Grant Agreement No.: 101070473
Call: HORIZON-CL4-2021-DATA-01
Topic: HORIZON-CL4-2021-DATA-01-05
Type of action: HORIZON-RIA

OPEN CALL FLUIDOS OVERVIEW

Revision: v.1.0

TABLE OF CONTENTS

Funded by Horizon Europe Framework Programme of the European Union

1 THE FLUIDOS COMPUTING CONTINUUM

At first sight, the computing continuum seems to be a reality today, without the necessity of big investments in terms of technology and research. For instance, many software applications already rely on multiple components that are installed and operated in different locations (e.g., data gathering at the far edge, data aggregation at the telco edge, deep data processing in the cloud), hence apparently already implementing a computing continuum.

This section highlights how the FLUIDOS approach to Computing Continuum (a.k.a., liquid computing) has three distinctive characteristics that are not matched by existing approaches.

1.1 DEPLOYMENT TRANSPARENCY

When an application composed of multiple microservices such as in Figure 1 is deployed in the current silos-based computing continuum, each component must be explicitly configured to land on a specific target, e.g., edge datacenter vs. cloud. The location of each component is therefore fixed and decided a-priori; no modifications are allowed with respect to the location of each different component, unless starting a new re-deployment phase. Consequently, any possible dynamic optimization that could be carried out at run-time is more complex, as it requires the presence of an overarching orchestrator that re-deploys all the required components in the new optimal location, which is something that does not exist with the current technology.

Instead, **the FLUIDOS intent-based interface guarantees that each micro-service is started in the best location and provides also dynamic optimizations if required.** Hence, DevOps have simplified operations with FLUIDOS, as all services leverage a single point of deployment and control, while the FLUIDOS "magic" will start the above services in the most appropriate location given the service requirements and the infrastructure status.

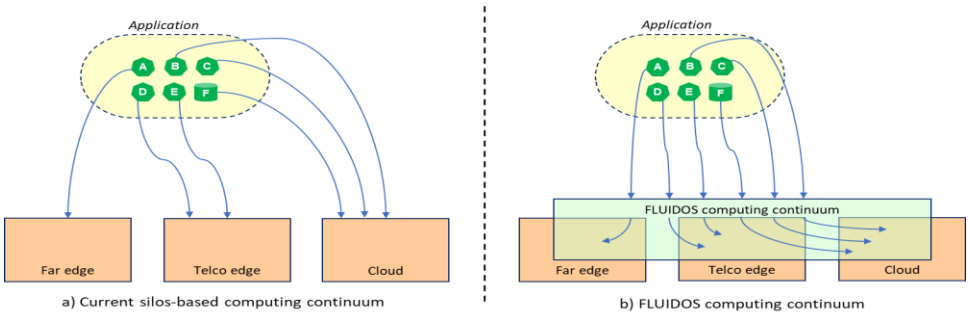


FIGURE 1. COMPUTING CONTINUUM: DEPLOYING APPS WITH TRADITIONAL APPROACH VS. FLUIDOS.

1.2 COMMUNICATION TRANSPARENCY

Funded by Horizon Europe Framework Programme of the European Union

The communication between different microservices is different whether they belong to the same communication space (e.g., the same Kubernetes cluster) or not. For instance, by default, all the communications inside a cluster are allowed, while all the communications from external components are forbidden. Furthermore, a service that accepts communications from inside the cluster requires primitives (e.g., the Kubernetes ClusterIP service) that are different from the ones used to accept data from outside the cluster (e.g., the Kubernetes NodePort or LoadBalancer services). Therefore, services must be explicitly configured to talk to each other based on their respective location, hence further complicating the actions required to carry out a possible re-deployment of the components mentioned above. It is worth mentioning that some technologies can partially overcome this problem, e.g., when communication among micro-services occurs through message brokers (e.g., technologies based on publish/subscribe primitives, such as Kafka). However, this requires all the micro-services to rely on the above communication primitives, which is not always possible because of the intrinsic characteristics of the pub/sub technology (e.g., cannot guarantee reduced latency), or because applications do not use this technology (e.g., they are based on HTTP or gRPC protocols).

The FLUIDOS approach to the computing continuum is visible in the right part of Figure 2: a virtual cluster spanning across multiple real clusters is envisioned, and applications are operating on the above virtual space. **With FLUIDOS, all communications between micro-services are mediated by the FLUIDOS virtual network fabric, which guarantees seamless communications independently from the location of each microservice**. Hence, the communication between two services belonging to the same virtual space happens as they were inside the same cluster, hence avoiding the necessity of complex and error-prone configurations no matter if they are deployed in the same cluster, or in two separate clusters.

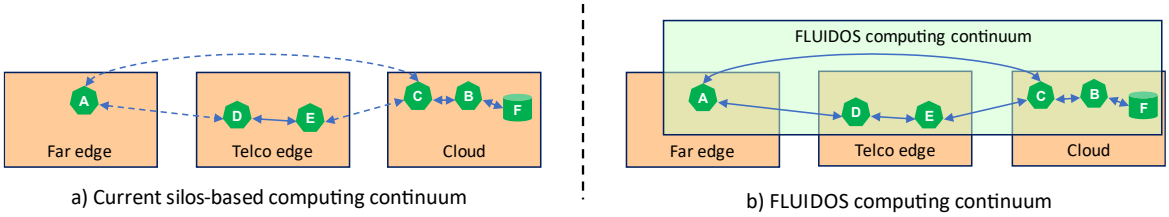


FIGURE 2. COMPUTING CONTINUUM: SERVICE-TO-SERVICE COMMUNICATIONS WITH TRADITIONAL APPROACH VS. FLUIDOS.

1.3 RESOURCE AVAILABILITY TRANSPARENCY

With the current technology, each microservice can use only the resources that are inside its own cluster. This statement holds for both normal operations (e.g., when the service is started), and when an update (e.g., automatic scaling) is requested. This behaviour prevents a service from using available resources located in other parts of the continuum, hence possibly ending up in a service disruption even in the presence of available resources, but located elsewhere in the continuum.

Funded by Horizon Europe Framework Programme of the European Union

This problem is not very important in cloud datacenters, where it is very unlikely to run into a lack of resources. Even in the case of small clusters obtained by acquiring only a small subset of the datacenter resources, the problem is irrelevant thanks to the capability to (automatically) resize the cluster, e.g., by adding/removing worker nodes to the given cluster. Instead, this problem is more important when considering a small pool of resources available at the edge, in which usually a few servers are available. The capability to acquire new physical resources, when needed, can only be achieved by leveraging other worker nodes possibly available in the closest vicinity.

FLUIDOS overcomes the above limitation by enabling the creation of a virtual computing space spanning across multiple physical domains, hence enabling a service (which has been started in the virtual space) to leverage all the resources belonging to the same virtual domain, independently from their physical location. Hence, in FLUIDOS a service can seamlessly scale based upon the availability of resources within the entire virtual infrastructure, e.g., ending up having one instance running in the telco edge, and another in the cloud datacenter, hence blurring the current rigid cluster boundaries.

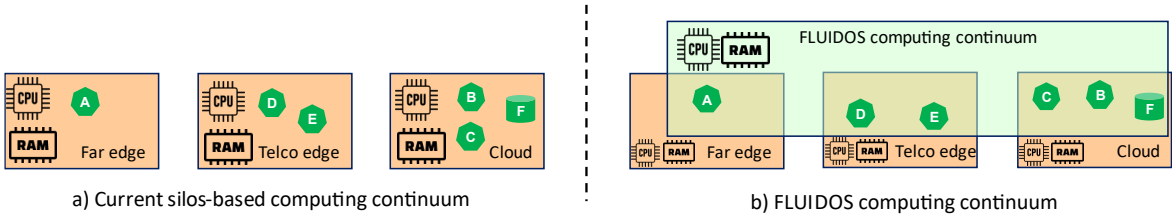


FIGURE 3. COMPUTING CONTINUUM: AVAILABLE RESOURCES WITH TRADITIONAL APPROACH VS. FLUIDOS.

Funded by Horizon Europe Framework Programme of the European Union

2 FLUIDOS ARCHITECTURE

This section presents the main interactions envisioned in FLUIDOS and the main components in the FLUIDOS architecture.

2.1 INTERACTIONS AMONG FLUIDOS NODES

FLUIDOS nodes interact with each other according to two dimensions, horizontal and vertical, building a virtual continuum that can be completely decentralized, without any single point of control. In the following, we present an initial definition of the high-level concepts inherent in horizontal and vertical interactions.

2.1.1 Horizontal interactions

The **horizontal** (east-west) interaction enables the creation of a fluid domain among peers, which can share their resources and services, or part of them, based upon a set of policies (e.g., enable sharing from node X but not from node Y). Horizontal interactions are carried out according to a peer-to-peer paradigm, hence without the need for any centralized entity that controls and supervises the entire process. Applications started on a FLUIDOS node can leverage any resource, either local or remote, available in the new virtual space. The FLUIDOS orchestrator is responsible for determining the best location for each component based on (i) the features requested through the intent-based API (e.g., computing power, maximum latency between components and toward end users, resiliency properties, number and location of replicas, reduction of energy and costs or carbon, etc.) and (ii) the additional policies set for the creation of the virtual space and/or the offloading of any component (e.g., no offloading on un-trusted domains for critical components).

The establishment of a peering relationship is a multi-step process, which mainly involves the following phases:

- **Discovery**, allowing to discover the existence of other peering candidates.
- **Negotiation**, enabling FLUIDOS nodes to issue requests for resources and/or services to other nodes, which in turn advertise their available capabilities.
- **Reservation and contract signing**, formalizing the acceptance of an offer from a FLUIDOS node, and possibly agreeing on compensations.
- **Peering**, establishing the virtual fabrics required to enable seamless workload offloading to remote nodes (e.g., networking, storage, …).
- **Usage**, leveraging the resources and services acquired by other FLUIDOS nodes to satisfy service requests.

Funded by Horizon Europe Framework Programme of the European Union

- **Depeering**, tearing down the previously established computing continuum abstractions, which are no longer necessary.

2.1.2 Vertical interactions

The **vertical** (north/south) interaction introduces new concepts such as aggregation and hierarchical scaling into the picture. A fluid domain can be created by a FLUIDOS "**supernode**" that aggregates multiple nodes, hence exporting a virtual space that is the union of the resources (and services, objects) of the composing nodes. The supernode-backed fluid domain is created with the same mechanisms already in place for the horizontal interactions; individual nodes are still in control of their own resources (ownership principle) and can withdraw them (if allowed by the signed contract). In addition, a special "tethered mode", available upon explicit configuration, allows individual nodes to be completely controlled by the supernode, giving up their local intelligence (and thus autonomy and ownership) in exchange for a simplified control plane. This mechanism, for instance, would be used to bring resource-constrained devices, such as micro-controllers, into the FLUIDOS ecosystem.

The aggregation paradigm can be applied recursively, with "supernodes" becoming part of a bigger "hyper-fluid domain", managed by an "hypernode" (or controlled, if the "tethered mode" is enabled), for a theoretically endless number of hierarchical levels. This provides the foundation for the hierarchical scaling property of FLUIDOS (recursive hierarchical architecture, from a single device to domains of domains). In this respect, the north/south interface, which enables interactions with different nodes (either local devices, clusters or other FLUIDOS "supernodes") all featuring the same interface, will be backed by novel resource aggregation algorithms that can aggregate the resources of each single node (normal, super, etc) in a scalable way.

Finally, a FLUIDOS Broker is an aggregation, consolidation, and brokering (i.e., reselling) point that supports also interactions between multiple administrative domains, whose duties include the capability to (1) observe the resources offered by many domains and the associated performance when involved in task offloading; (2) to monitor (and predict) the quality of network connections; (3) to suggest the "best node" when asked for an offloading request. However, given that super/hyper nodes include brokering activities albeit limited within the boundaries of a single admin domain, FLUIDOS will reuse for the above nodes the great part of algorithms (and software) developed for the Broker, assigning to this component a major role in the architecture.

2.2 ARCHITECTURE OVERVIEW

A FLUIDOS node builds on top of Kubernetes, which takes care of abstracting the underlying (physical) resources and capabilities in a uniform way, no matter whether dealing with single devices or full-fledged clusters (and the actual operating system) while providing at the same time standard interfaces for their

Funded by Horizon Europe Framework Programme of the European Union

consumption. Specifically, it properly extends Kubernetes with new control logic responsible for handling the different node-to-node interactions, as well as to enable the specification of advanced policies and intents (e.g., to constrain application execution), which are currently not understood by the orchestrator.

Given this precondition, the main architectural components of a FLUIDOS node are depicted in Figure 4 and converge around the **Node Orchestrator** and the **Available Resources** database. The former is in charge of orchestrating service requests, either on the local node or on remote nodes of the same fluid domain, coordinate all the interactions with local components (e.g., local scheduler) and remote nodes (e.g., to set up the computing/network/storage/service fabrics), and make sure that the service behaves as expected (e.g., honoring trust and security relationships). The latter keeps up-to-date information about resources and services available either locally or acquired from remote nodes, following the resource negotiation and acquisition process. Additional modules (and their companion communication interfaces), are required to handle the discovery of other FLUIDOS nodes and carry out the resource negotiation process, to monitor the state of the virtual infrastructure and to make sure that offloaded workloads/services behave as expected both in terms of security and negotiated SLAs, to take care of security and privacy issues (e.g., isolation), and to create the virtual continuum within the fluid space.

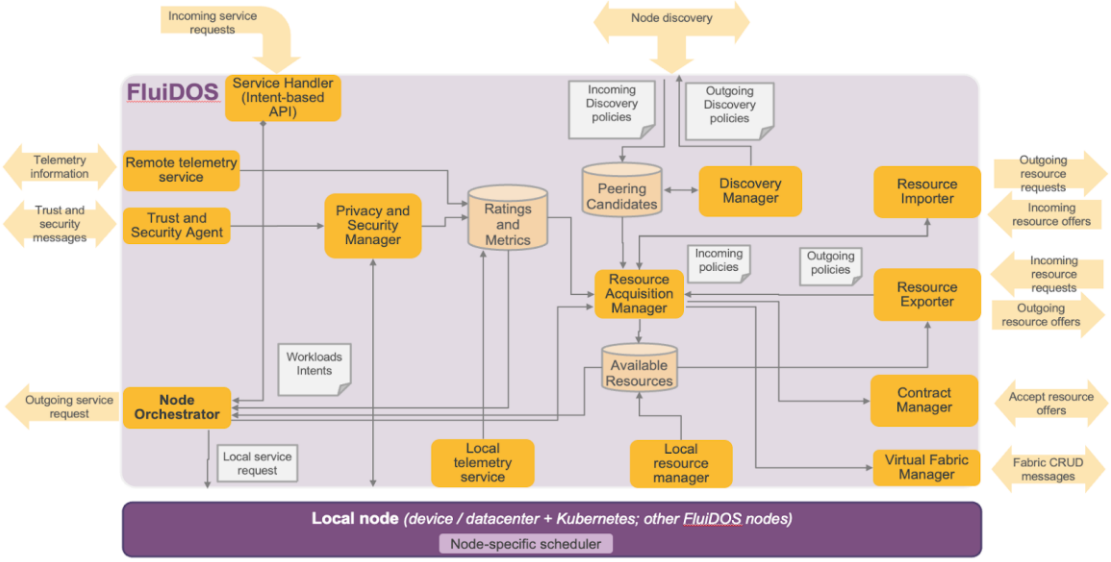


FIGURE 4. FLUIDOS ARCHITECTURE

The **discovery manager** is the module responsible for the discovery of other FLUIDOS nodes, producing as output a local database of feasible peering candidates. Specifically, each peering candidate is characterized by a globally unique identifier, the set of parameters necessary during the peering and resource acquisition phase (e.g., target network endpoints, …), as well as a set of distinguishing features (e.g., geographical location, whether it is available to sell computing resources, specific hardware functionalities or software services), and possibly including pricing/billing models. These features are expected to be exposed at a high level (e.g., through generic key/value labels); and enable both

Funded by Horizon Europe Framework Programme of the European Union

an initial policy-driven filtering (i.e., excluding undesired nodes from the list of peering candidates) and a priori filtering and ranking during the resource acquisition phase.

The **node orchestrator** is the FLUIDOS module responsible for the orchestration of the service requests, either on the local node, or offloading them to a remote FLUIDOS node, based on the current snapshot of the available resources database. Additionally, it interacts with and coordinates other components, mainly the resource acquisition manager, to trigger the acquisition of new resources and the setup of the appropriate network and storage fabrics enabling transparent execution continuum, in case already available ones are not sufficient to satisfy the incoming request.

The **resource acquisition manager** is the FLUIDOS module responsible for the negotiation process performed to acquire resources and services from remote FLUIDOS nodes. It can be triggered either proactively, based on policies, to ensure that a given amount of resources (with certain characteristics) is always available to fulfil foreseen future requests, or by the node orchestrator, reacting to the lack of matching resources to satisfy a service request.

The **virtual fabric manager** is the FLUIDOS component in charge of establishing the computing continuum abstractions to enable the seamless execution of workloads spread across multiple nodes. Specifically, it acts once a resource offer is accepted by the resource acquisition manager, and appropriately sets up the virtual node abstraction, along with the network and storage fabrics through the interaction with the remote node.

The **privacy and security manager** is the FLUIDOS module, investigated and detailed in section 2.6, which is in charge of guaranteeing the security of the different parties involved in the resource continuum, and it is intertwined with all the other different processes. This task is additionally fulfilled through the interaction with the trust and security agent, a trust anchor part of each FLUIDOS node that certifies the correctness of predefined operations.

The **telemetry service** is the FLUIDOS component responsible for the monitoring of the infrastructure (e.g., actual CPU load, memory, network, as well as possibly more detailed indexes such as memory page faults), including the collection of all the observability parameters key to enforce and verify the satisfaction of the workload requirements expressed through the intent-based API.

The **Cost Manager** is the FLUIDOS module responsible for evaluating the burdens of carrying out a computational load on a node. Hereby, the burdens can be both monetary and non-monetary.

2.2.1 Relevant Documentation

Additional details on the FLUIDOS architecture can be found in:
- **Deliverable 2.1**, available on the website here.
- Documentation on GitHub here.
- REAR protocol for resource Advertisement/Reservation on GitHub here.

Funded by Horizon Europe Framework Programme of the European Union

2.3 FLUIDOS AT THE EDGE

In many sectors (such as industrial control environment, smart city, smart agriculture, etc.), cloud computing has proven unprecedented cost-effectiveness and flexibility to users, nevertheless, it fails in all use cases where latency is important or where the bandwidth is an issue. For such a reason in the coming years, we expect to see more and more applications and services that require moving computation from the cloud to the edge. FLUIDOS at the Edge provides a way to extend cloud infrastructure by creating a continuum to the edge, managing, and deploying applications seamlessly across both cloud and edge environments.

The heterogeneity of edge devices largely contributes to the complexity of the task since the Edge can be broken down into 3 sublayers known as **Meta Edge**, **Deep Edge**, and **Micro Edge**. Edge devices belonging to these sublayers can be distinguished according to the degree of distribution and decentralization of the computation. For instance, the devices at the Micro edge, referred to as IoT edge devices, include sensors (e.g., temperature, inclinometer, pressure), MEMS, wired/wireless communication, ADC, microphone, or cameras, with a low cost/low power microcontroller. Deploying containerized applications on edge devices with existing solutions like KubeEdge is a way to build a baseline continuity from the cloud to the edge. However, the vanilla version of KubeEdge has many limitations, including, the limited support of the well-known IoT protocols such as (LoRA, sigfox, Bluetooth, MATTER), with the associated limitations to manage the heterogeneity of IoT Edge devices. For this reason, FLUIDOS improves and extends several components of the vanilla KubeEdge.

Modern IoT Edge devices can share raw sensor data or the computed values with different applications. Such computed data may include simple raw sensor values (e.g., temperature, humidity, …) or the computed outcome of a fusion AI algorithm. It is important to point out that, compared to the current solutions, the developed FLUIDOS at the Edge will enable the multicasting operations for sharing the pre-processed, computed data or raw data between several applications with a single data transmission from the Far Edge to the cloud. This would result in a significant reduction in power consumption, requiring fewer IoT edge devices and reducing the amount of data transmitted towards cloud applications.

Managing data at the edge is a challenging task due to the large number of connected devices that capture and send data for analysis and decision-making towards the cloud. Although projects like K3s offer complete Kubernetes instances for edge devices, several factors contribute to complex data management. These factors include the increasing variety of IoT devices, the coexistence of different wireless technologies and protocols, as well as the presence of specific operating systems and services. Therefore, the FLUIDOS architecture for the Edge introduces an innovative solution for implementing the continuum at the edge to deal with the above challenges.

2.3.1 Relevant Documentation

Funded by Horizon Europe Framework Programme of the European Union

Additional details on the FLUIDOS architecture at the edge can be found in:

- **Deliverable 2.1**, available on the website here.
- Installation guide on GitHub here.
- Prototype implementation on GitHub here.
- Sample of development kit and reference design of an IoT Edge device based on STM32 available on request.

2.4 THE FLUIDOS NODE

The objectives of the development of the FLUIDOS node can be summarized in the following:

- Define what is needed to "fluidify" the underlying infrastructure.
- Define abstractions and models to describe resources and services exported by each domain in order to enable a consumer to connect and consume the above resources/services.
- Define two sets of APIs: southbound to ensure metrics collection and northbound to facilitate the exposure of those metrics, objects, and service abstractions to FLUIDOS "supernodes" (Brokers).
- Leverage a broker node to increase scalability and support aggregation of simple nodes.

Starting from the outcomes of the development of previous section, the this work stream should formulate the design of FLUIDOS Core Services. This includes defining the primary functional elements, establishing their interrelationships, and outlining the overarching operational logic. The goal is to facilitate modularity and enable the "fluidification" of resources within the FLUIDOS system. Subsequently, the task will focus on crafting interfaces and associated software components necessary for aggregating and coordinating FLUIDOS node resources. Southbound APIs will be developed to gather metrics from the underlying layer (e.g., CPU, storage, physical objects, software services) and issue appropriate commands to the hosting platform (e.g., full virtualization in the host OS). On the other hand, Northbound APIs will aggregate these metrics in a hierarchical model, making them accessible for consumption by upper nodes.

Each FLUIDOS domain is mandated to present its own resources and services. External entities, including end customers and foreign domains, have the option to consume these resources, potentially involving some form of economic exchange. The resources and services are designed to be versatile, encompassing various objects such as network connectivity, data center resources (e.g., CPU, storage), physical objects (e.g., IoT devices), and software services (e.g., data stores, databases, analytics, etc.). As a consequence, we will define the abstractions and models necessary to describe the resources and services exported by each domain, facilitating foreign customers in connecting and utilizing the aforementioned resources and services. The chosen programmability models will

Funded by Horizon Europe Framework Programme of the European Union

provide a unified abstraction of different programmability paradigms, potentially through an ontology that is easily extensible to accommodate future requirements or custom objects/properties.

The provided implementation would then enable both mixed broker and peer-to-peer architectures. A prototype broker (or supernode) will be developed to enhance scalability and simplify deployment, especially in the case of constrained devices, while potentially reducing overhead and congestion. In a brokerized scenario, FLUIDOS nodes delegate resource allocation and policies to the broker, enabling aggregation and control on a broader scale. This approach empowers a locally decentralized environment, potentially extending to cross-domain scenarios, where business decisions can be truly intent-based.

2.4.1 Relevant Documentation

Additional details on FLUIDOS can be found in:
- Documentation on GitHub here.
- Implementation of the node functionalities on GitHub here.
- FLUIDOS Ontology on GitHub here.
- REAR protocol for resource Advertisement/Reservation on GitHub here.

2.5 THE FLUIDOS CONTINUUM

The objectives of this work stream can be summarised in the following:

- Define the decentralised interactions between FLUIDOS instances whether in the “horizontal” domain, i.e., direct peering among FLUIDOS instances, or in the “vertical” domain enabled by hierarchical stacking of multiple FLUIDOS instances.
- Deploy hyper-distributed applications and services within a virtual continuum (computing, storage, network, and services) spanning across devices, edge/cloud resources, and FLUIDOS instances.
- Define and develop APIs/protocols that handle the interaction between user/service developers and FLUIDOS as well as between the FLUIDOS instances themselves.
- Design intent-based and policy-based service meta-orchestration and provisioning.
- Enable AI-assisted orchestration and proactive provisioning of edge resources, enabled by AI for prediction, traffic forecasting, and, possibly, mobility prediction (e.g., for drones and AVs) to optimally allocate and place resources.
- Augment AI-based methods with privacy and security capabilities.

Funded by Horizon Europe Framework Programme of the European Union

In detail, this requires first defining the essential components responsible for managing the FLUIDOS continuum, along with detailing the compositions and interactions of these elements across multiple nodes (e.g., network fabric, storage, and service continuum). Additionally, the task will craft protocols to facilitate functionalities like node discovery, resource aggregation, and task offloading. As part of its scope, techniques for confidentiality-preserving service monitoring and service assurance will be designed and developed. The interfaces and interactions with the security and privacy components outlined in below will be identified and defined. Lastly, exploration and integration with North/South-bound scalability and the structure of supernodes (broker) will be undertaken.

Furthermore, the outcome of this work stream will empower intent-based service orchestration by translating desired intents into corresponding policies. This translation encompasses converting specific performance goals, expressed in terms of predefined Key Performance Indicators (KPIs) related to latency, throughput, energy consumption (in collaboration with energy and carbon aware workstream), and security (in collaboration with trust, privacy, and security workstream), into policies. The subsequent optimization will be conducted through deterministic or fuzzy approaches based on machine learning. Policy reasoning for consistency checks will be established, with policy-based orchestration relying on optimization models that define mathematical programming problems and algorithmic solutions. These solutions will consider a set of constraints derived from specified policies and requirements inferred from desired intents. Machine learning-based approaches and approximation algorithms can be leveraged in these solutions.

In addition, a deep focus will be devoted to AI-based algorithms for resource allocation, task scheduling, or anomaly detection, taking into account privacy constraints in the orchestration process. To achieve this, existing AI-based approaches used for orchestration will be modified to suit the FLUIDOS framework, incorporating privacy constraints into algorithm design and training. For privacy preservation, techniques such as federated learning, where FLUIDOS nodes collaboratively train AI models without exchanging or collecting data, or differential privacy techniques, involving the addition of noise to data samples, will be employed.

2.5.1 Relevant Documentation

Additional details on FLUIDOS can be found in:
- Documentation on GitHub here.
- FLUIDOS Ontology on GitHub here.
- Implementation of the node functionalities on GitHub here.

2.6 TRUST, SECURITY AND PRIVACY

The objectives of this workstream can be summarized in the following:

Funded by Horizon Europe Framework Programme of the European Union

- Design and implement zero-trust authenticated access to geographically distributed resources.
- Develop an isolated and trusted environment for the execution of workloads over horizontally distributed FLUIDOS instances.
- Provide monitoring techniques and anomaly detection and mitigation mechanisms to protect FLUIDOS nodes.
- Design algorithms and procedures providing QoS-aware orchestration of security functions.

Specifically, this workstream explores the establishment and monitoring of trust between nodes. It aims to deliver a secure and scalable solution for resource access control in a highly dynamic and distributed Edge/Cloud scenario, enforcing domain-specific access control policies in line with the zero-trust paradigm. It defines the components within the FLUIDOS architecture responsible for enforcing access control policies, privacy management, and data sharing. These components, part of the FLUIDOS Privacy and Security Manager, will include a Distributed Authorization Engine, functioning as a lightweight Policy Enforcement Point (PEP) to enable authorization within the Attribute-based Access Control model (ABAC).

One of the main objectives is to ensure the secure execution of workloads, requests, and response messages across the FLUIDOS ecosystem by maintaining confidentiality and integrity. Confidentiality aspects include securing the execution of workloads, handling input requests, managing work order responses, and overseeing secret key management. Integrity considerations involve safeguarding the worker registry, work order queue, and network communication within the FLUIDOS ecosystem. These security properties are being achieved by leveraging Trusted Execution Environments (TEEs), with consideration for different implementations such as AMD-SEV or ARM Trustzone. Additionally, we explore the authenticity of code and hardware through remote attestation mechanisms to trust the execution of applications within secure enclaves. Attestation of FLUIDOS container execution covers secure provisioning of secrets, file system encryption, and authentication. Lastly, the secure communication between horizontally distributed FLUIDOS instances running in different environments will be evaluated.

Another aspect of this work package involves identifying and mitigating security and privacy threats targeting the FLUIDOS container-based infrastructure. Given the limited isolation containers provide, we analyse existing mechanisms like SELinux, Apparmor, and Seccomp to enhance the security of the host OS kernel. Novel techniques are being proposed to define and enforce dynamic, fine-grained security policies to ensure secure execution of containerized applications, adhering to the principle of least privileges and the zero-trust paradigm. Additionally, the task is exploring privacy leakages within the FLUIDOS container environment and propose techniques to quantify and minimise such leakages, particularly relevant in the context of confidential computing.

Funded by Horizon Europe Framework Programme of the European Union

At the same time, a malicious user could attempt to disrupt the proper functioning of a FLUIDOS node or target other services within the infrastructure. In this context, this workstream has been working on AI-based threat and anomaly detection solutions. Specific emphasis has been given to balancing the trade-off between detection accuracy and monitoring depth, with the goal of achieving precise detection while maintaining a high level of effectiveness in mitigation. The usage of federated learning to train a model for threat detection across different administrative FLUIDOS domains is also being proposed.

Recognising that methods for threat and intrusion detection can be evaded (through adversarial machine learning, for instance), generate a substantial number of alarms, and are vulnerable to zero-day attacks, we are putting forth a proposal to augment them with a cloud-native approach to cyber-deception. This approach relies on orchestration capabilities to offer a resource-aware strategy for creating and deploying decoys.

Finally, this workstream delves into the design of orchestration algorithms for activating security services on applications running on FLUIDOS. The goal is to ensure adequate protection against cybersecurity threats while considering both application-level and infrastructure-level parameters.

2.6.1 Relevant Documentation

Additional details on FLUIDOS can be found in:
- Documentation on GitHub here.
- Relevant repositories on GitHub here.

2.7 ENERGY AND CARBON AWARE

The objectives of this workstream can be summarized in the following:

- Define an energy- and carbon-aware computation model that can shift loads both in time and geography.
- Devise cost-effective infrastructure optimisations for industrial environments.
- Use Artificial Intelligence and Machine Learning methods for performance prediction and enhancement.

This workstream will create an energy- and carbon-aware computing model within the FLUIDOS framework. In addition to presenting its own resources, each FLUIDOS domain will communicate the current carbon intensity of its electricity (measured in grams of CO2 per kilowatt-hour), which varies across locations due to specific electricity mixes and fluctuates hourly at each location. This model goes beyond the inherent energy-saving features of FLUIDOS, providing the capability to optimize energy consumption and carbon emissions in edge computing. FLUIDOS will utilize this model to perform load shifting both in time and

Funded by Horizon Europe Framework Programme of the European Union

geography, capitalizing on nodes close to energy sources or powered by a significant share of renewable energy. The task will design the energy- and carbon-aware model, outlining its main functional elements and additional technical specifications, including functionalities and interfaces.

In tandem with the focus on energy and carbon efficiency, this task aims to reduce infrastructure costs. These savings will arise from the inherent resource and energy efficiency of FLUIDOS and additional efficiency measures from the previous task. For example, some nodes may employ less powerful CPUs and offload tasks to nearby nodes, resulting in overall energy and cost reduction. While there is often synergy between OPEX cost savings and energy/carbon savings, depending on electricity prices, conflicts may arise. To address these potentially conflicting goals, the energy- and carbon-aware computing model of FLUIDOS will enable setting fine-granular priorities and reflecting desired trade-offs between the two when relevant.

Furthermore, the work package will delve into the training of artificial intelligence and machine learning models for two purposes. Firstly, to learn from past loads and predict the energy demand of future edge computing tasks. Secondly, to compare and learn from strategies used by other nodes for achieving better individual and overall energy efficiency. The outcomes of the energy demand prediction task will contribute to the model from the previously described tasks, which communicate the energy, carbon, and costs of edge computing tasks with the aim of overall minimization.

2.7.1 Relevant Documentation

Additional details on FLUIDOS can be found in:

- Documentation on GitHub here.
- Relevant repositories on GitHub here and here.

Funded by Horizon Europe Framework Programme of the European Union

3 EARLY ADOPTERS

What follows here is a general description of the early adopter of the FLUIDOS project.

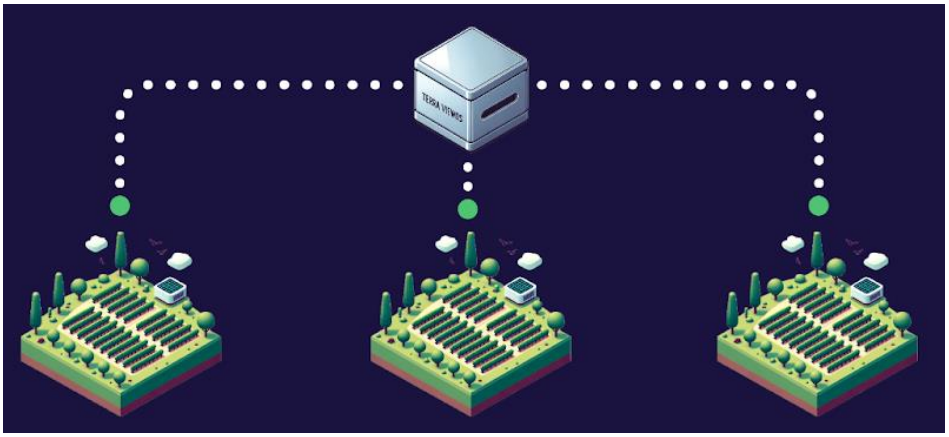
3.1 TERRAVIEW

3.1.1 Use Case Description

TerraviewOS, is a unified platform for viticulture, enables the grower to manage information from many sources, returning high-value info such as yield estimation, smart irrigation, and disease prediction and diagnosis. Since TerraviewOS is cloud-based, a key problem is the interaction with on-field devices in the presence of poor network connectivity. Operations such as drone aerial surveys (approximately 30-40GB of data for a modest area of 10 hectares) may return their valuable results with large delays, resulting in potentially poor user experience or indeed non-operation for customers.

3.1.2 Use Case Problem

In the following diagram, the use case's scenario is described. The scenario shows a conceptual representation of a distributed computing architecture, "edge" architecture. In this scenario there are 3 nodes, each providing the TerraviewOS service to an individual and different customer. These nodes are computing units that a customer can either purchase or lease from Terraview. These nodes are connected over different networking technologies depending on what is available at the customer site. They are connected to the central TerraviewOS service. At the local node specific functionality of TerraviewOS is executed and other functionality is provided by the core TerraviewOS service shown as the top box in the diagram.



Each FLUIDOS node is represented by one customer, a customer having one or more vineyards to manage. Each customer is isolated yet connected to shared computational resources that are provided by Terraview. The goal of this scenario

Funded by Horizon Europe Framework Programme of the European Union

will be to deliver an edge architecture that integrates the central service delivered by the technology of FLUIDOS and solves the following problems.

- **No simple and integrated network fabric over cloud continuum**.

  FLUIDOS will provide an integrated Network Fabric over Cloud Continuum. The current landscape lacks a unified network fabric across the cloud continuum, leading to connectivity and interoperability challenges. The smart viticulture use case involves using the FLUIDOS unified networking framework to streamline data transfer and resource management across various cloud services, from private to public clouds and edge computing. This shows the simplification of multi-cloud management and ensures reliable network behaviours.

- **No easy way to define and manage a cloud continuum application.**

  Today, there is no straightforward method for managing applications in cloud continuum environments, complicating resource utilisation and application lifecycle management. The smart viticulture use case will take advantage of the FLUIDOS technology for easy application definition, deployment, orchestration and management across different cloud models. It will include intuitive interfaces, standard templates, automated deployment tools, and robust monitoring for optimal application performance in distributed cloud environments.

- **Deficit of security, privacy or trust.**

  Significant concerns around security, privacy, and trust hamper cloud computing's effectiveness. The mart viticulture use case will focus on integrating advanced security frameworks (TEE/TPM, anomaly detection etc) for robust encryption, secure data handling, and strict access controls. It will also incorporate privacy-preserving technologies and comply with international data protection laws to enhance user trust and ensure data integrity in the cloud.

3.1.3 Current Approach

The current approach faces many limitations:

- **Everything is centralised.**

  The current centralised system architecture concentrates all services and processing in one location, leading to potential system failures and latency issues. A revised technical use case would explore decentralised architectures to distribute processing across multiple nodes. This would improve system resilience, reduce latency, and offer a scalable solution for geographically dispersed users.

- **All data resides with the service provider.**

Funded by Horizon Europe Framework Programme of the European Union

With all data stored by the service provider, concerns arise about privacy and security. The technical use case should develop a distributed data storage solution, possibly using blockchain or decentralised databases, to enhance data security and give users more control. Ensuring data encryption, access control, and compliance with data protection laws are key aspects of this approach.

- **No reliability on the customer side - network dead, no service.**

  Dependence on continuous network connectivity leads to service disruptions during network outages. A technical use case should focus on offline capabilities or local caching to maintain basic functionalities during disconnections. Systems designed to operate offline and sync with central servers once online would ensure uninterrupted service.

- **Huge data ingress over limited network connection.**

  Significant data ingress through limited network connections causes congestion and inefficiency. The technical use case would look into optimising data transfer using data compression, selective transmission, or edge computing to process data closer to its source. This approach aims to minimise data transfer volume, ease network congestion, and enhance system performance.

3.1.4 The FLUIDOS Approach

The architecture of TerraviewOS is designed with a focus on efficient and secure operation across multiple customer on-premise sites leveraging the support services operated upon centralised cloud infrastructure. At the core of this architecture lies the concept of the FLUIDOS domain, which encompasses both customer sites and centralised services.

In this use case, each customer site will host a FLUIDOS node, responsible for running the majority of TerraviewOS services along with the user interface. These nodes are designed to ensure security, with data encrypted both at rest and in transit, and will provide access to necessary centralised services. The backend services of TerraviewOS will run in the cloud, with a crucial design choice being the avoidance of one backend instance per FLUIDOS node. This approach aims to facilitate the secure sharing of workloads and data across FLUIDOS nodes that reside in the cloud.

The FLUIDOS nodes will be designed to ensure that different nodes cannot interfere with each other, maintaining a high level of operational integrity and security. This is further enhanced by the requirement for workloads to run on Trusted Execution Environments (TEEs), which can be specified in the application descriptors through intents.

Centralised cloud services of TerraviewOS, operating on a FLUIDOS node, will be multi-tenant, ensuring a clear separation between different customers. This separation is pivotal in maintaining data integrity and privacy for each customer.

Funded by Horizon Europe Framework Programme of the European Union

The key objective of this architecture is to minimise any duplication of service functionality, as such redundancy could lead to inefficiencies, technical debt, and unnecessary costs.

All deployments for customer sites will be managed centrally, streamlining the operational process and ensuring consistency across the FLUIDOS domain. This centralised management approach allows for better control and oversight of the services provided, ensuring that each customer site operates effectively and securely within the overarching TerraviewOS ecosystem.

Therefore, the expected advantages can be summarized in the following:

- **Business continuity: adapt to network changes, operate without uplink, and replicate for failover.**

  This technical use case focuses on maintaining business continuity through network adaptability, offline operation, and failover replication. The system should adjust to network fluctuations, ensuring functionality without a primary network connection, and use alternative communication methods when needed. Critical operations and data must be replicated across different nodes to enable seamless service continuation in the event of a node failure, thereby reducing downtime and data loss.

- **Security: raw data stays local, isolation and trusted execution on network.**

  Prioritising security, this use case emphasises keeping raw data local and isolated within the network for enhanced protection. The system is designed to store and process data locally, minimising external data exposure. Implementing network isolation techniques like VPNs and secure environments for data processing, the system ensures data integrity and confidentiality, protecting against unauthorised access and breaches.

- **Decentralisation: local access, no central cloud, reduced network traffic, workloads at edge and core.**

  The focus here is on decentralising system architecture to facilitate local access and minimise network traffic. By distributing workloads between edge and core, the system reduces data transmission distances, lowers latency, and optimises network usage. Local processing at the edge enhances real-time decision-making and lightens the load on central servers. The design aims for efficient resource management across distributed nodes, improving performance and scalability.

3.2 RSE

3.2.1 Use Case Description

Funded by Horizon Europe Framework Programme of the European Union

The introduction of a massive number of renewable energy sources such as solar panels require a highly monitored distribution grid that can coordinate energy producers and consumers in real-time. This relies on a large number of automatic measuring devices such as PMUs (Phasor Measurement Units), with real-time data collection and synchronization carried out by PDCs (Phasor Data Concentrators). These devices, already in use on the high voltage grid, provide the data used to compute the grid state estimate for monitoring and control applications.

3.2.2 Use Case Problem

The introduction of PMUs into the distribution network brings about various challenges, including:

- **Scalability**

  The number of PMUs required for the power grid to be observable is significantly higher for the distribution case. While hundreds of PMUs are needed for the Italian transmission network to be observable, the distribution network requires a number of the order of thousands. This implies the need to manage a massive amount of distributed devices and their corresponding data flows.

- **Resiliency**

  With thousands of devices in play, enhancing the resilience of the ICT infrastructure for data collection is crucial. For instance, managing faults or planned maintenance should be automated, as well as the possibility for the above devices when disconnected from the Internet and/or unable to access companion services running in the cloud.

- **Latency**

  This particular application has strict data latency requirements (of the order of milliseconds) to enable real-time control of the power grid.

- **Cybersecurity**

  The power grid is a critical infrastructure for the country and, as such, must be protected from any malicious attack, including the ones targeting monitoring and control services.

3.2.3 Current Approach

While the Italian transmission grid is centrally managed by a single PDC, the introduction of PMUs in the distribution grid is still at an experimental level. Nevertheless, the massive amount of PMUs and PDCs in the future network makes this centralized approach (a) hardly scalable, (b) not resilient to network failures or forced disconnections (e.g., to preserve a portion of the network from an ongoing attack).

3.2.4 The FLUIDOS Approach

Funded by Horizon Europe Framework Programme of the European Union

Traditionally, PDCs were monolithic applications running on dedicated hardware; however, with the increasing computational power available at lower costs, this is changing in recent years. Experimental efforts are underway to virtualize applications and utilize Kubernetes for orchestrating the deployment of PDCs and real-time analysis applications at the edge. This is aimed at reducing latency issues and improving resiliency, avoiding the need of operator physical assistance in case of outages, and paves the way for their usage within a FLUIDOS-based environment.

In fact, FLUIDOS creates a continuum of resources from the edge to the cloud and enables the displacement of workloads, such as data collection and analysis processes, based on specific scenarios (faults, reconfiguration, maintenance). The main features of the approach enabled by FLUIDOS are:

- **Computing Continuum**

  FLUIDOS would enable PDCs and analysis applications to continue functioning even if communication with control centers is interrupted by migrating PDC services to an adjacent node in case of fault.

- **Intent-Based Orchestration**

  FLUIDOS can automatically orchestrate PDCs based on the latency between the node and PMUs, thereby improving the power grid state estimate or responding to faults.

- **Cybersecurity**

  FLUIDOS ensures service isolation from other applications on the hosting node with different usage permissions. It also leverages logging and anomaly detection capabilities and provides survival capabilities in case a portion of the grid is disconnected from the main network, hence preserving its operations in case of a cyber-attack.

3.3 ROBOTNIK

3.3.1 Use Case Description

Mobile robots for Industry 4.0, smart logistics, and retail are resource-constrained and battery-powered mobile robots that operate in shared spaces with humans and other IoT devices, such as elevators, automatic doors, and other mobile robots. For autonomous operation, mobile robots are equipped with sensors and high-performance computational algorithms that (1) need considerable computing power and (2) need to be executed at the highest speed.

3.3.2 Use Case Problem

This kind of robotics, by definition, uses battery-powered devices, unlike stationary robot arms. This adds constraints to an already complicated application. In

Funded by Horizon Europe Framework Programme of the European Union

robotics logistics, idle robots are not productive. And intrinsically, robots need to charge their batteries to continue working. During this time, the robot is not available to move things around.

The robot's software is complex and requires heavy computation, so it is mandatory to equip the robot with powerful enough computing engines that, however, should limit the consumed power. Onboarding an ultra-low-resource device will lead to a robot that is not capable of processing the required tasks, and onboarding an ultra-high-resource device will deplete the battery very quickly.

There is an additional relationship between the computer load and the battery drain. Heavy computational tasks drain the battery faster.

3.3.3 Current Approach

To solve the above computing problem with the current technology, we can follow either one of the following two approaches: (a) centralized processing, (b) individual computing.

- **Centralized processing**

  The robot fleet operates under a centralized communication architecture, whereby the intelligence (i.e., processing resources and software) is placed in a central component, such as the fleet manager (or orchestration system). All communication between the individual robots and the fleet manager is routed through an intermediary edge or cloud device. In this setup, the central device acts as a hub that (a) implements processing algorithms, and (b) facilitates and controls the communication flow within the network.

  This centralized approach may introduce a single point of failure and potential latency issues as all communications must pass through the central hub, and it is not appropriate when robots may experience connectivity issues (e.g., poorly connected area in a factory).

- **Individual computation**

  In this approach, robots typically compute everything on the device itself. This practice, however, may lead at least to three main problems. First, computational inefficiency, as it may result in scenarios where some robots remain idle and underused, while others are overwhelmed with computational tasks, nearing the point of overloading their onboard computers. Second, over-dimension of the available computing resources (or, alternatively, the necessity to deploy simplest computing algorithms), given that the necessary hardware resources to comply with peak computing demands must be available on the robot itself; alternatively, to save on the cost of the above hardware, simplest (and less resource-hungry algorithms, but also less efficient) should be used. Third, excessive battery usage, as all the processing happens on the robot itself, substantially draining the battery power, which is particularly important on small robots

Funded by Horizon Europe Framework Programme of the European Union

(with a small-size battery, while computing power for robot navigation and other algorithms is an invariant with respect to the size of the robot.

3.3.4 The FLUIDOS Approach

The FLUIDOS continuum has the capability to transparently use computing resources nearby, increasing the overall system's productivity by intelligently and dynamically externalizing the robotics workload to other devices (e.g., a server at the factory premises) and/or using the robot's idle time (i.e., when the robot is docked in the battery charging station) to increase the entire system's computational capabilities. Each robot can leverage this approach when it is well-connected to the network, while it can rely on its sole onboard computing capabilities (which are turned on only upon necessity) when moving in a poorly connected area.

This approach will lead to a significant decrease in battery usage and an increase in the robot's computational capabilities beyond its onboard limits, with the capability to dynamically adapt its computing behavior (i.e., onboard or offloaded) based on the actual operating conditions.

With FLUIDOS, we can apply the cloud continuum computing approach by considering each robot as an edge device and intelligently and dynamically outsourcing robotics workloads to other robots or devices depending on the environment. This can be achieved without interfering with the robotics task, so robot developers will be able to run their applications without changing their way of working.

Instead of using monolithic bare-metal workloads, the robots will use cloud-native technologies like containerization and Kubernetes to split and dynamically place the workloads in different devices. All robots will be treated as edge devices that can accept or externalize workloads, instead of being isolated devices.

Given the potential capability of the FLUIDOS intent-based orchestrator to pursue different objectives, workload distribution among the different available systems can be optimized to achieve diverse goals such as:

- Maximize the battery life of the robots.
- Minimize the time it takes for the robots to complete their tasks.
- Ensure that all of the robots are evenly utilized.
- Avoid overloading any individual robot.

Highly dynamic decisions can be envisioned as well. For example, if a robot is low on battery, FLUIDOS might move its workloads to other robots with more battery power. Or, if a robot is overloaded, FLUIDOS might move some of its workloads to other robots that are less busy.

In a nutshell, the FLUIDOS approach to robot workload orchestration enables to improve the performance, efficiency, and reliability of your entire system.

Funded by Horizon Europe Framework Programme of the European Union

4 OPEN CALLS: RESEARCH CHALLENGES AND USE CASES

The upcoming FLUIDOS Open Calls, scheduled for launch in December 2023, are aimed at European high-tech SMEs, researchers, and innovators. FLUIDOS Open Calls present a unique opportunity for participants to explore the project´s novel mechanisms for data sharing and processing in the computing continuum.

The FLUIDOS Open Calls aim to achieve the following objectives:

- Facilitate Technological Advancements: Encouraging applicants to offer additional Open-Source functionalities to be integrated with FLUIDOS, fostering continuous innovation and expansion of the platform's capabilities.
- Validate FLUIDOS Architecture: Through the FLUIDOS Use Case call, applicants will be invited to test and validate FLUIDOS architecture and software by integrating their own additional software on top of the FLUIDOS system, exploring new sectors beyond environmental monitoring, mobility, healthcare, and security.
- Promote Environmental Sustainability: Encouraging participants to leverage FLUIDOS' minimised energy consumption for data processing to create business models that are more environmentally sustainable.

Funded by Horizon Europe Framework Programme of the European Union

5 PUBLIC DOCUMENTATION AND QUICK START

Add here anything that might be relevant for an Open Call partner to learn more about the project.

Funded by Horizon Europe Framework Programme of the European Union