, "Distributed Network Monitoring and Debugging with SwitchPointer," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 453–456.

[46] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network Monitoring in Software-Defined Networking: A Review," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958–3969, Dec. 2018.

[47] T. Nguyen, X. Marchal, G. Doyen, T. Cholez, and R. Cogranne, "Content Poisoning in Named Data Networking: Comprehensive characterization of real deployment," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, Portugal, May 2017, pp. 72–80.

[48] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Computer Networks*, vol. 65, pp. 84–98, Jun. 2014.

[49] "ISO 15408 CC – Common Criteria," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 648–648.

[50] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, "A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 30–43, Jan. 2020, conference Name: IEEE Transactions on Services Computing.

[51] M. Anisetti, C. Ardagna, E. Damiani, and G. Polegri, "Test-Based Security Certification of Composite Services," *ACM Trans. Web*, vol. 13, no. 1, pp. 3:1–3:43, Dec. 2018.

[52] P. Stephanow, G. Srivastava, and J. Schütte, "Test-based cloud service certification of opportunistic providers," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. San Francisco, CA, USA: IEEE, 2016, pp. 843–848.

[53] M. Egea, K. Mahbub, G. Spanoudakis, and M. R. Vieira, "A Certification Framework for Cloud Security Properties: The Monitoring Path," in *Accountability and Security in the Cloud*, M. Felici and C. Fernández-Gago, Eds. Cham: Springer International Publishing, 2015, vol. 8937, pp. 63–77.

[54] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A Survey of Security Attacks in Information-Centric Networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1441–1454, 2015.

[55] M. Anisetti, C. A. Ardagna, F. Berto, and E. Damiani, "Security Certification Scheme for Content-centric Networks," in *2021 IEEE International Conference on Services Computing (SCC)*. Chicago, IL, USA: IEEE, Sep. 2021, pp. 203–212.

[56] G. D'Acquisto, J. Domingo-Ferrer, P. Kikiras, V. Torra, Y.-A. de Montjoye, and A. Bourka, "Privacy by design in big data: an overview of privacy enhancing technologies in the era of big data analytics," *ENISA*, 2015.

[57] W. L. Chang, "NIST Big Data Interoperability Framework: Volume 4, Security and Privacy," National Institute of Standards and Technology, Tech. Rep., 2015.

[58] CSA, "Big Data Security and Privacy Handbook: 100 Best Practices in Big Data Security and Privacy," Cloud Security Alliance, Tech. Rep., Aug. 2016.

[59] D. S. Terzi, R. Terzi, and S. Sagiroglu, "A survey on security and privacy issues in big data," in *2015 10th International Conference for*

Internet Technology and Secured Transactions (ICITST). IEEE, Dec. 2015, pp. 202–207.

[60] S. Yakoubov, V. Gadepally, N. Schear, E. Shen, and A. Yerukhimovich, "A survey of cryptographic approaches to securing big-data analytics in the cloud," in *Proc. of HPEC 2014*, Waltham, MA, USA, Sep. 2014.

[61] X. Zhang, L. Qi, W. Dou, Q. He, C. Leckie, R. Kotagiri, and Z. Salcic, "MRMondrian: Scalable Multidimensional Anonymisation for Big Data Privacy Preservation," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 125–139, Feb. 2022, publisher: IEEE.

[62] G. Manogaran, C. Thota, and M. V. Kumar, "MetaCloudDataStorage architecture for big data security in cloud computing," *Procedia Computer Science*, vol. 87, pp. 128–133, 2016, publisher: Elsevier.

[63] P. P. Sharma and C. P. Navdeti, "Securing big data hadoop: a review of security issues, threats and solution," *Int. J. Comput. Sci. Inf. Technol*, vol. 5, no. 2, pp. 2126–2131, 2014.

[64] D. B. Rawat, R. Doku, and M. Garuba, "Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2055–2072, Nov. 2021.

[65] I. Hababeh, A. Gharaibeh, S. Nofal, and I. Khalil, "An Integrated Methodology for Big Data Classification and Security for Improving Cloud Systems Data Mobility," *IEEE Access*, vol. 7, pp. 9153–9163, 2019, publisher: Institute of Electrical and Electronics Engineers Inc.

[66] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From Security to Assurance in the Cloud: A Survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–50, Sep. 2015.

[67] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Rao, "Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, Sep. 2014, publisher: IEEE.

[68] A. Kiesow, N. Zarvic, and O. Thomas, "Continuous Auditing in Big

Data Computing Environments: Towards an Integrated Audit Approach by Using CAATTs." in *GI-Jahrestagung*, 2014, pp. 901–912.

[69] W. Li, Y. Yang, and D. Yuan, *Reliability Assurance of Big Data in the Cloud: Cost-Effective Replication-based Storage*. Elsevier, Dec. 2015.

[70] J. Gao, C. Xie, and C. Tao, "Big data validation and quality assurance - Issuses, challenges, and needs," in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. Oxford, UK: IEEE, Mar. 2016, pp. 433–441.

[71] M. Anisetti, C. A. Ardagna, C. Braghin, E. Damiani, A. Polimeno, and A. Balestrucci, "Dynamic and Scalable Enforcement of Access Control Policies for Big Data," in *The 13th International Conference on Management of Digital EcoSystems*, vol. 8. ACM, Nov. 2021, pp. 71–78, issue: 21.

[72] M. Prasinos, I. Basdekis, M. Anisetti, G. Spanoudakis, D. Koutsouris, and E. Damiani, "A Modelling Framework for Evidence-Based Public Health Policy Making," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 2388–2399, May 2022, publisher: Institute of Electrical and Electronics Engineers Inc.

[73] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," in *Proc. of CTS 2013*, San Diego, CA, USA, May 2013.

[74] A. Jakóbik, "Big Data Security," in *Resource Management for Big Data Platforms: Algorithms, Modelling, and High-Performance Computing Techniques*, F. Pop, J. Ko\lodziej, and B. Di Martino, Eds. Cham: Springer International Publishing, 2016, pp. 241–261.

[75] "Cloud Controls Matrix (CCM) v3.0.1," Cloud Security Alliance, Tech. Rep., 2019, backup Publisher: Cloud Security Alliance.

[76] P. Stephanow and N. Fallenbeck, "Towards Continuous Certification of Infrastructure-as-a-Service Using Low-Level Metrics," in *Proc. of IEEE UIC-ATC-ScalCom*, Beijing, China, Aug. 2015.

[77] L. Wang, J. P. Near, N. Somani, P. Gao, A. Low, D. Dao, and D. Song,

"Data Capsule: A New Paradigm for Automatic Compliance with Data Privacy Regulations," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, Cham, 2019, vol. 11721 LNCS, pp. 3–23.

[78] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Saonara, "A Test-based Security Certification Scheme for Web Services," *ACM Transactions on the Web (TWEB)*, vol. 7, no. 2, pp. 1–41, May 2013.

[79] C. A. Ardagna, V. Bellandi, M. Bezzi, P. Ceravolo, E. Damiani, and C. Hebert, "Model-Based Big Data Analytics-as-a-Service: Take Big Data to the Next Level," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 516–529, Mar. 2021.

[80] G. Spanoudakis, P. Katrakazas, D. Koutsouris, D. Kikidis, A. Bibas, and N. H. Pontopidan, "Public Health Policy for Management of Hearing Impairments Based on Big Data Analytics: EVOTION at Genesis," in *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, vol. 2018-January. IEEE, Oct. 2017, pp. 525–530.

[81] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, "A Security Benchmark for OpenStack," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, vol. 2017-June. IEEE, Jun. 2017, pp. 294–301.

[82] B. A. Bouna, C. Clifton, and Q. Malluhi, "Efficient sanitization of unsafe data correlations," in *EDBT/ICDT 2015 Joint Conference*, Brussels, Belgium, Mar. 2015.

[83] C. Carrión, "Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 138:1–138:37, Dec. 2022.

[84] Z. Rejiba and J. Chamanara, "Custom Scheduling in Kubernetes: A Survey on Common Problems and Solution Approaches," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 151:1–151:37, Dec. 2022.

[85] C. A. Ardagna, R. Asal, E. Damiani, N. E. Ioini, M. Elahi, and C. Pahl, "From Trustworthy Data to Trustworthy IoT: A Data Collec-

tion Methodology Based on Blockchain," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 1, pp. 1–26, Dec. 2021.

[86] C. A. Ardagna, R. Asal, E. Damiani, N. El Ioini, and C. Pahl, "Trustworthy IoT: An evidence collection approach based on smart contracts," in *2019 IEEE International Conference on Services Computing (SCC)*. IEEE, 2019, pp. 46–50.

[87] "IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing," *IEEE Std 1934-2018*, pp. 1–176, 2018.

[88] M. Anisetti, C. A. Ardagna, N. Bena, and R. Bondaruc, "Towards an Assurance Framework for Edge and IoT Systems," in *2021 IEEE International Conference on Edge Computing (EDGE)*, Guangzhou, China, 2021, pp. 41–43.

[89] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, Aug. 2019.

[90] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, Mar. 2020.

[91] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani, "An Assurance Framework and Process for Hybrid Systems," in *Communications in Computer and Information Science*, vol. 1484 CCIS, 2021, pp. 79–101.

[92] N. Rieke, J. Hancox, W. Li, F. M. ı, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, pp. 1–7, Sep. 2020, publisher: Nature Publishing Group.

[93] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The Internet of Things, Fog and Cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, Oct. 2018.

[94] N. Akhtar, A. Raza, V. Ishakian, and I. Matta, "COSE: Configuring Serverless Functions using Statistical Learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 129–138.

[95] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan, "A Serverless Real-Time Data Analytics Platform for Edge Computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

[96] A. Das, A. Leaf, C. A. Varela, and S. Patterson, "Skedulix: Hybrid Cloud Scheduling for Cost-Efficient Execution of Serverless Applications," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, Oct. 2020, pp. 609–618.

[97] S. Nastic, P. Raith, A. Furutanpey, T. Pusztai, and S. Dustdar, "A Serverless Computing Fabric for Edge & Cloud," in *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, 2022, pp. 1–12.

[98] J. G. Quenum and J. Josua, "Multi-Cloud Serverless Function Composition," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–10, event-place: Leicester, United Kingdom.

[99] M. Anisetti, C. A. Ardagna, and N. Bena, "Multi-Dimensional Certification of Modern Distributed Systems," *IEEE Transactions on Services Computing*, pp. 1–14, 2022.

[100] "GS MEC 003 - V3.1.1 - Multi-access Edge Computing (MEC); Framework and Reference Architecture," vol. 1, 2022.

[101] F. Giannone, P. A. Frangoudis, A. Ksentini, and L. Valcarenghi, "Orchestrating heterogeneous MEC-based applications for connected vehicles," *Computer Networks*, vol. 180, p. 107402, Oct. 2020, publisher: Elsevier.

[102] K. Liolis, J. Cahill, E. Higgins, M. Corici, E. Troudt, and P. Sutton, "Over-the-air demonstration of satellite integration with 5G core net-

work and multi-access edge computing use case," in *IEEE 5G World Forum, 5GWF 2019 - Conference Proceedings*, Sep. 2019, pp. 1–5.

[103] M. Anisetti, C. A. Ardagna, and E. Damiani, "Security Certification of Composite Services: A Test-Based Approach," in *Proc. of the 20th IEEE International Conference on Web Services (ICWS 2013)*, San Francisco, CA, USA, Jul. 2013, pp. 475–482.

[104] M. Anisetti, C. A. Ardagna, E. Damiani, F. Gaudenzi, and G. Jeon, "Cost-effective deployment of certified cloud composite services," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 203–218, 2020.

[105] M. M. Rathore, A. Paul, A. Ahmad, M. Anisetti, and G. Jeon, "Hadoop-Based Intelligent Care System (HICS): Analytical Approach for Big Data in IoT," *ACM Trans. Internet Technol.*, vol. 18, no. 1, Nov. 2017, place: New York, NY, USA Publisher: Association for Computing Machinery.

[106] V. Grimblatt, G. Ferre, F. Rivet, C. Jego, and N. Vergara, "Precision agriculture for small to medium size farmers – an IoT approach," in *Proc. of 2019 IEEE ISCAS*. IEEE, May 2019, place: Sapporo, Japan.

[107] J. Gai, L. Tang, and B. L. Steward, "Automated crop plant detection based on the fusion of color and depth images for robotic weed control," *Journal of Field Robotics*, vol. 37, no. 1, pp. 35–52, 2020.

[108] M. Friedli, N. Kirchgessner, C. Grieder, F. Liebisch, M. Mannale, and A. Walter, "Terrestrial 3D laser scanning to track the increase in canopy height of both monocot and dicot crop species under field conditions," *Plant Methods*, vol. 12, no. 1, p. 9, Jan. 2016.

[109] Y. Hu, L. Wang, L. Xiang, Q. Wu, and H. Jiang, "Automatic non-destructive growth measurement of leafy vegetables based on Kinect," *Sensors (Basel)*, vol. 18, no. 3, Mar. 2018.

[110] F. Golbach, G. Kootstra, S. Damjanovic, G. Otten, and R. van de Zedde, "Validation of plant part measurements using a 3D reconstruction method suitable for high-throughput seedling phenotyping," *Machine Vision and Applications*, vol. 27, no. 5, pp. 663–680, Jul. 2016.

[111] J. C. Rose, S. Paulus, and H. Kuhlmann, "Accuracy analysis of a multi-view stereo approach for phenotyping of tomato plants at the organ level," *Sensors (Basel)*, vol. 15, no. 5, pp. 9651–9665, Apr. 2015, publisher: MDPI AG.

[112] J. L. Araus and J. E. Cairns, "Field high-throughput phenotyping: the new crop breeding frontier," *Trends Plant Sci.*, vol. 19, no. 1, pp. 52–61, Jan. 2014, publisher: Elsevier BV.

[113] A. Hartmann, T. Czauderna, R. Hoffmann, N. Stein, and F. Schreiber, "HTPheno: an image analysis pipeline for high-throughput plant phenotyping," *BMC Bioinformatics*, vol. 12, no. 1, p. 148, May 2011, publisher: Springer Science and Business Media LLC.

[114] A.-K. Mahlein, "Plant disease detection by imaging sensors - parallels and specific demands for precision agriculture and plant phenotyping," *Plant Dis.*, vol. 100, no. 2, pp. 241–251, Feb. 2016, publisher: Scientific Societies.

[115] R. Benadjila, L. Khati, and D. Vergnaud, "Secure storage–Confidentiality and authentication," *Computer Science Review*, vol. 44, p. 100465, May 2022.

[116] S. More and S. Chaudhari, "Third Party Public Auditing Scheme for Cloud Storage," *Procedia Computer Science*, vol. 79, pp. 69–76, Jan. 2016.

[117] R. A. Nafea and M. A. Almaiah, "Cyber Security Threats in Cloud: Literature Review," in *2021 International Conference on Information Technology (ICIT)*. IEEE, Jul. 2021, pp. 779–786.

[118] F. M. Awaysheh, M. N. Aladwan, M. Alazab, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by Design for Big Data Frameworks Over Cloud Computing," *IEEE Transactions on Engineering Management*, vol. 69, no. 6, pp. 3676–3693, Dec. 2022, publisher: Institute of Electrical and Electronics Engineers Inc.

[119] D. Balouek-Thomert, E. G. Renart, A. R. Zamani, A. Simonet, and M. Parashar, "Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows," *The International Jour-*

nal of High Performance Computing Applications, vol. 33, no. 6, pp. 1159–1174, 2019, _eprint: https://doi.org/10.1177/1094342019877383.

[120] Y. Zhao, W. Wang, Y. Li, C. Colman Meixner, M. Tornatore, and J. Zhang, "Edge Computing and Networking: A Survey on Infrastructures and Applications," *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019, conference Name: IEEE Access.

[121] A. Orive, A. Agirre, H.-L. Truong, I. Sarachaga, and M. Marcos, "Quality of Service Aware Orchestration for Cloud–Edge Continuum Applications," *Sensors*, vol. 22, no. 5, p. 1755, Feb. 2022, publisher: MDPI.

[122] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive Resource Efficient Microservice Deployment in Cloud-Edge Continuum," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1825–1840, Aug. 2022, publisher: IEEE.

[123] A. Ullah, H. Dagdeviren, R. C. Ariyattu, J. DesLauriers, T. Kiss, and J. Bowden, "MiCADO-Edge: Towards an Application-level Orchestrator for the Cloud-to-Edge Computing Continuum," *Journal of Grid Computing*, vol. 19, no. 4, p. 47, Dec. 2021, publisher: Springer.

[124] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A Unified Model for the Mobile-Edge-Cloud Continuum," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–21, May 2019, publisher: ACM New York, NY, USA.

[125] P. Trakadas, N. Nomikos, E. T. Michailidis, T. Zahariadis, F. M. Facca, D. Breitgand, S. Rizou, X. Masip, and P. Gkonis, "Hybrid Clouds for Data-Intensive, 5G-Enabled IoT Applications: An Overview, Key Issues and Relevant Architecture," *Sensors*, vol. 19, no. 16, p. 3591, Aug. 2019, publisher: MDPI.

[126] F. Spinelli and V. Mancuso, "Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 596–630, 2021.

[127] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint Communication, Computation, Caching, and

Control in Big Data Multi-Access Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020, publisher: IEEE.

[128] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017, publisher: IEEE.

[129] P. Wang, J. Zhang, X. Zhang, Z. Yan, B. G. Evans, and W. Wang, "Convergence of Satellite and Terrestrial Networks: A Comprehensive Survey," *IEEE Access*, vol. 8, pp. 5550–5588, 2020, conference Name: IEEE Access.

[130] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues," *IEEE Network*, vol. 34, no. 3, pp. 224–231, May 2020, conference Name: IEEE Network.

[131] S. Hamdan, M. Ayyash, and S. Almajali, "Edge-Computing Architectures for Internet of Things Applications: A Survey," *Sensors*, vol. 20, no. 22, p. 6441, Jan. 2020, number: 22 Publisher: Multidisciplinary Digital Publishing Institute.

[132] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou, "Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1761–1804, 2020, conference Name: IEEE Communications Surveys & Tutorials.

[133] F. Rinaldi, H.-L. Maattanen, J. Torsner, S. Pizzi, S. Andreev, A. Iera, Y. Koucheryavy, and G. Araniti, "Non-Terrestrial Networks in 5G & Beyond: A Survey," *IEEE Access*, vol. 8, pp. 165 178–165 200, 2020, conference Name: IEEE Access.

[134] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017, conference Name: IEEE Access.

[135] B. Zolfaghari, G. Srivastava, S. Roy, H. R. Nemati, F. Afghah, T. Koshiba, A. Razi, K. Bibak, P. Mitra, and B. K. Rai, "Content Delivery Networks: State of the Art, Trends, and Future Roadmap," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 34:1–34:34, Apr. 2020.

[136] C. A. Ardagna, R. Asal, E. Damiani, T. Dimitrakos, N. El Ioini, and C. Pahl, "Certification-Based Cloud Adaptation," *IEEE TSC*, pp. 1–1, 2018.

[137] X.-H. Wu, J.-P. Li, and W. Yao, "A network security evaluation model based on common criteria," in *Proc. of IEEE ICACIA*. IEEE, 2008, pp. 416–420.

[138] G. Bossert and F. Guihery, "Security evaluation of communication protocols in common criteria," in *Proc. of IEEE ICC*, 2012.

[139] H. L. Mai, T. Nguyen, G. Doyen, R. Cogranne, W. Mallouli, E. M. de Oca, and O. Festor, "Towards a security monitoring plane for named data networking and its application against content poisoning attack," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan, Apr. 2018, pp. 1–9.

[140] N. L. M. van Adrichem and F. A. Kuipers, "NDNFlow: Software-defined Named Data Networking," in *Proc. of the 1st IEEE NetSoft*, Apr. 2015, pp. 1–5.

[141] Y. Zhou, J. Bi, T. Yang, K. Gao, J. Cao, D. Zhang, Y. Wang, and C. Zhang, "HyperSight: Towards Scalable, High-Coverage, and Dynamic Network Monitoring Queries," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1147–1160, Jun. 2020.

[142] A. Bialas, M. Michalak, and B. Flisiuk, "Anomaly Detection in Network Traffic Security Assurance," in *Proc. of DepCoS-RELCOMEX*, ser. Advances in Intelligent Systems and Computing, W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, Eds. Cham: Springer International Publishing, 2020, pp. 46–56, event-place: Cham.

[143] K. Löfgren and C. W. R. Webster, "The value of Big Data in govern-

ment: The case of 'smart cities'," *Big Data & Society*, vol. 7, no. 1, p. 205395172091277, Jan. 2020, publisher: SAGE Publications Ltd.

[144] F. Berto, C. Ardagna, M. Torrente, D. Manenti, E. Ferrari, A. Calcante, R. Oberti, C. Fra', and L. Ciani, "A 5G-IoT enabled Big Data infrastructure for data-driven agronomy," in *2022 IEEE Globecom Workshops (GC Wkshps)*. Rio de Janeiro, Brazil: IEEE, Dec. 2022, pp. 588–594.

[145] S. A. Lokhande, "Effective use of Big Data in Precision Agriculture," in *Proc. of 2021 IEEE ESCI*, Mar. 2021, pp. 312–316.

[146] C. Mi, X. Peng, L. Peng, C. Zhao, and X. Deng, "Research on Crop Disaster Stress Risk Mapping System Based on Agriculture Big Data," in *Proc. of 2021 ICEITSA*, Dec. 2021, pp. 525–530.

[147] J. C. S. D. Anjos, K. J. Matteussi, P. R. R. D. Souza, G. J. A. Grabher, G. A. Borges, J. L. V. Barbosa, G. V. Gonzalez, V. R. Q. Leithardt, and C. F. R. Geyer, "Data Processing Model to Perform Big Data Analytics in Hybrid Infrastructures," *IEEE Access*, vol. 8, pp. 170 281–170 294, 2020, publisher: Institute of Electrical and Electronics Engineers Inc.

[148] M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero, "A Methodology for Non-Functional Property Evaluation of Machine Learning Models," in *Proceedings of the 12th International Conference on Management of Digital EcoSystems*, ser. MEDES '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 38–45.

[149] C. Tankard, "Big data security," *Network Security*, vol. 2012, no. 7, pp. 5–8, Jul. 2012.

[150] G. D. Samaraweera and J. M. Chang, "Security and Privacy Implications on Database Systems in Big Data Era: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 1, pp. 239–258, Jan. 2021, publisher: IEEE Computer Society.

[151] D. Yadav, D. H. Maheshwari, and D. U. Chandra, "Big Data Hadoop: Security and Privacy," *SSRN Journal*, 2019.

[152] M. Khan and M. D. Ansari, "Security and privacy issue of big data over

the cloud computing: A comprehensive analysis," *International Journal of Recent Technology and Engineering*, vol. 7, no. 6, pp. 413–417, 2019.

[153] E. Bertino and E. Ferrari, "Big Data Security and Privacy," in *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*. Springer, 2018.

[154] L. Zhou, A. Fu, S. Yu, M. Su, and B. Kuang, "Data integrity verification of the outsourced big data in the cloud environment: A survey," *Journal of Network and Computer Applications*, vol. 122, pp. 1–15, Nov. 2018.

[155] J. Koo, G. Kang, and Y.-G. Kim, "Security and Privacy in Big Data Life Cycle: A Survey and Open Challenges," *Sustainability*, vol. 12, no. 24, p. 10571, Dec. 2020.

[156] K. S, Y. S, and R. V. P, "An evaluation on big data generalization using k-Anonymity algorithm on cloud," in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, Jan. 2015, pp. 1–5.

[157] S.-B. Jang, "A study of performance enhancement in big data anonymization," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, vol. 2018-January. IEEE, Aug. 2017, pp. 1–4.

[158] H. Kupwade Patil and R. Seshadri, "Big Data Security and Privacy Issues in Healthcare," in *Proc. of IEEE Big Data 2014*, Anchorage, AK, USA, Jun. 2014.

[159] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo, "Protection of Big Data Privacy," *IEEE Access*, vol. 4, pp. 1821–1834, 2016.

[160] M. Elsayed and M. Zulkernine, "Towards Security Monitoring for Cloud Analytic Applications," in *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, May 2018, pp. 69–78.

[161] M. Sameen and S. O. Hwang, "TIMPANY - deTectIon of Model Poison-

ing Attacks usiNg accuracY," *IEEE Access*, vol. 9, pp. 139 415–139 425, 2021, publisher: Institute of Electrical and Electronics Engineers Inc.

[162] G. Ra, D. Kim, D. Seo, and I. Lee, "A Federated Framework for Fine-Grained Cloud Access Control for Intelligent Big Data Analytic by Service Providers," *IEEE Access*, vol. 9, pp. 47 084–47 095, 2021, publisher: Institute of Electrical and Electronics Engineers Inc.

[163] P. Zhang, X. Zhou, W. Li, and J. Gao, "A Survey on Quality Assurance Techniques for Big Data Applications," in *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, Apr. 2017, pp. 313–319.

[164] D. Lee, "Big Data Quality Assurance Through Data Traceability: A Case Study of the National Standard Reference Data Program of Korea," *IEEE Access*, vol. 7, pp. 36 294–36 299, 2019.

[165] V. Casola, A. D. Benedictis, S. D. Martino, N. Mazzocca, and L. L. L. Starace, "Security-Aware Deployment Optimization of Cloud–Edge Systems in Industrial IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 724–12 733, Aug. 2021.

[166] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[167] K. Jedlička and K. Charvát, "Visualisation of Big Data in Agriculture and Rural Development," in *Proc. of 2018 IST-Africa*, May 2018, pp. Page 1 of 8–Page 8 of 8.

[168] J.-c. Zhao and J.-x. Guo, "Big data analysis technology application in agricultural intelligence decision system," in *Proc. of 2018 IEEE ICCCBDA*, Apr. 2018, pp. 209–212.

[169] M. N. Islam Sarker, M. Wu, B. Chanthamith, S. Yusufzada, D. Li, and J. Zhang, "Big Data Driven Smart Agriculture: Pathway for Sustainable Development," in *Proc. of 2019 IEEE ICAIBD*, May 2019, pp. 60–65.

[170] B. Gokaraju, S. Samiappan, and Y. S. Kale, "Big Data Fusion Chal-

lenge: Unmanned Aerial System Based Precision Agriculture," in *Proc. of 2021 IEEE ICETCI*, Aug. 2021, pp. 77–81.

[171] M. Dholu and K. Ghodinde, "Internet of Things (IoT) for Precision Agriculture Application," in *Proc. of 2018 IEEE ICOEI*, May 2018, pp. 339–342.

[172] D. K. Singh and R. Sobti, "Role of Internet of Things and Machine Learning in Precision Agriculture: A Short Review," in *Proc. of 2021 IEEE ISPCC*. Solan, India: IEEE, Oct. 2021, pp. 750–754.

[173] L. G. Rios and J. A. I. Diguez, "Big Data Infrastructure for analyzing data generated by Wireless Sensor Networks," in *Proc. of 2014 IEEE BigData*, Jun. 2014, pp. 816–823.

[174] Y. Demchenko, F. Turkmen, C. de Laat, C. Blanchet, and C. Loomis, "Cloud based big data infrastructure: Architectural components and automated provisioning," in *Proc. of 2016 IEEE HPCS*, Jul. 2016, pp. 628–636.

[175] C. A. Ardagna, E. Damiani, and F. Berto, "Script Language Security," in *Encyclopedia of Cryptography, Security and Privacy*, S. Jajodia, P. Samarati, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Sep. 2023, pp. 1–3.

[176] M. Anisetti, F. Berto, and R. Bondaruc, "QoS-aware Deployment of Service Compositions in 5G-empowered Edge-Cloud Continuum," in *2023 IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 2023, pp. 471–478.

[177] F. Berto, L. Calderoni, M. Conti, and E. Losiouk, "Spatial bloom filter in named data networking: a memory efficient solution," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. Brno Czech Republic: ACM, Mar. 2020, pp. 274–277.