Swarmchestrate: Towards a Fully Decentralised Framework for Orchestrating Applications in the Cloud-to-Edge Continuum

Tamas Kiss[1], Amjad Ullah[2], Gabor Terstyanszky[1], Odej Kao[3], Soren Becker[3], Yiannis Verginadis[4,5], Antonis Michalas[6], Vlado Stankovski[7], Attila Kertesz[8], Elisa Ricci[9], Jörn Altmann[10], Bernhard Egger[10], Francesco Tusa[1], Jozsef Kovacs[1,11], Robert Lovas[11, 12]

[1] University of Westminster, London, UK, email:{kisst, terstyg, f.tusa, j.kovacs}@westminster.ac.uk

[2] Edinburgh Napier University, Edinburgh, UK, email: a.ullah@napier.ac.uk

[3]Technische Universitaet Berlin, Germany, email: {soeren.becker, odej.kao}@tu-berlin.de

[4]Institute of Communications and Computer Systems, National Technical University of Athens, Greece email: jverg@aueb.gr

[5]School of Business, Department of Business Administration, Athens University of Economics and Business, Greece

[6]Tampere University, Tampere, Finland, email: antonios.michalas@tuni.fi

[7]University of Ljubljana, Ljubljana, Slovenia, email: vlado.stankovski@fri.uni-lj.si

[8]FrontEndArt Software Ltd, Szeged, Hungary, email: attila.kertesz@frontendart.com

[9]Fondazione Bruno Kessler, Trento, Italy, email: eliricci@fbk.eu

[10]Seoul National University, Seoul, South Korea, email: jorn.altmann@acm.org, bernhard@csap.snu.ac.kr

[11]Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN), Hungary, email: {jozsef.kovacs, robert.lovas}@sztaki.hu

[12]John von Neumann Faculty of Informatics, Óbuda University, Budapest, Hungary

Abstract. Collecting and analysing large amounts of data in the Cloud-to-Edge computing continuum raises novel challenges that traditional centralised orchestration solutions cannot handle efficiently. To overcome the limitations of current centralised application management approaches, this paper presents a fully decentralised application-level orchestrator, based on the notion of self-organised interdependent Swarms. Application microservices are managed in a dynamic Orchestration Space by decentralised Orchestration Agents, governed by distributed intelligence that provides matchmaking between application requirements and resources, and supports the dynamic self-organisation of Swarms. Knowledge and trust, essential for the operation of the Orchestration Space, are managed through blockchain-based trusted solutions and the utilisation of emerging methods such as Self-Sovereign Identities (SSI) and Distributed Identifiers (DID). End-to-end security of the overall system is assured by utilising state-of-the-art cryptographic and privacy-preserving data analytics algorithms. A digital twin, that runs in parallel to the physical system, further improves its behaviour with predictive feedback. The presented concept is going to be implemented in the EU-funded Swarmchestrate project that starts in 2024.

1 Introduction

Orchestration is referred to as the coordination and management processes of physical computational resources of an infrastructure environment to serve the application requirements, as defined by Jiang et al. [1]. The definition of infrastructure environment here is contextual. For example, Tomarchio et al. [2] discussed cloud orchestration and, therefore, referred to it as coordination and management processes of cloud resources, whereas Costa et al. [3] discussed this within the context of the fog paradigm. Irrespective of the underlying infrastructure environment, the overall goal of the orchestration system in terms of users' business requests is to ensure the meeting of Quality of Service (QoS) goals of applications. Therefore, in the absence of a universally agreed definition, a Cloud-to-Edge orchestration system is responsible for providing simultaneous access to the heterogeneous resource landscape of the continuum for the automation of application deployment and management over the resource landscape. It guarantees QoS goals, by handling the required complex tasks of resource selection, allocation, deployment and monitoring, and the run-time reconfiguration control of the resources and applications.

Although there are several research efforts and even relatively mature solutions providing orchestration capabilities, none of these can fulfil completely the highly dynamic and complex requirements imposed by the Cloud-to-Edge continuum.

The efficient and effective management and processing of the large amounts of data generated at the edges of the network must deal with versatile requirements. Some data need to be processed locally due to regulations, privacy issues and/or performance constraints, while some others may require access to long-term sophisticated computational cloud resources. Applications running in such systems have a wide range of requirements, including the execution of low-latency analytics closer to the data source, privacy sensitivity, context awareness, time and location awareness, as well as the need for simultaneous access to geographically distributed arrays of sensors, remote localised heterogeneous computational resources, and large-scale on-the-fly allocated multi-cloud infrastructure. Therefore, a new generation of orchestration tools and solutions is needed to handle this complexity efficiently and to take into consideration this dynamically changing set of complex requirements in an intelligent way.

All currently available orchestration tools, responsible for deploying and managing data processing applications in the Cloud-to-Edge computing continuum, are based on a certain level of centralisation [4]. Such centralisation, while relatively easy to implement, carries several disadvantages. The central component can become a single point of failure, can be easily overloaded as the system scales, and provides a good target for security attacks [5]. Additionally, such a centralised approach does not fit well with the highly distributed and dynamically changing nature of the computing environment. A centralised management approach cannot react fast enough to some changes in local environments (e.g. volatility of resources) and cannot support fast adaptation of resources and application requirements (e.g. due to the movement of certain computing elements).

An alternative is a decentralised self-adaptive system that can be aware of its surroundings and can organise and reorganise itself without any central control and management. While the implementation of such a system is complex, recent advances in various fields of computing, including Swarm computing, distributed AI, distributed ledger systems and decentralised identity management, now enable the efficient realisation of such an approach.

To address these challenges Swarmchestrate, a new EU-funded research project kicking off in 2024, aims to combine and extend the above-mentioned emerging technologies and create a completely decentralised autonomous and self-organised application management system. The approach applied by Swarmchestrate is fundamentally new to application orchestration and suitable to manage hyper-distributed applications that span across large distances and the different layers of the dynamic compute continuum.

The rest of this paper overviews the current efforts towards decentralised orchestration, introduces the high-level concepts and fundamental building blocks of the Swarmchestrate framework, and outlines its main components that need to be implemented.

2 From Centralised to Decentralised Orchestration

Existing Cloud-to-Edge orchestration solutions can be divided into three categories: static [6], rule-based [7] and machine learning (ML) based [8] intelligent approaches. Static approaches place the burden on system engineers for structuring the system, as they statically map the different parts of an application to different resources of the compute continuum. Rule-based approaches embed some predefined threshold-based rules that help in determining the selection of resources. Lastly, ML-based approaches make informed decisions based on the collected data at runtime. For a more detailed list of different orchestration solutions from these categories, refer to Ullah et al. [9]. Most of these existing solutions, irrespective of their individual characteristic and underlying implementation techniques, follow a centralised model where a central entity, usually running in the cloud, collects data for further decision-making from the entire compute continuum.

In contrast to the centralised model, there are only a few solutions that follow the decentralised approach. Some examples include HYDRA [10] and Caravela [11] that provide a peer-to-peer (P2P) network of nodes where each node is both orchestrator and resource; mF2C [12] that follows a hierarchical architecture, where different agents work at different layers of the ecosystem and facilitate proactive decision making; Ozyar et. al. [13] that utilises Blockchain to ensure security, however, its scope is limited to container placement at the edge layer; and finally, EPOS Fog [14], a multi-agent system where each node has its own software agent that defines which service is deployed on which host in the neighbourhood of the agent.

The Swarmchestrate project has the ambition to evolve the concept of decentralised Cloud-to-Edge orchestration in dimensions: (1) Elaborate the concept of decentralised orchestration from an application centric approach, in contrast to the currently available resource-oriented orchestration solutions that target and optimise resource provider objectives. (2) Develop novel standards and protocols for the collaboration of the decentralised orchestrators based on Swarm computing principles and the collaboration of multiple Swarms for the fulfilment of the overall objectives of the target applications. (3) Develop novel deployment and reconfiguration strategies with the aim of optimising application centric objectives based on certain requirements, e.g. application topology, fault tolerance, performance goals and various contextual constraints, such as resources, energy utilisation and geographical location.

The developed algorithms and solutions will be utilised to transform a centralised open-source Cloud-to-Edge orchestrator, called MiCADO-Edge [15], into a fully decentralised solution based on Swarm computing principles. The new decentralised orchestrator will then be applied in several use cases to improve wastewater manhole management, create a metaverse digital twin of natural habitat, provide more efficient parking space management, and improve video scene analytics in cities.

3 The Proposed Swarmchestrate Architecture

The Swarmchestrate project aims to develop a novel and innovative decentralised application-level orchestration solution with the potential to change the fundamentals of how applications are managed and executed in the highly dynamic Cloud-to-Edge compute continuum. Within the decentralised orchestration spectrum, Swarmchestrate's approach follows an application-centric view in contrast to the currently available resource-oriented orchestration solutions that target and optimise resource provider objectives.

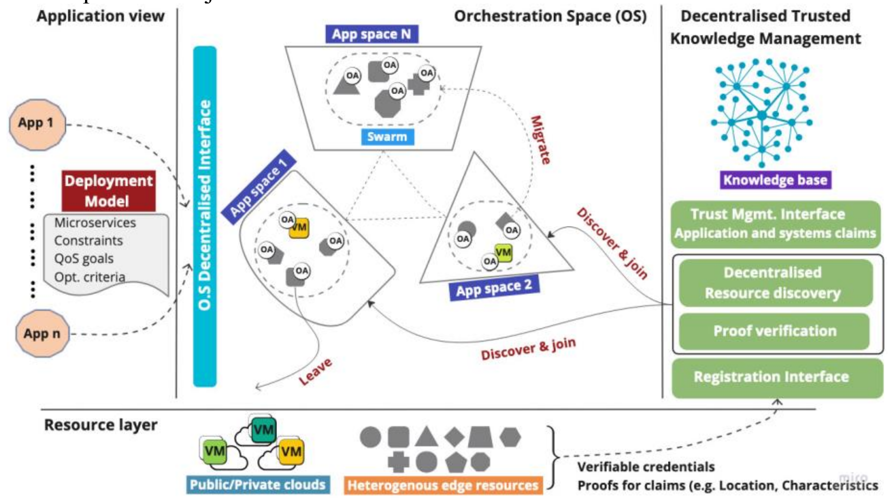

Fig. 1. High-level Architecture of Decentralised Orchestration in Swarmchestrate

The overall methodology of Swarmchestrate has been developed to address the inherent highly decentralised nature of the compute continuum, which is formed of several heterogeneous resources, spanning multiple administrative domains, that exhibit highly dynamic behaviour in terms of capacity and availability. Therefore, the development of intelligent mechanisms for the orchestration of applications deployed within this continuum becomes essential. Application owners do not have to be aware of the complexity of the underlying resource infrastructure. In fact, they can use a single and uniform high-level and interoperable descriptor that incorporates both the topology and the constraints of an application, as well as the related optimisation goals and performance requirements. Swarmchestrate abstracts the low-level details of the Cloud-to-Edge continuum and allows users to run their applications' microservices in a multi-domain infrastructure while taking advantage of its built-in trust and guaranteed security features.

Based on these essential aspects, Fig. 1 presents a high-level architecture, depicting the overall vision of Swarmchestrate and how it works. From a structural viewpoint, the overall system is divided into four distinctive parts, comprising of (1) Application view – dealing with application specification, (2) Orchestration space – handling all core functions of orchestration, (3) Trusted knowledge management – secure handling and management of all system-wide knowledge and interface required to enable trust and transparency, and (4) Resource layer – representing resources of the continuum that span across multiple cloud environments and non-cloud layers. The technical implementation of these parts is being realised using the following fundamental concepts described below.

3.1 Application View

Within the Application view, DevOps can describe their applications, including resource needs, governing policies and QoS requirements. There are several such specification formats, especially for cloud computing environments, and these are essential to ensure portability and avoid vendor lock-in. However, the emergence of the Cloud-to-Edge model raises the quintessential need for the extension of such cloud description languages. There are clear shortcomings when capturing deployment, monitoring, contextualisation, and reconfiguration aspects of Cloud-to-Edge applications. Current approaches cannot sufficiently cope with the concept of decentralisation required in application management and orchestration, especially when distributed Cloud-to-Edge continuum resources are involved.

To improve this, Swarmchestrate introduces modelling artefacts that can extend a standards-based specification, such as TOSCA [16], to cover all the necessary specification details, allowing the decentralised and automated DevOps platform to manage and orchestrate microservices-based applications in the heterogeneous and dynamic Cloud-to-Edge continuum. The project concentrates on enhancing the specification for the entire application lifecycle, allowing application and resource specification across the Cloud-to-Edge continuum, which is still missing at large.

In Swarmchestrate, an application can be of different types (e.g., batch-based, or long-lasting services) and can consist of multiple container-based microservices, whose blueprints are possibly hosted in different repositories. As represented in Fig. 1, by using the above-mentioned enhanced modelling features, application owners can describe a deployment model of their applications consisting of the application topology and the high-level description of its contextual requirements in terms of resources (e.g., security, geographical constraints), the application optimisation criteria and the related QoS goals in terms of performance (e.g., latency, trust). Such an approach takes care of automating the instantiation of the required microservices on the underlying Cloud-to-Edge infrastructure according to the given deployment model, without the need of further involvement of the application owner.

3.2 Orchestration Space - Decentralisation, Swarms and Intelligence

In Swarmchestrate, the applications are submitted to the Orchestration Space, which is a distributed entity with no central access point. The notion of Orchestration Space is the confluence of three concepts – decentralisation, Swarms and intelligence – for achieving efficient, optimised and trusted orchestration of applications in the Cloud-to-Edge eco-system.

As it was analysed in Section 2, the majority of existing orchestration solutions are typically based on centralised architectures. Whilst such centralised solutions have a number of benefits, they do not fit well with the distributed nature of the Cloud-to-Edge continuum, which requires a more decentralised orchestration approach. Resources closer to the edge of the network are typically volatile, their network connection may be lost from time to time, and their processing and data storage capabilities are limited. Therefore, more emphasis must be put onto local decision-making and to the collaboration of multiple interacting entities every time global decisions related to the behaviour of applications need to be made. To address these issues, Swarmchestrate follows a decentralised approach towards orchestration where multiple players are responsible for executing the core functions of orchestration in the Cloud-to-Edge ecosystem in place of a centralised entity. This notion of decentralisation will be realised through the implementation of Swarms.

The key characteristic of Swarm computing [17] is the emergence of the collective behaviour and intelligence of individual agents as a result of interactions between them, rather than being explicitly controlled by a central entity. The usage of distributed agents enables a self-organised, highly scalable, and adaptable orchestration approach, which fits perfectly with the highly dynamic and distributed nature of Cloud-to-Edge systems. The common goal that the distributed agents are aiming to achieve in this scenario is the execution of orchestration functions for the applications submitted to the Orchestration Space.

Swarms are commonly associated with the concept of close proximity [18] meaning that agents geographically located near to each other can come together to form a Swarm dynamically, in order to cooperate to the completion of a task. However, in the Cloud-to-Edge compute continuum, an application – often consisting of multiple interconnected microservices – requires simultaneous access to resources distributed across the different layers of the compute continuum. Based on this aspect, Swarmchestrate extends the concept of Swarm-formation from close proximity to logical proximity [19]. This is determined based on the application's requirements and characteristics, i.e., resource requirements (such as CPU, memory, storage), security requirements, locality, performance, availability, energy constraints, trust factors, etc., instead of considering the resources' geographical distribution only. In this regard, we formalise the semantics for logical proximity based on application requirements and characteristics of involved resources and further define the mechanism of self-organisation based on logical proximity. We are also defining protocols for intra- and inter-swarm coordination.

In Orchestration Space, Swarms can be formed based on the above-mentioned concept of logical proximity. They are self-organised and fully dynamic, as resources can join or leave a Swarm based on their changing requirements/availability (or preferences) and/or dynamically changing application characteristics. Furthermore, based on the notion of logical proximity, we also envision that a single resource may be part of more than one Swarm. Each Swarm aims to fulfil the requirements of a particular application within the Orchestration Space. Hence, a Swarm provides this notion of Application Space (see Fig. 1) that is potentially changing at any point in time, based on application requirements and resource behaviours. Although Swarms can change, the overall lifetime of the Application Space is directly correlated with the application's lifetime. After completing/terminating the application, the Swarm dissolves as a result of collective self-organising decisions.

Swarms within Orchestration space are aware of each other, hence, they can also influence each other's behaviour. For example, a resource that is part of multiple Swarms can become overloaded as a result of the load in one particular Swarm, ultimately affecting the performance of other Swarms too. Therefore, strategies for inter-Swarm coordination are further formalised to achieve the overall objectives at global level.

From a technical viewpoint, Orchestration Agents (OA) in Swarms are responsible for picking up submitted applications and their microservices, and carrying out the tasks. OAs are attached to microservices and responsible for the self-organisation of the Swarms and for the inter-Swarm communication, as described above. In the absence of a central entity, these agents interact with each other based on certain simple principles and are able to share information within and across Swarms. Through the holistic intelligence based on the interaction and cooperation amongst OAs, the application-level objectives can be achieved.

Swarmchestrate exploits the use of distributed AI techniques to establish Swarm intelligence systems with an aim to optimise the overall dynamics of Swarms. This includes aspects such as the dynamic formation of Swarms, the individualistic behaviour of OAs, principles for intra- and inter-Swarm coordination, interaction with the environment, information sharing, and adaptation in case of changes in the operating conditions.

3.3 Trusted Knowledge Management

Trust in a decentralised environment is difficult to achieve. Swarmchestrate intends to generate various verifiable credentials/presentations and proofs that can be used in the context of obtaining and providing trust (e.g. proof of presence, proof of location, proof of computing capabilities). The possibility to generate various Zero-Knowledge proofs is considered across all levels of the applications. Swarmchestrate deals with trust as the most essential and fundamental pillar of the platform.

For this purpose, and to support the overall philosophy of decentralisation, Swarmchestrate develops a Blockchain-based decentralised knowledge and trust engine/infrastructure, which is shown on the right-hand side of Fig. 1. The role of this infrastructure is twofold: it is responsible for the global handling and management of knowledge, as well as for facilitating overall transparency and assuring trust amongst the system, the distributed resource layer and the users' applications running on the Swarmchestrate platform. The functionalities of this persistent trusted knowledge management infrastructure are available and used by both the Resource and the Orchestration Space layers.

Relevant sets of trust attributes, essential for the transparent and trustworthy interactions amongst the entities of the system within the context of orchestration in the Cloud-to-Edge compute continuum, are the subject of an initial investigation carried out by the project. This aims at the development of formal models of trustworthiness that help us in guaranteeing the dependable and trusted interaction between entities, stakeholders and services in a decentralised environment. These formal models provide foundation to the development of evidence- and blockchain-based trust management solutions using methods of SSI (Self-Sovereign Identity) [20] and DID (Decentralised Identifier) [21]. Using such a solution, identities are created and associated with various functionalities when resources join Swarmchestrate, so that these can only operate under circumstances where proper rights are given to them. Hence, it will be demonstrated that full transparency, traceability and privacy-preserving identity and role management can be achieved based on the above mechanisms.

Furthermore, the Blockchain-based knowledge base manages the overall information related to resource descriptions, system interactions and decision making, as well as applications using smart contracts and decentralised oracles. Therefore, at any point in time, the resources can be discoverable based on various contextual attributes and trust factors for Swarms' formation, as well as for the verification of system and application-level claims through external entities.

3.4 Resources Layer

In the Swarmchestrate concept, a resource is referred to as any computational resource ranging from a dynamically created virtual machine in the cloud, or a physical node that exists at any layer of the compute continuum, to an intelligent sensor with processing capability. Furthermore, a resource can also be a pre-deployed software service running on some dedicated hardware. These resources, shown at the bottom of Fig. 1, can be heterogeneous and can belong to different administrative domains. A resource can be characterised by various contextual attributes such as hardware characteristics, supported operating system, geographical location, mobility and battery power. Such characteristics are used to identify the suitability of a resource for a particular task at any given time.

In Swarmchestrate, a resource can be considered as a trusted resource, once it gets registered using DID-based identities and becomes able to produce proofs for each of the claims, i.e., the assertions made regarding its characteristics. These proofs are verified before the formation of the Swarm and/or when the resource joins a particular Swarm. The verification aims to establish the truth of the above claims to ensure the suitability of that resource for a particular Swarm.

4 Implementation of the Swarmchestrate Concept

The implementation of the above-mentioned fundamental concepts of Swarmchestrate is being achieved through a set of independent technical components that interwork to realise the overall vision of the project. Fig. 2 shows these components as pluggable blocks from which the overall framework is built. A short description of each component is provided below.

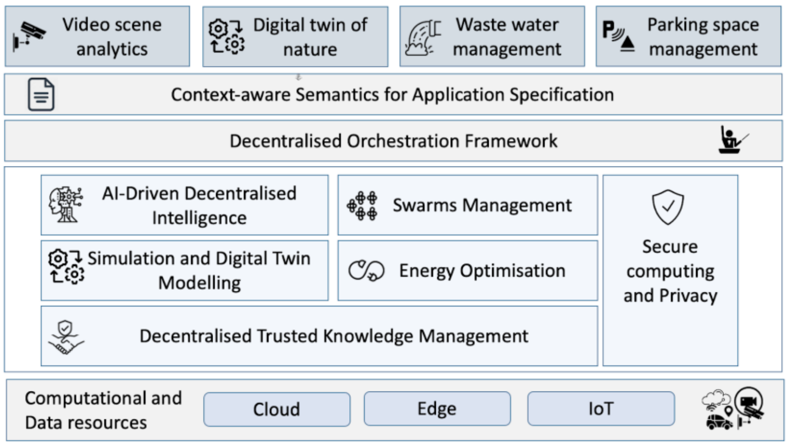
Video scene analytics
Digital twin of nature
Waste water management
Parking space management
Context-aware Semantics for Application Specification
Decentralised Orchestration Framework
AI-Driven Decentralised Intelligence
Swarms Management
Secure computing and Privacy
Simulation and Digital Twin Modelling
Energy Optimisation
Decentralised Trusted Knowledge Management
Computational and Data resources
Cloud
Edge
IoT

Fig. 2. Components of the Swarmchestrate Framework

Context-aware Semantics for Application Specification. This component supports the modelling of fog and edge nodes in addition to cloud resources. Moreover, it focuses on modelling entities that are necessary from the viewpoint of decentralisation and AI-enabled operations, such as the specification of context-aware attributes and aspects of reconfiguration. The modelling approach is being enhanced with suitable constructs related to the decentralised concept of the Swarmchestrate platform to allow the parallel deployment of topology by decentralised orchestrators in the application space.

Decentralised Orchestration Framework. The component is responsible for producing the overall integrated orchestration solution, in line with the decentralised vision of Swarmchestrate. The implementation of the component substantially transforms and extends an existing open-source Cloud-to-Edge orchestrator, called MiCADO [22], into a fully decentralised solution by utilising the developed algorithms and solutions implemented in other technical components of the framework.

Swarms Management. It is responsible for managing the ad-hoc formation of Swarms from the available resources, which are then used for the deployment and execution of an application. In contrast to traditional Swarm computing management tasks, the utilised resources are located in different dimensions of the Cloud-to-Edge continuum, which requires a peer-to-peer overlay network protocol to enable communication between them. Subsequently, this component implements such a protocol in each of the participating resources to enable a discovery and connection establishment process that automates the formation of a common overlay network, independently from the underlying network architectures.

AI-Driven Decentralised Intelligence. This component comprises of solutions for enabling effective and flexible utilisation of AI algorithms, in support of decentralised orchestration decisions. The AI algorithms serve two different but related purposes. First, the distributed algorithms are executed throughout the Cloud-to-Edge computing continuum to provide matchmaking functionality between the requirements of the application's microservices and the available resources. The matchmaking is performed both at deployment and also at run-time to support the continuous reconfiguration of applications to fulfil QoS requirements. Second, the AI algorithms support the Swarms Management module when considering the formation, reformation and interactions between multiple Swarms.

Energy Optimisation. The component provides an allocation schema for all application microservices onto the available resources, focusing on energy preservation and utilising the AI-driven decentralised intelligence described above. The calculation of the energy optimisation solution requires detailed input information about the available devices in the distributed system (technical aspects) and the applications to be executed (economic aspects). With this input, the component can consider cross-layer energy optimisation issues (e.g., hardware, software, and networking) for all types of Cloud-to-Edge devices and application-specific energy issues (e.g., urgency expressed by a high willingness to pay for computing services). As the calculation of the optimisation solution (even with the help of heuristics for multi-objective optimisation algorithms) is computationally too expensive, time-consuming, and not accounting for the highly dynamic environment of the envisioned Cloud-to-Edge ecosystem, continuous learning approaches will be applied.

Simulation and Digital Twin Modelling. This component is responsible for modelling decentralised self-organising orchestration services using simulation. The component is based on the open-source DISSECT-CF-Fog simulator [23], which is able to utilise a multi-layered Cloud-to-Edge infrastructure. However, new modelling constructs are being defined to incorporate decentralised resource management and decision-making by introducing an orchestration layer and a Swarm manager component to the simulation architecture. Later in the project, the simulator will be further enhanced into a digital twin solution that runs in parallel to the real system, evaluates its behaviour in real-time based on possible alternative scenarios, and initiates certain reconfiguration decisions, if required.

Decentralised Trusted Knowledge Management. The component provides a persistent and trusted knowledge base for the global management of application-descriptors and market-tradable resources and services in the Cloud-to-Edge continuum. It is based on standard compatible and blockchain-based self-sovereign identities that describe all participating entities in the fully distributed environment. The component is being implemented by following the principles of Decentralised Identities (DID) and Verifiable Credentials. as defined by the W3C [24].

Secure computing and privacy. The secure computing and privacy layer provides end-to-end security for the orchestrator and the targeted and managed applications. It utilises several modern encryption techniques, e.g. Functional Encryption (FE) and Hybrid Homomorphic Encryption (HHE), to analyse encrypted data stored in distributed locations as if they were unencrypted (i.e., in a privacy-preserving way). It also provides a decentralised trust management solution using blockchain-based FE mechanism to facilitate overall transparency and assure trust amongst the system, the distributed computing infrastructure, and the storage resources. Finally, it aims to provide an anonymous Sybil-resistant DID solution so that each entity can only get one ID.

5 Conclusions and Further Steps

Due to the increasing adaptation of the Cloud-to-Edge continuum by applications with complex and changing requirements, it is crucial to develop novel mechanisms for the management of large microservices-based applications in such environments. Traditional centralised application management and orchestration approaches are quickly becoming bottlenecks in these scenarios. To tackle this challenge, the Swarmchestrate project developed the concept of a novel fully decentralised application-focused orchestration framework that is based on Swarm computing principles and utilises distributed AI and self-sovereign identities for application life-cycle management.

The project starts in 2024, and after conducting a detailed analysis of currently available technologies, it will develop its framework using an incremental and iterative methodology. Swarmchestrate will demonstrate its results by reengineering an existing centralised orchestrator and implementing four real-life use cases.

References

1. Y. Jiang, Z. Huang, D. H. Tsang, Challenges and Solutions in Fog Computing Orchestration, IEEE Network (2018). doi:10.1109/MNET.2017.1700271.
2. O. Tomarchio, D. Calcaterra, G. D. Modica, Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks, Journal of Cloud Computing (2020). doi:10.1186/s13677-020-00194-7.

3 B. Costa, J. Bachiega Jr, L. R. de Carvalho, A. P. Araujo, Orchestration in fog computing: A comprehensive survey, ACM Computing Surveys (CSUR) 55 (2) (2022) 1–34.

4. Svorobej, S., Bendechache, M., Griesinger, F., Domaschka, J.: Orchestration from the Cloud to the Edge. The Cloud-to-Thing Continuum: Opportunities and Challenges in Cloud, Fog and Edge Computing 61–77 (2020)
5. Hong, C. H., & Varghese, B. (2019). Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. ACM Computing Surveys (CSUR), 52(5), 1-37.
6. Kumara, I., Mundt, P., Tokmakov, K., Radolović, D., Maslennikov, A., González, R. S., ... & Meditskos, G. (2021). Sodalite@rt: orchestrating applications on cloud-edge infrastructures. Journal of Grid Computing, 19, 1-23.
7. X. Masip-bruin et al., "Managing the cloud continuum: Lessons learnt from a real fog-to-cloud deployment," Sensors, vol. 21, no. 9, May 2021.
8. Verginadis, Y., Apostolou, D., Taherizadeh, S., Ledakis, I., Mentzas, G., Tsagkaropoulos, A., ... & Paraskevopoulos, F. (2021). Prestocloud: a novel framework for data-intensive

multi-cloud, fog, and edge function-as-a-service applications. Information Resources Management Journal (IRMJ), 34(1), 66-85.

9. Ullah, A., Kiss, T., Kovács, J., Tusa, F., Deslauriers, J., Dagdeviren, H., ... & Hamzeh, H. (2023). Orchestration in the Cloud-to-Things compute continuum: taxonomy, survey and future directions. *Journal of Cloud Computing*, *12*(1), 135.
10. Jimenez, L. L., & Schelen, O. (2020). HYDRA: Decentralised location-aware orchestration of containerized applications. IEEE Transactions on Cloud Computing, 10(4), 2664-2678.
11. Pires, A., Simão, J., & Veiga, L. (2021). Distributed and decentralised orchestration of containers on edge clouds. Journal of Grid Computing, 19, 1-20.
12. Masip-Bruin, X., Marín-Tordera, E., Sánchez-López, S., Garcia, J., Jukan, A., Juan Ferrer, A., ... & Kennedy, J. (2021). Managing the cloud continuum: Lessons learnt from a real fog-to-cloud deployment. Sensors, 21(9), 2974
13. Özyar, U. C., & Yurdakul, A. (2022, August). A Decentralised Framework with Dynamic and Event-Driven Container Orchestration at the Edge Espoo, Finland, 2022, pp. 33-40, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00017.
14. Nezami, Z., Zamanifar, K., Djemame, K., & Pournaras, E. (2021). Decentralised edge-to-cloud load balancing: Service placement for the Internet of Things. IEEE Access, 9, 64983-65000.
15. Ullah, A.; Dagdeviren, H.; Ariyattu, R.; DesLauriers, J.; Kiss, T.; Bowden, J. MiCADO-Edge: Towards an Application-level Orchestrator for the Cloud-to-Edge Computing Continuum. Journal of Grid Computing 2021, 19. doi:10.1007/s10723-021-09589-5.
16. Tsagkaropoulos, Andreas, et al. "Extending TOSCA for Edge and Fog Deployment Support. Electronics 2021, 10, 737.
17. K. Kaur and Y. Kumar, "Swarm Intelligence and its applications towards Various Computing: A Systematic Review," 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 2020, pp. 57-62, doi: 10.1109/ICIEM48762.2020.9160177.
18. I. Lera, C., et al. "Availability-Aware Service Placement Policy in Fog Computing Based on Graph Partitions," IEEE Internet Things J., vol. 6, 2019.
19. Sharma, V., Kumar, R., & Rathore, N.. Topological Broadcasting Using Parameter Sensitivity-Based Logical Proximity Graphs in Coordinated Ground-Flying Ad Hoc Networks. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl., 2015
20. Preukschat, Alex, and Drummond Reed. Self-sovereign identity. Manning Publications, 2021.
21. [on-line] Decentralised Identifiers (DIDs) v1.0: https://www.w3.org/TR/did-core/, accessed 10/12/2023.
22. Kiss T., Kacsuk P., Kovacs J., Rakoczi B., Hajnal A, Farkas A., Gesmier G., Terstyanszky G., MiCADO - Microservice-based Cloud Application-level Dynamic Orchestrator, Future Generation Computer Systems, Volume 94, pp 937-946 (2019).
23. Markus, A., Kertesz, A. (2021). Investigating IoT Application Behaviour in Simulated Fog Environments. In: Ferguson, D., Pahl, C., Helfert, M. (eds) Cloud Computing and Services Science. CLOSER 2020. Communications in Computer and Information Science, vol 1399. Springer, Cham.
24. Sporny, M., Longley, D., & Chadwick, D. (2022, March 3). Verifiable credentials data model V1.1. W3C. Retrieved March 18, 2023, from https://www.w3.org/TR/vc-data-model