Designing Swarm-based Decentralised Systems: Requirements for Performance and Scalability

Abhinaba Chakraborty* Didier Colle* Mario Pickavet* Enrique Areizaga† Akis Kourtis‡
Andreas Oikonomakis‡ Adnan Imeri§ Wouter Tavernier*
*ID Lab, University of Ghent-imec, Ghent, Belgium †Tecnalia Research and Innovation, Spain
‡ NCSR Demokritos, Athens, Greece § Luxembourg Institute of Science and Technology, Luxembourg

***Abstract*—In the era of 5G and the upcoming 6G, current software stacks are ineffective when it comes to intelligent and fast decision-making, which necessitates the need for sustainable yet performant and scalable solutions. A significant amount of research has been done in this area, but an end-to-end solution was lacking. In this scope, we identify that we need a novel software stack which can handle the growing need for a vast amount of data generation with the help of the cloud-edge continuum, swarm programmability, along with secure deployments of applications. In this context, we propose OASEES, an architectural framework for a decentralised AI/ML computing stack that unifies diverse computing resources, peer-to-peer coordination protocols, and secure middleware. Our design outlines modular different compute infrastructures(CPUs, GPUs/TPUs and custom ASICs) orchestrated by a lightweight container-based runtime and governed by a blockchain-enabled tamper-proof data storage, decentralised coordination (decentralised autonomous organisation, voting, etc) to facilitate transparent discovery, allocation, and incentivization. We also propose a detailed performance and scalability metrics framework covering latency, throughput, resource utilisation, and cost-per-inference, intended as the foundation of subsequent evaluations.**

I. Introduction

The accelerating convergence of cloud computing, edge intelligence, and specialised AI hardware has catalysed a paradigm shift in how intelligent services are conceived, deployed, and managed across diverse industrial and societal sectors [1], [2]. This trend has opened unprecedented opportunities for enabling real-time, distributed intelligence across domains such as manufacturing, energy, healthcare, and smart cities. However, as digital systems scale and diversify, traditional, centrally governed infrastructures—designed around static provisioning and uniform control—are proving inadequate for meeting the demands of a sustainable, sovereign, and circular digital economy [3].

Specifically, emerging operational and regulatory requirements expose critical limitations in current architectures related to *data sovereignty* [4], *multi-tenant interoperability* [5], and *dynamic resource orchestration* [6]. Rigid and monolithic AI/ML pipelines lack the agility to adapt to fluctuating computational workloads, geographic distribution constraints, and evolving privacy and compliance regulations. These constraints present a growing barrier to strategic ambitions to retain control over its data assets and to promote digitally and environmentally responsible value chains.

Against this backdrop, the *OASEES project* (Open autonomous programmable cloud apps & smart sensors) aims to fundamentally rethink the architecture and operational models for distributed AI/ML systems. It envisions a federated, yet coherent, computing fabric that spans the cloud-edge continuum and is capable of adapting to diverse contexts through *decentralised coordination*, *swarm programmability*, and *policy-aware orchestration*. The approach abandons *one-size-fits-all* deployment strategies in favour of a modular, scalable framework [7].

A central tenet of the OASEES project is native support for *heterogeneous computing hardware*—from general-purpose CPUs and accelerators like GPUs/TPUs to domain-specific AI ASICs. Within this ecosystem, the discovery, allocation, and accounting of resources must be not only secure and transparent but also aligned with incentive mechanisms that ensure equitable stakeholder participation. Such capabilities directly support the stakeholders' data sovereignty directive while enabling the deployment of privacy-preserving, fault-tolerant, and energy-efficient AI services tailored to local and cross-border needs.

To realise this vision, OASEES proposes a modular decentralised AI/ML computing stack composed of three key pillars:

1) *Heterogeneous compute nodes* integrated with AI accelerators for adaptable workload execution;
2) A *lightweight container-based runtime* that orchestrates workload placement and lifecycle management across cloud and edge environments; and
3) A *blockchain-enabled system* that governs, decentralised autonomous organisation, voting and resource discovery, tamper-proof storage and incentive-driven governance, etc.

To support empirical validation and benchmarking of such decentralised architecture, we also propose a comprehensive performance and scalability metrics framework that includes latency, throughput, resource utilisation, scalability, fault-recovery overhead, and cost-per-inference. This evaluation strategy is designed to guide performance optimisations and facilitate cross-platform comparisons, ultimately contributing to a standards-based, federated AI infrastructure aligned with sustainable environmental and digital goals.

II. Related Works

The rapid evolution of IT and network infrastructures—accelerated by the advent of 5G and beyond—has catalysed the emergence of several platforms focused on the management and orchestration of edge infrastructure and services [?], [8]–[10]. These include proprietary, standardised, and open-source systems that address specific layers of the edge-to-cloud continuum. A variety of orchestration frameworks have emerged, including those aligned with cloud-native principles, NFV MANO frameworks [11], [12], Software Defined Networking (SDN) controllers [13], and edge-focused platforms for IoT [14]–[16]. Despite these advances, the current orchestration ecosystem remains fragmented and siloed, with most solutions optimised for either cloud, network, or device-level orchestration, rather than the full continuum [17]. Managing compute resources across the continuum-spanning from high-performance data centres to low-power edge devices. This requires efficient lifecycle orchestration. As outlined in [18], [19], this involves several critical phases: registration and discovery, scheduling and placement, runtime management and scaling, etc. OASEES adopts this lifecycle but advances it with an edge-first perspective [20]–[23], deploying models via lightweight containers to minimise startup latency and footprint. Its orchestrator balances conflicting objectives such as inference latency, network cost, and energy consumption.

Runtime scaling mechanisms integrate fine-grained telemetry—covering CPU, memory, power, and inference throughput—to trigger horizontal (replication-based) or vertical (resource augmentation) scaling.

Scalable decentralised architectures must accommodate multi-actor, multi-domain operations where resource ownership, policy, and trust boundaries vary. Several efforts—including the European Open Science Cloud (EOSC) [24]—are advancing this vision by federating independently managed infrastructures, offering marketplaces for data, services, and compute capabilities. Within OASEES, secure federation is enforced via trusted communication channels, resource certification, and runtime attestation. It supports monetisation and capability exchange across administrative domains, aligning with decentralised economic models such as data/token marketplaces and AI model licensing.

The scalability and performance of decentralised systems have been studied across dimensions such as latency, throughput, resource elasticity, membership growth, and network overhead [25]. For example, systems like EdgeX Foundry [26] and KubeEdge [27] have demonstrated horizontal scaling across edge clusters, though with challenges in consistent service discovery and data propagation. In terms of metrics, latency (e.g., time to respond to inference or control requests), throughput (requests per second), and scalability efficiency (performance as resources increase) are widely used [25]. Energy efficiency and cold-start latency (especially relevant for AI workloads) are gaining prominence [25]. Moreover, resource orchestration overhead, consistency guarantees, and state synchronisation times are critical when designing truly decentralised platforms.

III. Motivation

The OASEES project involves six use cases [7]. The main challenge of OASEES is not to be too case-specific. The main requirement of the project is to be use case agnostic, and we should still be able to map the stack to every use case.

One of the use cases concerns high-mast inspection using a swarm of drones. This use case enables high-mast inspection, leveraging a fully decentralised swarm architecture. The infrastructure and operation of this use case can scale dynamically to accommodate many (in the order of 100) drones, ensuring flexibility in adapting to varying requirements. The technologies involved for this use case are drones, NG-RAN, etc. There are several research projects which involve drones and inspection, such as *5G!Drones* [28], *Drones4Safety* [29]. The main criterion of this use case is to find faults in high masts. For that, we use some state-of-the-art object detection model which will be run in a drone fitted with power power-efficient AI accelerator. The main challenge here is to orchestrate the workload in such a way that we can have a faster inference of captured images, which will also be energy efficient.

Similarly, other use cases have their own performance and scalability requirements, which makes the performance evaluation complicated and makes taking a scalability decision harder. To this end, we provide an architecture which can be mapped to all the use cases and more and a performance and scalability metrics framework to evaluate such decentralised systems.

IV. Decentralized Architecture

OASEES adopts a decentralised-by-design paradigm for swarm intelligence, grounded in distributed ledger technology and blockchain, which enables the recording of immutable transactions and achieves consensus across all nodes. Smart contracts underpin the decentralised applications (DApps) [30], while Decentralised Autonomous Organisations (DAOs) orchestrate device-level proposals and votes for agile, automated governance. A cloud-native CI/CD pipeline facilitates rapid development, testing and edge deployment of intelligent services. Security is reinforced by a dedicated Identity Layer—a *Self-Sovereign Identity* (SSI) model [31] that manages DAO membership and access via verifiable credentials, thereby preserving privacy and enforcing distinct security zones. The OASEES framework involves the following different actors with associated roles,

1) *Infrastructure Providers (IP)* supply and orchestrate devices—whether cloud, network or edge—for OASEES to build atop.
2) *Specialists* process OASEES data or engage with DApps via smart contracts, enabling human-in-the-loop decision-making.
3) *Developers* consume services to create OASEES workflows or author DApps and smart contracts for DAO deployment.
4) *Data Consumers* (OASEES users) utilise service outputs—e.g. patients accessing support applications.
5) *OASEES Service Providers* (swarm operators) manage the portal, coordinate underlying infrastructure and deliver DApps to users.

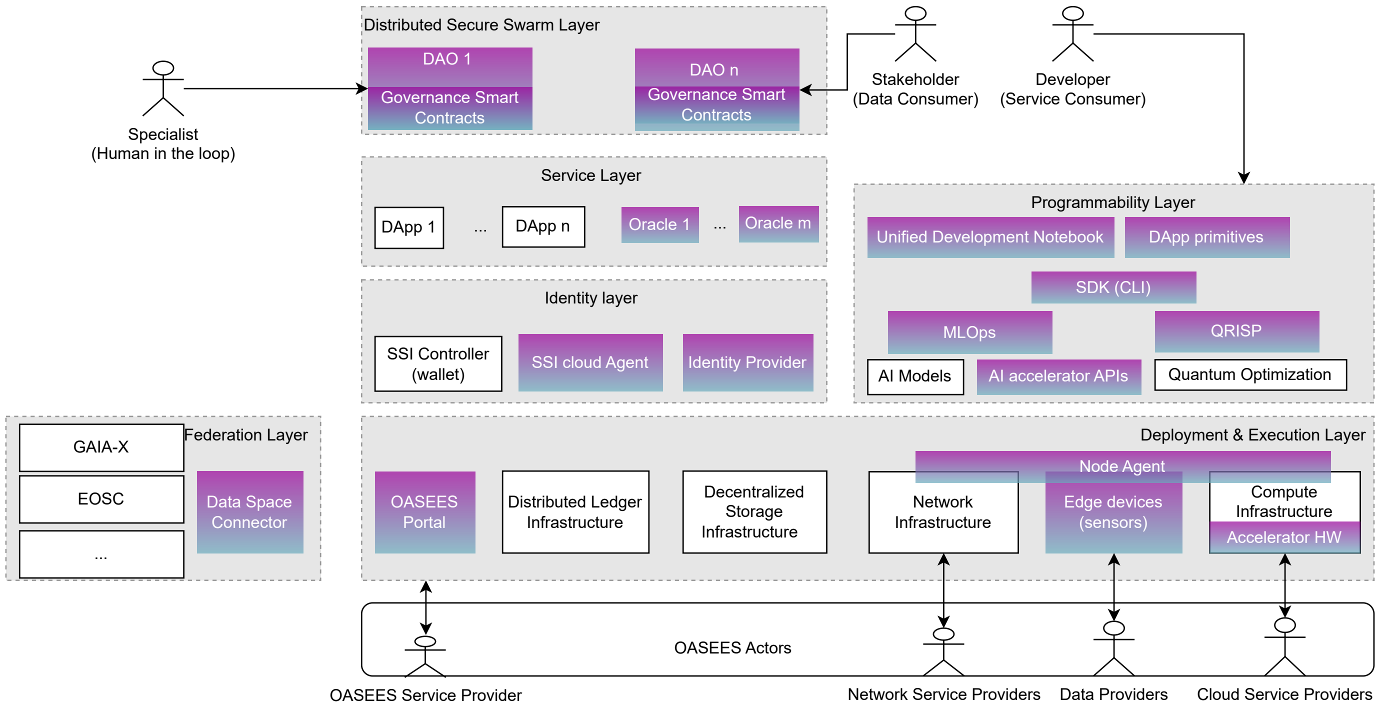

Fig. 1: Key Layers of OASEES Architecture

The OASEES architecture leverages decentralised infrastructure by integrating swarm intelligence, distributed ledger technology (DLT), and blockchain to enable secure data exchange and collaboration. It adopts a layered, modular, and extensible design to streamline interactions among the core components of the system, allowing for the seamless addition of new services and APIs. Fig-1 shows the OASEES architecture in detail.

A. Deployment and Execution Layer

At the foundation of OASEES lies the *Deployment and Execution Layer*, which encompasses the physical edge and cloud infrastructure—including different kinds of sensors, accelerator hardware, networking, storage, and *Distributed Ledger Technologies (DLT)* systems—along with their associated management functionalities. This layer also includes the OASEES portal, which serves as the primary access point for provisioning swarm services. Infrastructure orchestration facilitates the coordinated management of heterogeneous IoT–edge–cloud environments, while OASEES agents deployed on edge nodes and smart devices support efficient resource allocation.

B. Federation Layer

Above the deployment layer, the *Federation Layer* enables multi-instance deployments within federated cloud service and data space frameworks. Aligned with Gaia-X principles, it bridges infrastructure and data ecosystems, supporting regulatory compliance, federated governance, and secure data exchange. OASEES operates as both an infrastructure and data service provider, employing edge-aware brokerage mechanisms to enhance collaboration and resource sharing. Its integration with the European Open Science Cloud (EOSC) allows it to contribute services and added value through monitoring tools and helpdesk support.

C. Identity Layer

The OASEES platform aims to establish a secure and trustworthy edge ecosystem through a decentralised approach, primarily leveraging Self-Sovereign Identity (SSI) technologies for a portable digital identity that relies on decentralised authorities. Central to this initiative are *Decentralised Identifiers* (DIDs) [32] and Verifiable Credentials (VCs), which ensure persistence, global resolvability, cryptographic verifiability, and decentralisation. DIDs, standardised by the W3C [33], allow for self-sovereign control over digital identities, providing a persistent and globally resolvable means of identification. VCs complement DIDs by enabling the cryptographic verification of credentials, ensuring that digital identities remain authentic and tamper-evident. Within the OASEES framework, the Identity Layer is fundamental in managing digital identities and access control, providing the necessary tools to create, manage, and verify these identities securely. This layer supports two SSI approaches, one for devices and another for human users, and the second one to cater to the unique requirements of IoT and edge devices as well as human actors. This dual approach ensures a resilient and interoperable identity infrastructure, facilitating decentralised interactions across the ecosystem.

D. Service Layer

The *Service Layer* provides the foundation for developing and deploying DApps, which comprise microservices and smart contracts. Microservices leverage traditional cloud and edge infrastructures, whereas smart contracts operate on DLT platforms. Oracles within this layer enable communication between smart contracts and external data sources, thereby bridging decentralised and conventional computing environments.

E. Distributed Swarm Layer

This layer focuses on the key OASEES concept of the Decentralised Autonomous Authority(DAO) [34] as a framework to regulate interaction between multiple parties, including humans and/or swarms of devices. The DAO, equipped with human-in-the-loop (HTTL) mechanisms, facilitates resource management incentives and decision making for swarm collaboration. Users interact with the DAO through digital wallets, token staking, voting on proposals and receiving rewards.

F. Programming Layer

Finally, the *Programming Layer* provides tools for developers, including support for *Unified Development Notebooks* and accelerator APIs. The architecture also anticipates integration with the Eclipse QRISP quantum development kit [35] for future deployment within DApps accessible via the OASEES portal. In short, it encompasses OASEES SDK, which includes CLI support, OASEES notebook, the Rapid Development Kit, MLOps, and external API support.

V. Compute, Network and Storage Infrastructure

The OASEES platform provides support for managing both virtual machines and containers at the network edge, enabling flexible and scalable edge computing capabilities. It is designed to work with a variety of platforms (*x86*,*Arm*), allowing deployments to be tailored to specific workload requirements, ranging from lightweight IoT devices to data-intensive edge analytics. A key feature of the platform is its orchestration layer, which automates application and virtual machine deployment, scaling, and lifecycle management. This automation reduces administrative complexity and ensures that infrastructure remains responsive under dynamic workload conditions. Cloud infrastructure in OASEES is managed primarily through Kubernetes (K8S) clusters [36]. To better support edge deployments, the platform leverages K3S [37], a lightweight Kubernetes distribution, and KubeEdge [27], an extension that brings Kubernetes orchestration to edge nodes. Additionally, support has been provided to abstract hardware-level complexities by providing a unified interface for edge devices, further simplifying device discovery and interaction. To streamline service deployment on Kubernetes-based clusters, the OASEES project extends the native Kubernetes API. This extension enables developers to efficiently deploy, manage, and scale services, while automation tools handle tasks such as service distribution, load balancing, and service discovery.
Network connectivity between OASEES clusters is achieved using Software-Defined Networking (SDN) and Software-Defined Wide Area Networking (SD-WAN) technologies. These approaches decouple control logic from hardware, enabling more flexible, programmable network management. Virtual networks can thus be tailored to meet the specific needs of running applications, providing isolation, security, and performance guarantees. The SDN/SD-WAN component of OASEES focuses on several core functionalities: 1) Utilises tools such as Kubernetes Node Feature Discovery (NFD) and custom scripts to identify hardware/software capabilities. 2) Captures metrics including CPU, memory, storage, and network interfaces for optimal service placement. 3) Exposes this information via the OASEES SDK, enabling developers to target deployments without low-level device knowledge. 4) Automatically configures edge node network settings, including IP addressing, routing protocols, and secure communication links. 5) Tailors network configurations for resource-constrained edge agents using low-power communication protocols such as MQTT [38] and 5G RedCap [39]. 6) Prioritises secure, bandwidth-efficient operation in power-sensitive environments. 7) Coordinates services and resource allocation across multiple node clusters (swarms). 8) Ensures high availability, fault--tolerance, and efficient workload distribution. 9) Maintains service continuity in the face of cluster partitioning or failure.
The OASEES stack is designed to integrate low-power IoT protocols such as 6LoWPAN, enhancing device connectivity in resource-constrained environments. Specifically, Low Rate Wireless Personal Area Networks (LR-WPAN), compliant with the IEEE 802.15.4-2003 standard, provide cost-effective, energy-efficient communication for sensor networks and other embedded systems. The 6LoWPAN protocol further extends this by assigning IPv6 addresses to individual nodes, thereby enabling direct Internet connectivity and scalable network management. Moreover, support for 5G New Radio Reduced Capability (NR RedCap) technology is integrated to facilitate reliable and low-latency communication for emerging applications, including drones, wearables, and industrial automation systems. By balancing throughput, latency, and power consumption, 5G RedCap provides robust connectivity for real-time data transmission and remote control operations in industrial IoT environments.
For OASEES, we use IPFS (InterPlanetary File System) [40], which is a peer-to-peer protocol that enables data storage, sharing, and retrieval across a decentralised network instead of relying on a central server. It ensures efficiency, security, and data integrity by assigning each file and its blocks a unique content identifier (CID) based on its contents. Any change to the file alters its CID, allowing version control and easy integrity checks. Data in IPFS is distributed across multiple nodes, eliminating single points of failure and ensuring availability even if some nodes go offline. IPFS complements OASEES by offloading large data storage while leveraging the blockchain's immutable ledger. This enhances DApps and DAOs with improved efficiency, cost savings, and transparency, especially for oracles. By storing file hashes on the blockchain, anyone can retrieve and verify data from IPFS. Additionally, ERC721 [41] tokens can reference metadata (images, videos, etc) stored in IPFS via CIDs, keeping tokens

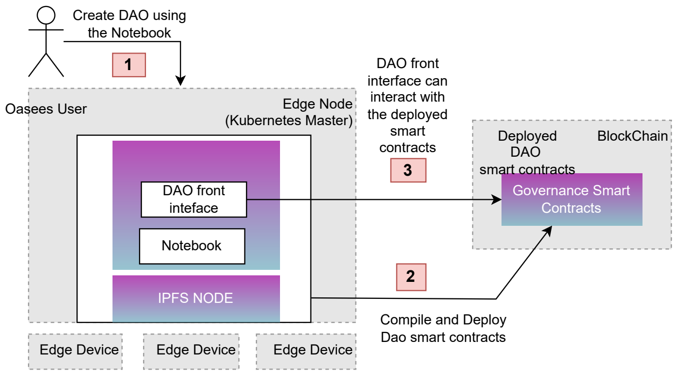
Create DAO using the Notebook
1
Oasees User
Edge Node (Kubernetes Master)
DAO front interface can interact with the deployed smart contracts
3
Deployed DAO smart contracts
BlockChain
DAO front inteface
Notebook
IPFS NODE
Governance Smart Contracts
2
Compile and Deploy Dao smart contracts
Edge Device
Edge Device
Edge Device

Fig. 2: DAO Creation

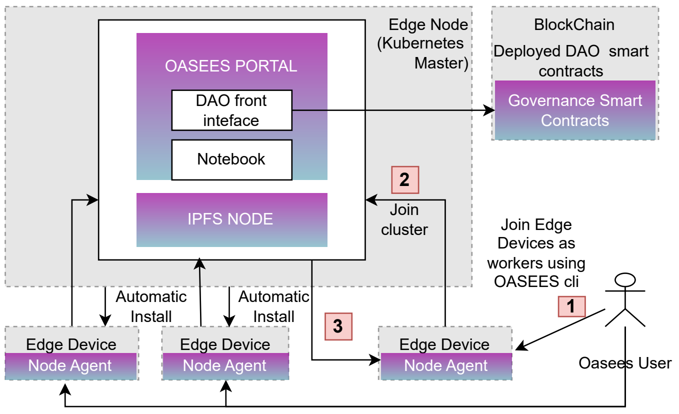

Fig. 3: Joining Workers

lightweight while linking to external data securely.

VI. Reference Use Case

To facilitate the identification and evaluation of system components, we propose a reference use case to elucidate the involved processes and the functionality of each layer of the OASEES platform. For this, we consider a swarm of entities that work collaboratively to accomplish a specific task. Each entity has a predefined role in the operation but retains the autonomy to act independently and participate in various events within the swarm system. To facilitate their involvement, the agents interact through an OASEES DAO, which enables them to cooperate in a trusted manner on top of a smart contract stack on DLT, without being limited by the number of participating entities or the scale of data involved. The entire process flow involving the OASEES platform stack is as follows,

A. DAO Creation

The creation of a DAO on the OASEES platform is a foundational step in enabling any swarm-based use case. This process begins with the user (admin) installing the OASEES Software Development Kit (SDK) on the designated master node and deploying the OASEES runtime stack on the target edge node. To support rapid onboarding, the system is designed for minimal setup time, ensuring a streamlined deployment process. Once the setup is complete, the administrator initiates the creation of a DAO, which subsequently governs the entire swarm system. The DAO logic is defined, compiled, and deployed using the *Unified Development Notebook*, a core component of the programmability layer. This environment provides seamless support for writing, testing, and deploying smart contracts that form the operational backbone of the DAO. Upon compilation and deployment to the blockchain, these smart contracts serve as the backend infrastructure for the DAO's interface and functionality, as depicted in Fig. 2. Several critical interfaces are established to ensure full integration and programmability within the OASEES ecosystem:

1) *DAO–Notebook Interface:* Enables developers to program, debug, and deploy DAOs and their associated smart contracts using the unified development environment.
2) *DAO–Portal Interface:* Acts as the primary user-facing interface to interact with the DAO, allowing administrators to add users, register devices, and manage permissions.
3) *DAO–Identity Provider Interface:* Supports authentication and verification of users and devices through integration with trusted identity providers.
4) *DAO–DApp Interface:* Connects the DAO to its constituent decentralised applications (DApps), which can be implemented as on-chain smart contracts or off-chain cloud/microservices (e.g., RESTful web services).
5) *DAO–Oracle Interface:* Facilitates interaction between on-chain and off-chain environments, enabling data exchange via oracles to incorporate external information into smart contract execution.

B. Joining Workers

Once DAO is created in the portal, edge nodes must be added to it. This requires edge nodes to be added as worker nodes to the Kubernetes platform. For this, the OASEES SDK CLI utility is installed on each edge device, which joins the master node using the typical Kubernetes worker node provisioning procedure. When the devices join the cluster, the node agent utilities of OASEES are installed automatically. This is illustrated in Fig-3.

C. Adding Edge Devices as DAO Members

Once edge devices have been added as worker nodes to the Kubernetes cluster, the devices are visible from the OASEES portal front interface, and users can onboard them as DAO members and provide them with voting tokens by performing a transaction to the smart contract of the DAO. That is responsible for distributing voting tokens. (Fig-4)

D. Training ML Models

Given the number of worker nodes, as well as a decentralised storage infrastructure (IPFS), users can now train models using data collected from edge devices. They can tweak hyperparameters through the Unified notebook interface in the OASEES stack or transfer learning through a pre-trained

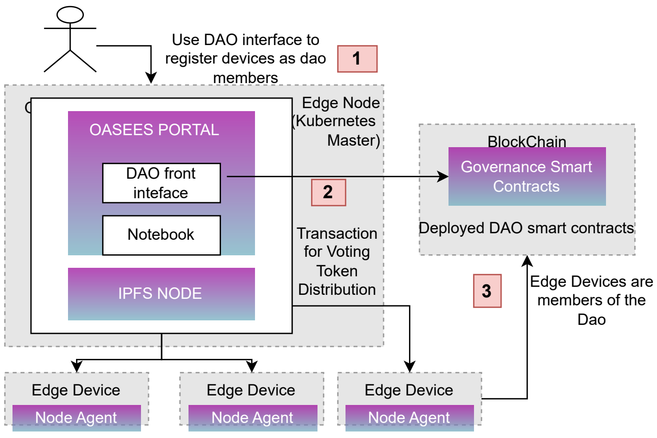
Use DAO interface to register devices as dao members
1
Edge Node (Kubernetes Master)
OASEES PORTAL
DAO front inteface
Notebook
IPFS NODE
2
BlockChain
Governance Smart Contracts
Deployed DAO smart contracts
Transaction for Voting Token Distribution
3
Edge Devices are members of the Dao
Edge Device
Node Agent
Edge Device
Node Agent
Edge Device
Node Agent

Fig. 4: Adding DAO Members

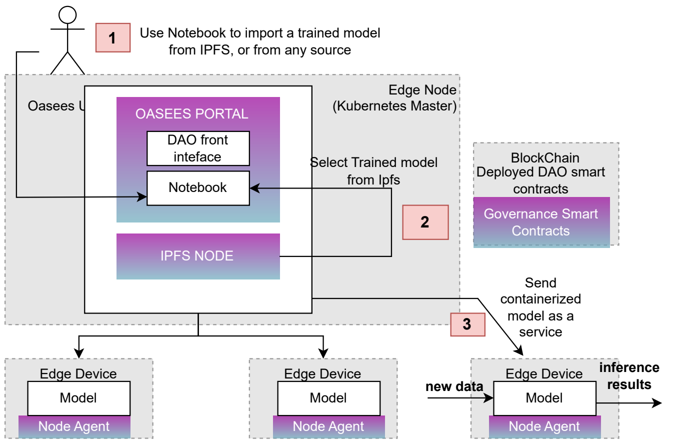
1
Use Notebook to import a trained model from IPFS, or from any source
Oasees U
Edge Node (Kubernetes Master)
OASEES PORTAL
DAO front inteface
Notebook
IPFS NODE
Select Trained model from Ipfs
2
BlockChain Deployed DAO smart contracts
Governance Smart Contracts
3
Send containerized model as a service
Edge Device
Model
Node Agent
Edge Device
Model
Node Agent
new data
Edge Device
Model
Node Agent
inference results

Fig. 6: Deploying ML Model

model. Both approaches are supported by the Unified Notebook and OASEES SDK. This is illustrated in Fig-5. Primarily, the training is supported by two popular frameworks, *PyTorch* and *TensorFlow*. Through the OASEES SDK, the primary support is one or multi-GPU training and federated learning. Complicated training methods such as FSDP [42] are not yet supported.

E. Deploy ML Model

Once training is complete, users can deploy the model on an edge accelerator platform using the SDK and deployment interface. The platform could be anything from x86 to ARM, but the OASEES SDK abstracts the complexities, so users don't need to worry about platform-specific issues details. After the deployment, edge devices can process real-time data input and perform inference on the data. Those inferences are collected in real-time and can be stored on IPFS. (Fig-6)

F. DAO Interaction

The DAO and its core smart contract counterparts can enable the voting procedure between users and devices. Users are notified of the proposal's existence and their status through the DAO front interface of the OASEES portal, and devices are notified by the event emitter, which is implemented in the DAO smart contracts. (Fig-7)
A typical reference workflow interacting with the DAO might be the following,

1) Pre-train a simple CNN classifier (ResNet50 [43], YoloV8n [44] etc) with the any image dataset (for example MNIST [45], ImageNet [46] etc.).
2) Use OASEES unified notebooks to deploy them on edge devices.
3) Gather images from different sensors.
4) Model produces output predictions with a confidence level
5) If the confidence level is below a predefined threshold for a specific sample, it is stored to IPFS and a proposal is created that contains the $ipfs\_hash$ of the specific sample and the prediction of the specified device.
6) Other devices fetch the sample using $ipfs\_hash$ and perform inference on this specific sample and cross-check with predictions.

VII. Performance Metrics for OASEES Stack

Initially, the OASEES stack and SDK must be set up quickly and efficiently, supporting a multi-node configuration that allows the Unified Notebook to execute tasks rapidly.

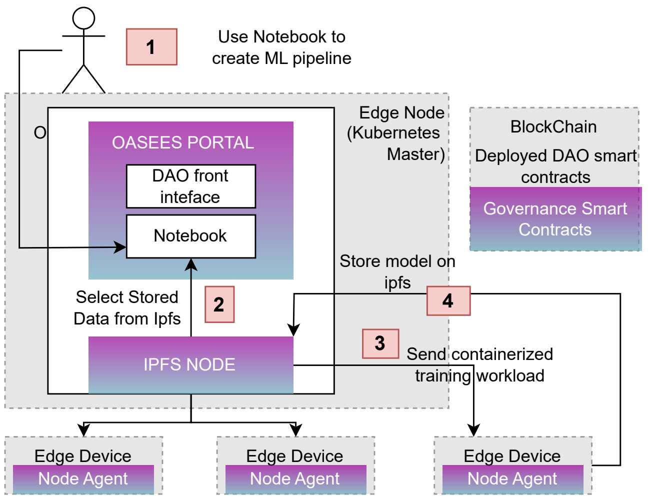
1
Use Notebook to create ML pipeline
Edge Node (Kubernetes Master)
O
OASEES PORTAL
DAO front inteface
Notebook
Select Stored Data from Ipfs
2
IPFS NODE
BlockChain
Deployed DAO smart contracts
Governance Smart Contracts
Store model on ipfs
4
3
Send containerized training workload
Edge Device
Node Agent
Edge Device
Node Agent
Edge Device
Node Agent

Fig. 5: Training ML Model

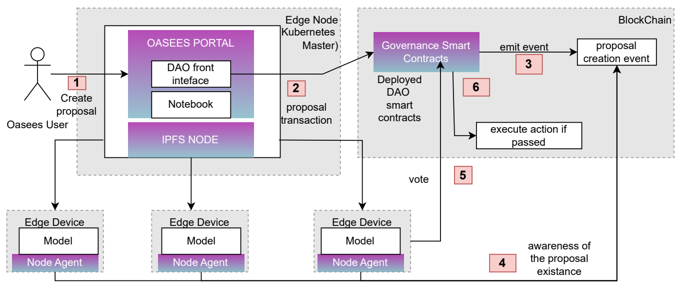

Fig. 7: DAO Interactions

Performance Metrics	Definitions
Proposal latency	Time taken to process proposals
Proposal throughput	Number of proposals processed per second
Participation rate	Percentage of stakeholders voting
Decentralisation	Distribution of voting power
Transaction latency	Time taken to finalise a transaction
Transaction throughput	Number of transactions per second
Finalisation rate	Speed of reaching final decisions
On-boarding time	Time required to onboard a device
Reporting time	Time taken to report device issues
Data Access Time	Time taken to retrieve data
Resource Utilisation	Efficiency of resource use

TABLE I: Performance Metrics for Distributed Swarm Layer

At the same time, the underlying blockchain must scale to handle a high volume of nodes and requests. Additionally, the portal is required to synchronise with ongoing events—such as the addition of new nodes—and maintain a swift response time, even when managing multiple sessions for the same user. In the realm of decentralised machine learning, training should conclude within a reasonable timeframe and achieve acceptable accuracy, while continuously monitoring edge devices for resource usage is essential. Once deployed, nodes are expected to gather real-time data and perform inference immediately, with the results stored in IPFS to ensure both prompt data retrieval and high availability. Finally, DAO operations, including voting and proposal management, must be robust enough to support high concurrent user activity and large data volumes, ensuring that the network can efficiently handle increasing traffic. Key sub-points include efficient resource management, rapid synchronisation, and real-time processing.

A. Distributed Secure Swarm Layer

A swarm of agents collectively performing tasks that typically require some form of intelligence without central control is known as swarm intelligence. Swarm intelligence in the frame of OASEES is a process where a group of edge devices and, in specific cases, human specialists work together to achieve a common goal, using decentralised technology to enable participation, coordination, and governance. Swarm computing is characterised by the following attributes for OASEES, 1) *Self-organisation* in swarms refers to the process by which a group of simple agents or entities organises itself into complex, structured patterns or behaviours without any central control. Instead, these entities follow simple rules and interact locally with one another, leading to the emergence of global order or intelligent behaviour. 2) *Decentralisation* in swarms refers to the lack of a central controlling entity or coordinator within the system. Rather than having a simple entity make decisions for the entire system, each agent participates in the decision process. This swarm computing system, the absence of a centralised control point, allows for a robust and flexible architecture. Each node, whether a sensor, device, or computational unit, operates based on local rules. This local autonomy enables the swarm to self-organise and adapt to new tasks without waiting for instructions from a central authority, making it highly resilient to single-point failures and capable of continuous operations despite uncertainties or changes in the network. The Table-I summarises key performance metrics pertinent to the Distributed Swarm Layer. It encompasses proposal and transaction latencies and throughputs, which respectively measure the time to process proposals or finalise transactions and the number handled per second. Participation rate and decentralisation gauge stakeholder engagement and the evenness of voting power distribution. Finalisation rate indicates how swiftly decisions are reached, while onboarding and reporting times reflect the duration to onboard devices and report their issues. Finally, data access time and resource utilisation assess the efficiency of data retrieval and overall resource usage.

Performance Metrics	Definitions
latency	Time taken to serve one request
Throughput	Number of requests served in one second

TABLE II: Performance Metrics for Service Layer

B. Service Layer

The layer plays a pivotal role in the operational management of decentralised applications (DApps) deployed over the heterogeneous and geographically distributed resources of the cloud-edge continuum. This layer acts as an intermediary abstraction that provides the necessary runtime environment and management capabilities for DApps, enabling dynamic deployment, lifecycle control, monitoring, and fault-tolerant operations. From a systems perspective, DApps require robust orchestration mechanisms to function effectively across diverse infrastructures, often characterised by varying latencies, bandwidth capacities, and computational resources. The Service Layer addresses these challenges by offering standardised APIs and toolchains that abstract the underlying complexity and heterogeneity of the infrastructure. It builds upon the foundational capabilities offered by the *Deployment and Execution Layer*, leveraging its functionalities to manage service placement, resource provisioning, and execution consistency across the continuum. Furthermore, the Service Layer ensures seamless orchestration of services by dynamically adapting to changes in resource availability, user demand, and network conditions. This dynamic adaptability is critical in edge-centric scenarios where nodes can be ephemeral or resource-constrained. In this context, latency and throughput emerge as essential performance indicators. Table-II provides a consolidated view of these key performance indicators.

C. Identity Later

The *Identity Layer* within the OASEES framework constitutes a foundational component for enabling secure and trustworthy interactions among heterogeneous entities across the cloud-edge-IoT continuum. It is responsible for the creation, management, and verification of digital identities, a critical capability in distributed systems where multiple human users, services, and IoT devices must interoperate seamlessly while preserving security and privacy.

In decentralised and edge-oriented environments, traditional identity and access management (IAM) models often fail to

Performance Metrics	Definitions
Latency	Time taken to process authentication
Cryptogenic Performance	Time to generate new DIDs for each agent

TABLE III: Performance Metrics for Identity Layer

meet the requirements for scalability, autonomy, and data sovereignty. To address these limitations, the Identity Layer in OASEES leverages *Self-Sovereign Identity* (SSI) paradigms, offering distinct implementations tailored to the characteristics of two key entity classes: human users and IoT/edge devices. For *human users*, SSI provides privacy-preserving identity management mechanisms, empowering users to control their credentials and selectively disclose information, in compliance with privacy regulations such as GDPR. For *devices*, the framework supports lightweight and autonomous identity provisioning methods, designed to accommodate the constrained computational and storage capabilities typical of edge and IoT nodes. This dual-SSI approach ensures that identity management remains flexible, secure, and context-aware, enabling fine-grained access control, secure service discovery, and robust authentication protocols across layers. Given the critical role of the Identity Layer in securing the system's operational integrity, it is necessary to assess its performance using relevant technical metrics. Table-III summarises key performance indicators for a representative identity management use case.

D. Deployment and Execution Layer

Resource management plays a critical role in the OASEES framework. The deployment and execution layer is responsible for orchestrating application services across a heterogeneous environment that involves IoT-to-edge-to-cloud infrastructure. They leverage mobility and quality of service (QoS) meta-information provided in the service description to perform tasks such as provisioning, upgrading, recovering, migrating and tear-down of application services are potentially driven by intelligence-enabled control loops. Edge computing infrastructure introduces additional challenges to orchestration, as the involved resources exhibit heterogeneity and volatility, and increases the number of deployment units that need to be managed. To this end, the *Deployment and Execution Layer* will be evaluated across twelve key performance metrics (Table-IV). Portal responsiveness is measured by loading time and RAM consumption, while user experience is further characterised by maximum session duration. AI hardware performance is assessed via inference latency and model deployment time, alongside overall AI model quality, encompassing accuracy, precision and recall. Resource efficiency is quantified through hardware utilisation, energy consumption during training and inference. Edge-level processing is monitored by device latency, and long-term maintainability is ensured via firmware update compatibility. Finally, network stability and capacity are tracked through connectivity reliability and bandwidth utilisation.

Performance Metrics	Definitions
Portal Loading Time	Time to load and synchronise the portal
User session duration	Ability to handle longer length of user session
Portal RAM Usage	Memory usage of the portal
AI hardware latency	Time to process an inference by AI hardware
Deployment time	time to deploy a DL process at hardware
AI model accuracy	Accuracy, precision and recall of the AI model.
Resource Usage	Hardware utilisation during training/inference
Energy consumption	Energy consumption due to training/inference
Edge device latency	delay incurred due to edge processing
Firmware support	Compatibility with newer updates
Network connectivity	Stability of the network connection
Network Bandwidth	Bandwidth usage of the network

TABLE IV: Performance Metrics for Deployment and execution Layer

E. Programmability Layer

We track five metrics that together capture both developer experience and runtime efficiency. First, *Notebook Loading Time* measures how long it takes for the unified development notebook to become fully interactive on an edge device. In typical settings, this ranges between 10s and 25s. These numbers are heavily dependent on system load and network stability.

Second, *Inference Time* captures the average latency from when an input is submitted until the model's prediction is returned. For example, across a suite of vision models, we observe mean inference times of 10ms to 30ms for lightweight models and 60ms to 100ms for more complex models. These numbers are very much dependent on the AI model in test, concurrency, system load, etc.

Third, *Model Warm-Up Time* denotes the preparatory interval from model instantiation through weight loading and runtime initialisation up to the first inference. In our experiments, warm-up times vary from 800ms on high-end edge servers to 1.5s on resource-constrained devices, reflecting the trade-off between portability and startup responsiveness.

Fourth, we quantify *Resource Utilisation*—the CPU, GPU, and memory footprint of inference operations—to ensure efficient hardware use.

Finally, *SDK Setup Time* encompasses the total duration required to install and configure the software development kit, deploy AI application artefacts, and initialise the model runtime. End-to-end, setup completes in roughly 250s to 500s, depending on network speed and edge device I/O speeds.

Performance Metrics	Definitions
Notebook loading time	Loading time of unified notebook in the agent's device
Inference time	Average inference time
Model Warm-Up time	Warm-up time for model deployment
Resource utilisation	Resource utilisation due to inference
SDK Setup time	Time required to set up SDK, AI App, and deploy model

TABLE V: Performance Metrics for Programmability Layer

Scalability Metrics	Definitions
Network Scalability	Increase in network traffic should be handled
Member growth rate	Continuous and sudden growth in membership
Horizontal Scalability	Increase performance in compute resources
Vertical scalability	Increase in number of computational resources

TABLE VI: Scalability Metrics for OASEES stack

VIII. Scalability Metrics for OASEES Stack

Table-VI summarises four key scalability metrics for decentralised systems. In OASEES, scalability is engineered as a multi-dimensional capability woven into every layer of the stack. At the network level, the system employs adaptive traffic steering and distributed service gateways so that as the volume of incoming requests grows, edge and cloud nodes collaboratively balance load without any single link or node becoming a bottleneck. This ensures that the end-to-end request latency remains stable even under surging demand.

At the membership level, OASEES is designed for dynamic peer participation. New nodes—whether user devices, IoT sensors, or micro-data centres—can join or leave the network seamlessly. A lightweight discovery protocol updates routing tables and service registries in real time, absorbing both gradual roll-outs and sudden spikes in participant numbers without manual reconfiguration.

From a horizontal or scale-out perspective, services in OASEES are packaged as containerised microservices orchestrated by a distributed scheduler. When compute requirements rise, the scheduler transparently spins up additional instances across available edge or cloud nodes, redistributing work and state via a shared, eventually consistent data layer. This approach lets you grow capacity simply by adding more machines, with minimal impact on running services.

Finally, vertical or scale-up strategies are built into each node's resource manager. When an individual node faces a heavy load—say, due to CPU-bound analytics or high storage I/O—the resource manager can hot-allocate extra CPU cores, memory, or local cache storage, drawing on reserved pools or bursting into nearby cloud resources. This on-the-fly resource augmentation keeps single-node performance high, even in the face of unpredictable workloads.

Together, these four system-level mechanisms—adaptive network traffic management, dynamic membership discovery, container-based horizontal expansion, and dynamic per-node resource augmentation—ensure that OASEES can grow and contract in response to real-world demands without sacrificing reliability, responsiveness, or operational simplicity.

IX. Conclusion

In this work, we present a framework for a decentralised system that manages a swarm. We then introduce a reference design for this system, along with performance and scalability measures to help evaluate it. As a next step, we will test the OASEES stack in real-world hardware using different use cases and report the results.

Acknowledgement

The research work presented in this article has been supported by the European Commission under the Horizon Europe Programme and the *OASEES* project (no. 101092702).

References

[1] N. Ali, G. Aloi, F. De Rango, C. Savaglio, and R. Gravina, "Edge-cloud continuum driven industry 4.0," *Procedia Computer Science*, vol. 253, pp. 2586–2594, 2025, 6th International Conference on Industry 4.0 and Smart Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050925003266

[2] D. Zeng, N. Ansari, M.-J. Montpetit, E. M. Schooler, and D. Tarchi, "Guest editorial: In-network computing: Emerging trends for the edge-cloud continuum," *IEEE Network*, vol. 35, no. 5, pp. 12–13, 2021.

[3] D. Khalyeyev, T. Bureš, and P. Hnětynka, "Towards characterization of edge-cloud continuum," in *Software Architecture. ECSA 2022 Tracks and Workshops*, T. Batista, T. Bureš, C. Raibulet, and H. Muccini, Eds. Cham: Springer International Publishing, 2023, pp. 215–230.

[4] P. Hummel, M. Braun, M. Tretter, and P. Dabrock, "Data sovereignty: A review," *Big Data & Society*, vol. 8, no. 1, p. 2053951720982012, 2021. [Online]. Available: https://doi.org/10.1177/2053951720982012

[5] L. Shwartz, N. Ayachitula, M. Buco, G. Grabarnik, M. Surendra, C. Ward, and S. Weinberger, "It service provider's multi-customer and multi-tenant environments," in *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, 2007, pp. 559–566.

[6] D.-N. Vu, N.-N. Dao, W. Na, and S. Cho, "Dynamic resource orchestration for service capability maximization in fog-enabled connected vehicle networks," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1726–1737, 2022.

[7] Chochliouros, Ioannis P. and Kourtis, Michail Alexandros and Xilouris, George and Tavernier, Wouter and Areizaga Sanchez, Enrique and Anastassova, Margarita and Bolzmacher, Christian and Tcholtchev, Nikolay and Corsi, Antonello and Trakadas, Panagiotis and Millet, Marta and Xenakis, Christos and Imeri, Adnan and Bellesini, Francesco and DOstilio, Paride and Markakis, Albertos and Engin, Ihsan Bal and Litke, Antonis and Quarato, Lucrezia Maria and Cugat, Diego and Gardikis, Georgios and Zarakovitis, Charilaos and Bouilland, Stephane and Zaharis, Zaharias and Lessi, Christina and Arvanitozisis, Dimitrios and Spiliopoulou, Anastasia S., "OASEES : an innovative scope for a DAO-based programmable swarm solution, for decentralizing AI applications close to data generation locations," in *ARTIFICIAL INTELLIGENCE APPLICATIONS AND INNOVATIONS. AIAI 2023 IFIP WG 12.5 INTERNATIONAL WORKSHOPS*, Maglogiannis, I and Iliadis, L and Papaleonidas, A and Chochliouros, I, Ed., vol. 677. Springer International Publishing AG, 2023, pp. 91–105. [Online]. Available: http://doi.org/10.1007/978-3-031-34171-7_7

[8] W. M. Shbair, M. Steichen, J. Francois, and R. State, "Blockchain orchestration and experimentation framework: A case study of KYC," in *IEEE/IFIP Man2Block 2018 - IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan, Apr. 2018. [Online]. Available: https://inria.hal.science/hal-01939865

[9] E. Zeydan, J. Baranda, J. Mangues-Bafalluy, and Y. Turk, "Blockchain for network service orchestration: Trust and adoption in multi-domain environments," *IEEE Communications Standards Magazine*, vol. 7, no. 2, pp. 16–22, 2023.

[10] E. Zeydan, J. Baranda, J. Mangues-Bafalluy, Y. Turk, and S. B. Ozturk, "Blockchain-based service orchestration for 5g vertical industries in multicloud environment," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4888–4904, 2022.

[11] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[12] F. Z. Yousaf, V. Sciancalepore, M. Liebsch, and X. Costa-Perez, "Manoaas: A multi-tenant nfv mano for 5g network slices," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 103–109, 2019.

[13] F. Bannour, S. Souihi, and A. Mellouk, "Distributed sdn control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[14] G. Castellano, F. Esposito, and F. Risso, "A service-defined approach for orchestration of heterogeneous applications in cloud/edge platforms," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1404–1418, 2019.

[15] 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.

[16] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, 2019.

[17] L. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal Bernabe, S. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," 06 2018.

[18] G. Breiter and M. Behrendt, "Life cycle and characteristics of services in the world of cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 3:1–3:8, 2009.

[19] M. Kiran, M. Jiang, D. J. Armstrong, and K. Djemame, "Towards a service lifecycle based methodology for risk assessment in cloud computing," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 449–456.

[20] F. Guim, T. Metsch, H. Moustafa, T. Verrall, D. Carrera, N. Cadenelli, J. Chen, D. Doria, C. Ghadie, and R. G. Prats, "Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 571–582, 2022.

[21] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, Sep. 2019. [Online]. Available: https://doi.org/10.1145/3326066

[22] J. Taheri, S. Dustdar, A. Zomaya, and S. Deng, *AI/ML for Service Life Cycle at Edge*. Cham: Springer International Publishing, 2023, pp. 55–109. [Online]. Available: https://doi.org/10.1007/978-3-031-22155-2_3

[23] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "Qos and resource-aware security orchestration and life cycle management," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2978–2993, 2022.

[24] A. V. d. Almeida, M. M. Borges, and L. Roque, "The european open science cloud: A new challenge for europe," in *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM 2017. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3144826.3145382

[25] A. E. Gelfand, C. Smith, M. Colony, and C. Bowman, "Performance evaluation of decentralized estimation systems with uncertain communication," in *2009 12th International Conference on Information Fusion*, 2009, pp. 786–793.

[26] H. Saxena and K. Salem, "Edgex: Edge replication for web applications," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 1041–1044.

[27] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 373–377.

[28] H. Koumaras, G. Makropoulos, M. Batistatos, S. Kolometsos, A. Gogos, G. Xilouris, A. Sarlas, and M. Kourtis, "5g-enabled uavs with command and control software component at the edge for supporting energy efficient opportunistic networks," *Energies*, vol. 14, p. 1480, 03 2021.

[29] N. H. Malle, F. F. Nyboe, and E. S. M. Ebeid, "Onboard powerline perception system for uavs using mmwave radar and fpga-accelerated vision," *IEEE Access*, vol. 10, pp. 113 543–113 559, 2022.

[30] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dapps: Distributed applications for real-time inference and control in o-ran," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 52–58, 2022.

[31] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013718301217

[32] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, and M. Conti, "A survey on decentralized identifiers and verifiable credentials," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2025.

[33] W. Semantic, "World wide web consortium," *http://www.w3org/2001/sw/*. [Online]. Available: https://cir.nii.ac.jp/crid/1570009750192210688

[34] J. Han, J. Lee, and T. Li, "A review of dao governance: Recent literature and emerging trends," *Journal of Corporate Finance*, vol. 91, p. 102734, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0929119925000021

[35] R. Seidel, S. Bock, R. Zander, M. Petrič, N. Steinmann, N. Tcholtchev, and M. Hauswirth, "Qrisp: A framework for compilable high-level programming of gate-based quantum computers," 2024. [Online]. Available: https://arxiv.org/abs/2406.14792

[36] V. Medel, O. Rana, J. a. Bañares, and U. Arronategui, "Modelling performance & resource management in kubernetes," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*, ser. UCC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 257–262. [Online]. Available: https://doi.org/10.1145/2996890.3007869

[37] Y. Zhang, M. Saberi, M. Wang, and E. Chang, "K3s: Knowledge-driven solution support system," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9873–9874, 07 2019.

[38] S. Quincozes, T. Emilio, and J. Kazienko, "Mqtt protocol: Fundamentals, tools and future directions," *IEEE Latin America Transactions*, vol. 17, no. 09, pp. 1439–1448, 2019.

[39] X. Li, X. Xu, and C. Hu, "Research on 5g redcap standard and key technologies," in *2023 4th Information Communication Technologies Conference (ICTC)*, 2023, pp. 6–9.

[40] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. abs/1407.3561, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561

[41] S. Barrington and N. Merrill, "The fungibility of non-fungible tokens: A quantitative analysis of erc-721 metadata," 2022. [Online]. Available: https://arxiv.org/abs/2209.14517

[42] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li, "Pytorch fsdp: Experiences on scaling fully sharded data parallel," 2023. [Online]. Available: https://arxiv.org/abs/2304.11277

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[44] M. Yaseen, "What is yolov8: An in-depth exploration of the internal features of the next-generation object detector," 2024. [Online]. Available: https://arxiv.org/abs/2408.15857

[45] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.