



ICT-56-2020 "Next Generation Internet of Things"  
Grant Agreement number: 957218

Ref. Ares(2021)7497848 - 05/12/2021

# IntelliIoT

## Deliverable D4.1 IoT/edge infrastructure management (first version)

Deliverable release date	30/11/2021
Authors	1. Andreas Ziller (SIEMENS) 5. Nour Fendri (HOLO-LIGHT) 2. Arne Bröring (SIEMENS) 3. Jérôme Härri (EURECOM) 4. Martijn Rooker (TTC)
Editor	Andreas Ziller (SIEMENS)
Reviewer	Beatriz Soret (AAU), Martijn Rooker (TTC)
Approved by	PTC Members: (Vivek Kulkarni, Konstantinos Fysarakis, Sumudu Samarakoon, Beatriz Soret, Arne Bröring, Maren Lesche) PCC Members: (Vivek Kulkarni, Jérôme Härri, Beatriz Soret, Mehdi Bennis, Martijn Rooker, Sotiris Ioannidis, Anca Bucur, Georgios Spanoudakis, Simon Mayer, Filippo Leddi, Harshitha Chandregowda, Maren Lesche, Fragkiskos Parthenakis)
Status of the Document	Final
Version	1.0
Dissemination level	Public

## TABLE OF CONTENTS

Abbreviations.....	4
1 Introduction .....	6
1.1 Challenges & research questions .....	6
1.2 Outline of this Deliverable .....	7
2 Edge-based infrastructure platform .....	8
2.1 Edge-based infrastructure platform.....	8
2.1.1 Agricultural Vehicles.....	8
2.1.2 Healthcare – Wearable Devices .....	10
2.1.3 Manufacturing – Industrial Edge Devices.....	11
2.2 Key Technologies for Edge-Based Infrastructure Platforms.....	12
2.2.1 Virtualization.....	12
2.2.2 (Deterministic) Networking .....	12
3 Optimized allocation of IoT application functions on edge resources .....	14
3.1 Edge Architecture.....	14
3.2 Edge Manager .....	15
3.3 Edge Apps.....	15
3.4 Edge Orchestrator.....	16
3.4.1 Architecture.....	16
3.4.2 Web API .....	17
3.4.3 Computational Resource Manager .....	20
3.5 Interaction Edge Orchestrator with other IntellioT services .....	21
3.5.1 HyperMAS Infrastructure and Agents.....	21
3.5.2 Wired and wireless network resource reservation .....	21
3.5.3 HIL Service.....	22
3.5.4 5G MEC .....	22
4 5G MEC.....	23
4.1 MEC microservice architecture definition.....	23
4.2 MEC Service API definition .....	24
4.3 Edge Orchestrator integration .....	25
4.4 IAKM MEC integration .....	26
5 Remote Rendering with AR/VR Visualization at the Edge.....	29
5.1 AR/VR Remote Rendering.....	29
6 Conclusions & Outlook .....	30

---

7	Bibliography .....	31
---	--------------------	----

## Abbreviations

AI	Artificial intelligence
API	Advance programming interface
AR	Augmented reality
CFS	Customer facing service
CloT	Cloudification of internet of things
CNC	Computerized numerical control
COTS	Components off the shelf
CPU	Central processing unit
DB	Data base
DLT	Distributed ledge technology
DNS	Domain name system
ECU	Electronic Control Unit
ETSI	European Telecommunications Standards Institute
GPU	Graphic processing unit
HIL	Human in the loop
HTTP	Hypertext transfer protocol
HyperMAS	Hypertext multi-agent system
IAKM	Infrastructure-assisted knowledge management
IEEE	Institute of Electrical and Electronics Engineers
IEM	Industrial Edge Management
IIoT	Industrial IoT
IoT	Internet of things
IPC	Industrial PC
ISAR	Interactive Streaming for Augmented Reality
ISG	Industry specification group
IT	Information technology
MCP	Multi-core processor
MEC	Multi-access edge computing
MQTT	Message Queuing Telemetry Transport
OS	Operating system
OT	Operational Technology
PC	Personal computer

---

PCC	Project coordination committee
PERT	Project evaluation and review technique
PTC	Project technical committee
QoS	Quality of service
REST	Representational State Transfer
RNIS	Radio Network Information Service
SDK	Software development kit
SDN	Software defined networking
TD	Thing description
TSN	Time-sensitive network
UC	Use case
UI	User interface
VM	Virtual machine
VR	Virtual reality
WoT	Web of things
WP	Work package
XR	Extended reality

# 1 Introduction

This deliverable, being the first output of Task 4.1 ("IoT/edge infrastructure management"), aims to describe the first version of the design and implementation for the intelligent management of the IoT/edge infrastructure.

These efforts are aligned with the project's 1<sup>st</sup> objective pertaining to the creation of a self-aware and semi-autonomous multi-agent system over an optimized computation and communication infrastructure that manages compositions of IoT/Edge Devices in closed loop with the network.

Since the infrastructure management is the basis of the IntellioT framework, there is a strong interplay between Task 4.1 (and the content of D4.1) and several other Work Packages (WPs) and Tasks of the project. In more detail, Task 4.1 receives valuable input from Task 2.1 ("Use cases specification & Open Call definition"), Task 2.2 ("Technology analysis & requirements specification"), and Task 2.3 ("Architecture specification & interoperability") concerning the intricacies of the project's use cases, the requirements, as well as the architectural design.

The infrastructure management documented in this deliverable will be implemented in close consideration of the implementation of Task 3.1 "Interoperable, self-aware & semi-autonomous multi-agent systems" to enable the deployment of application functions as specified through the HyperMAS. Also, the Distributed Ledger Technology as implemented in Task 3.4 "Decentralized trust via secure interaction & contracts" will be integrated into the infrastructure management. Further, all tasks of WP4 ("Efficient, reliable and trustworthy computation & communication infrastructure") will be integrated with the implementation of the Task 4.1's infrastructure management, to enable the closed-loop management with the network management as developed in Task 4.2 "5G network functionalities" and Task 4.3 "Dynamic network management". Also, the output of Task 4.4 "Trustworthy infrastructure by design" will be closely considered to enable the integration of security and trust measures into the infrastructure.

Furthermore, at the start of IntellioT's Cycle 2, the results and feedback from the Cycle 1 demonstration and evaluation carried out within the three Tasks of WP5 ("System integration & evaluation") will provide vital input to Task 4.1 to allow the improvement and enrichment of the framework's infrastructure management. These relationships are visualised in the PERT chart shown in Figure 1.

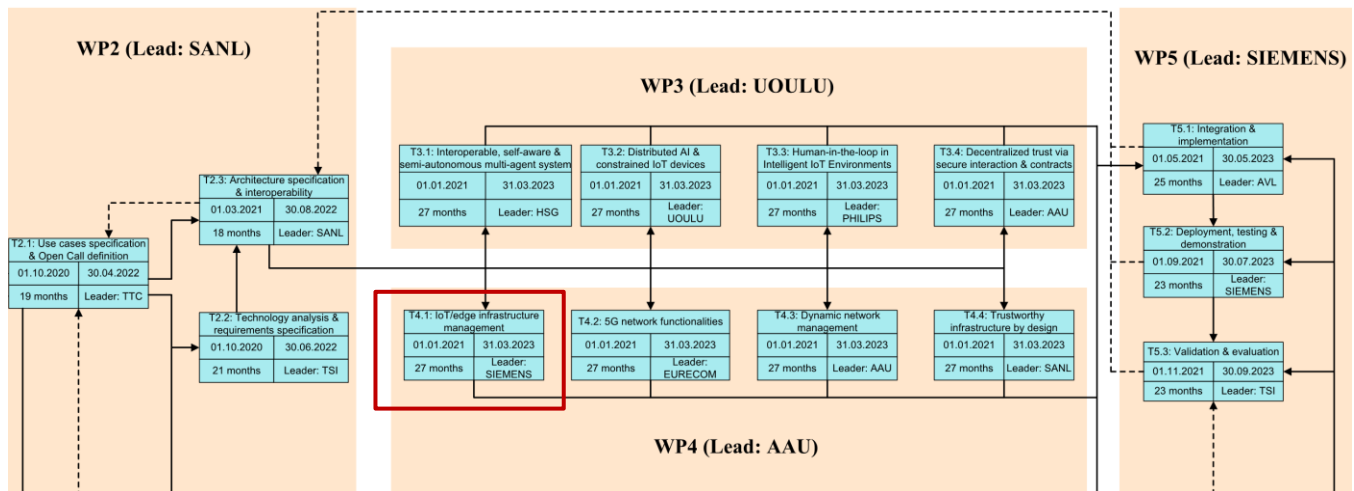


Figure 1: IntellioT's PERT chart with Task 4.1 highlighted through red rectangle.

## 1.1 Challenges & research questions

The management of the IoT/edge infrastructure is the foundation of the IntellioT framework. IoT applications are moving from the cloud into the edge, closer to the devices that produce and consume data, i.e., applications move

from the scalable and homogeneous cloud environment into a heterogeneous IoT/edge infrastructure. The IoT/edge infrastructure consists of numerous devices that are part of the IoT environment of the specific use case. E.g., in a manufacturing plant or a hospital, heterogeneous networked computing devices are used, such as dedicated Edge Devices [1], IoT devices with sensors/actuators, network gateways, over even powerful server racks provisioned through Multi-access Edge Computing (MEC) [2] of the 5G network. Collaboratively, the devices can execute IoT applications. Thereby, applications can be deployed as a whole on a single device, or applications can consist of multiple functions, which are deployed on different, connected devices.

Looking at the three use cases that are being developed in the IntellIoT project, various applications need to be deployed on the respective IoT/edge infrastructures of the use cases. In the agriculture use case, applications are, e.g., the AI of the semi-autonomous tractor, the interface to the controller of the tractor, or the VR application to take remote control. In case of the healthcare use case, example applications are the AI inference of the healthcare advisor, the AI re-training, or the interface for the physicians to study the patient data. For the manufacturing use case, the applications to be implemented include the controller of the engraver, the controller of the Computerized Numerical Control (CNC) milling machine, or the AI of the robot.

To run these various applications, IoT/edge devices need to be identified in the relevant IoT environment and efficiently selected to host the application function(s). However, IoT environments are highly dynamic (devices are added, removed, or re-located with time) and thus require advanced management. In addition, the local network of the IoT/edge infrastructure needs to provide reliable data transmission and low latency. For example, to enable tactile interaction between the human operator (using VR remote control) and the tractor controller. Hence, the computation part of the infrastructure (IoT and edge devices) needs to be managed in conjunction with the communication side of the infrastructure (i.e., the local 5G and TSN). Through the HyperMAS (developed in Task3.1), the functional planning and execution of composite applications will be enabled. The loose coupling of application functions enables non-functional network- and resource-based optimization. I.e., the functions of a composed IoT application can be deployed by considering the network infrastructure as well as timing, reliability, and other requirements of the application. In consequence, the computation and communication sides must be optimized in an integrated way. On the one hand, the deployment of application functions on IoT/edge resources need to be optimized under consideration of network constraints. On the other hand, the network must be dynamically managed and reconfigured to optimally serve the purpose of the application and the IoT/edge devices.

Thereby, the final research question for this task is: How to efficiently allocate application functions in an optimized way to distributed devices of a dynamically changing computation and communication infrastructure? This research question will only be fully answered with the finalizing of this task through the Cycle 2 of the project. Especially the mechanisms and algorithms for *optimizing* the allocation will be topic of this second part. In this first cycle and this first version of the deliverable, we focus on setting the basis with presenting the components for the infrastructure management and application allocation.

## 1.2 Outline of this Deliverable

To present the above, the deliverable is organized as follows:

- Section 1 (this section) describes the relation of this deliverable and the underlying Task 4.1 to the other tasks of the project. Further, it provides the motivation for this work.
- Section 2 gives an overview about edge computing with regard to the use cases within IntellIoT
- Section 3 presents the mechanisms for an optimized allocation of IoT application functions on IoT/edge resources.
- Section 4 describes the usage of and interplay with the 5G edge resources, i.e., the Multi-access Edge Computing, within the overall infrastructure management.
- Section 5 describes the specific case of the AR/VR rendering application.
- Finally, Section 6 provides the concluding remarks and pointers to the next steps.

## 2 Edge-based infrastructure platform

IoT solutions of the future are quickly developing towards an agglomeration of different technologies, like e.g., real-time control systems, artificial intelligence, security and safety concepts, computing, communications networks, etc. Additionally, more and more devices are being connected to each other and to handle all these different technologies at the devices (or sometimes called at the edge of the network), a new concept has been introduced called edge computing. Edge computing infrastructures can be made in many available versions. Within the IntellioT project, different infrastructures or devices are investigated as edge devices.

This chapter discusses how edge-based infrastructure platforms are being used inside the IntellioT framework and how different versions of edge-based infrastructure platforms can be deployed in the different domains. It will turn out that not a single approach can be applied in all domains, but dependent on the domain or the application targeted in the dedicated domain, a different infrastructure is required, which will also influence the deployment of the different applications on the platform or the management of the different edge devices available in the overall architecture.

### 2.1 Edge-based infrastructure platform

In the Industrial Internet of Things (IIoT) edge computing platforms provide a suitable architecture to cope with huge amounts of data arising from machinery and production sites. It leverages commercial-off-the-shelf (COTS) multicore processors (MCPs), which are typically found in information technology (IT), aiming to fulfil the stringent safety and real-time requirements stemming from the operations technologies (OT) domain. As a result, edge-based infrastructure platforms could support a convergence of IT and OT enabling new, cross-domain applications not only in manufacturing. Among others, IIoT applications pose challenges on the hardware/software stack that implements the edge computing architecture, which often can be found in Cloud Computing, such as scalability, reliability, availability, security, and integration of legacy systems. In Cloud Computing, we face such challenges by the extensive use of system virtualization. System virtualization, henceforth simply referred to as virtualization, allows to concurrently run several virtual machines (VMs) on a shared platform utilizing hypervisors. Sharing a physical host between multiple VMs, or guests, allows for a high level of scalability and availability as VM instances can easily be created, migrated, and destroyed on demand. At the same time, the hypervisor being in full control of the hardware and its virtual counterpart, enforces security by utilizing extensions of modern processors that allow for the spatial and temporal isolation of VMs. Finally, virtualization allows for the execution of legacy systems side by side with VMs running novel IIoT applications. In edge computing, virtualization, and deterministic communication (e.g., by using Time-Sensitive Networking (TSN)), are commonly accepted, and presumed to act as key enabler of IIoT. The usage of deterministic communication enables real-time control on all levels (from planning to shopfloor actuation) and is one of the key enablers for bringing real-time control to the shop floor. The deployment and usage of TSN will be described in more detail in deliverable D4.3 – Dynamic Network Management, belonging to Task 4.3.

Within the IntellioT project, it will be demonstrated that the IntellioT Framework can be deployed in a universal way in different domains, with different realizations of the available edge infrastructure. In the following subsection, a first identification of the edge-based infrastructures targeted in the domains will be presented and explained. Within the use cases and the second version of this deliverable a final version of the edge infrastructure will be provided.

#### 2.1.1 Agricultural Vehicles

Agricultural vehicles (e.g., tractors, harvesters, etc.) are becoming more and more computing-rich electronic systems, with multiple controllers / computers per vehicle. As autonomous operation of these vehicles is becoming more common and these systems are becoming more intelligent, more processing power and computing resources are needed to perform the required activities, like line following, autonomous decisions of where and when to harvest or to spray etc.

Within the IntellioT project, it will be investigated to equip a semi-autonomous tractor platform with multiple functionalities/applications that will be hosted on edge devices, specifically developed for so-called off-highway vehicles. These devices must be ruggedized to stand the conditions where these vehicles work on. These devices will

be mounted on the agricultural vehicle and will be used as the edge device for the vehicle. In the overall edge architecture for the agricultural domain, multiple entities (i.e., tractors, harvesters, drones, field sensors, base stations, etc.) can be present, as depicted in Figure 2.

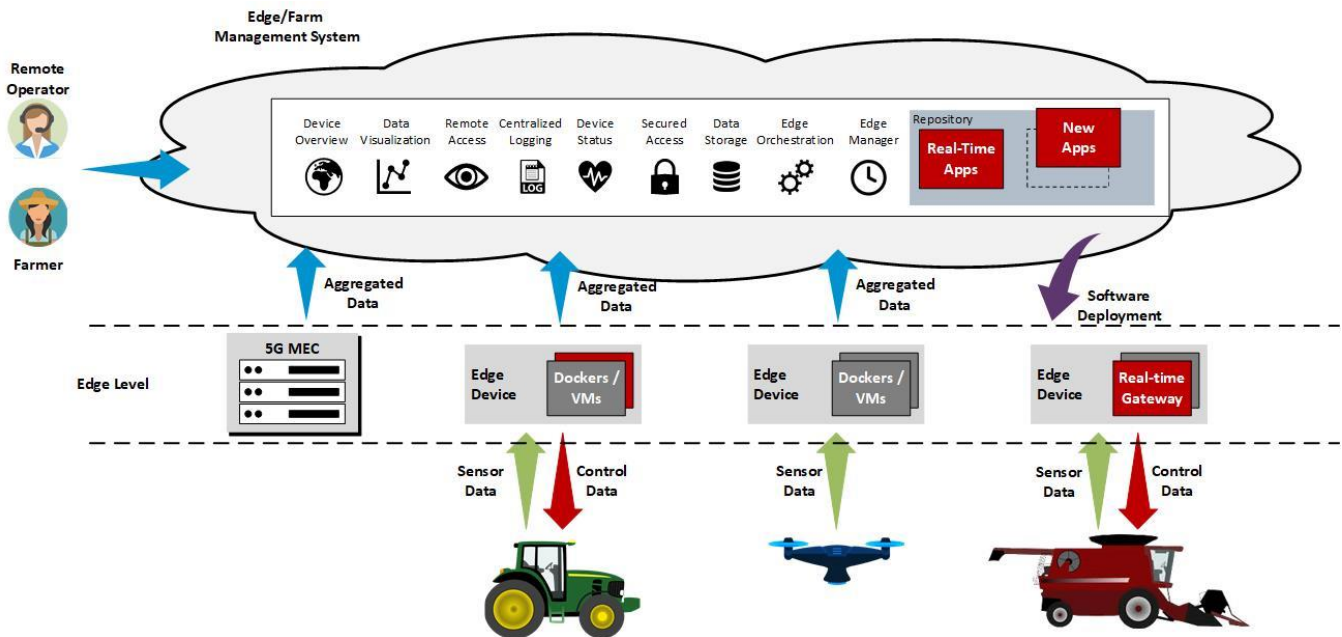


Figure 2: Edge-based infrastructure for Agriculture

Each entity can be interpreted as an individual edge device, hosting different (edge) applications for the dedicated entity. A tractor can have different applications hosted on its edge device, compared to a drone. The drone is maybe only sending sensed data to a cloud server or directly to the tractor, whereas the tractor has additional applications, for example to interpret the additional data and learn from the data. Thus, AI applications are executed on the edge device of the tractor. In a scenario like the agricultural domain, communication is of vital importance, but also one of the biggest challenges. Quite often, the communication infrastructure is not optimal (e.g., the connection to the cloud), there is sometimes direct device to device communication between the different entities or, in the worst case, no communication at all. Therefore, it is important that the edge device at the vehicle has access to all the data and knowledge required to perform the dedicated tasks at hand. Within the use case, to establish communication within the edge infrastructure, a 5G infrastructure will be investigated for communication between the entities and between the human operator and the different entities in the field. More information regarding the 5G infrastructure can be found in deliverable D4.2- 5G Network Functionalities. Additionally, the 5G MEC can be implemented to play the role of as an additional edge system within the infrastructure, functioning as a base station for the overall communication infrastructure and additionally hosting apps that are applicable to the whole infrastructure instead of a single entity. More information about the 5G MEC can be found in section 4.

For the Edge Management System (or can also be called farm management system), which will partly be based on the HyperMAS developed inside the project, different functionalities are of interest to this use case. For the farmer and/or the human operator, functionalities like device overview (tracking of all vehicles in the field), visualization of data from the individual vehicles, remote access, data logging and storing are of vital importance. The management of the individual edge devices (e.g., deploying new functionalities, potentially during run-time), or the orchestration of the

different edge devices (e.g., which vehicle is available at what time and can perform which tasks) are concepts that will be investigated within the dedicated use case.

### 2.1.2 Healthcare – Wearable Devices

Compared to the previous domain, within the healthcare domain the edge devices are considered different devices. Compared to the agriculture use case, the edge devices are more complex devices, and, in this situation, each individual device is an independent node in the network. In the specific situation that is being targeted in the healthcare use case, the edge devices are considered to be smaller handheld devices, like e.g., mobile phones, that are being used by patients. This is depicted in Figure 3. These lightweight devices are mostly small devices, with limited processing power, thus it is crucial that the applications running on these devices are also lightweight. As can be seen in Figure 3, the mobile phones will not have dockerized applications or any virtual machines, but Android packages in this case. Within the healthcare use case, there will be local AI running on the individual edge devices, whereas the global AI will be running on the 5G MEC (also an edge device). More information regarding the AI can be found in deliverable D3.2 – Distributed AI algorithms & methods.

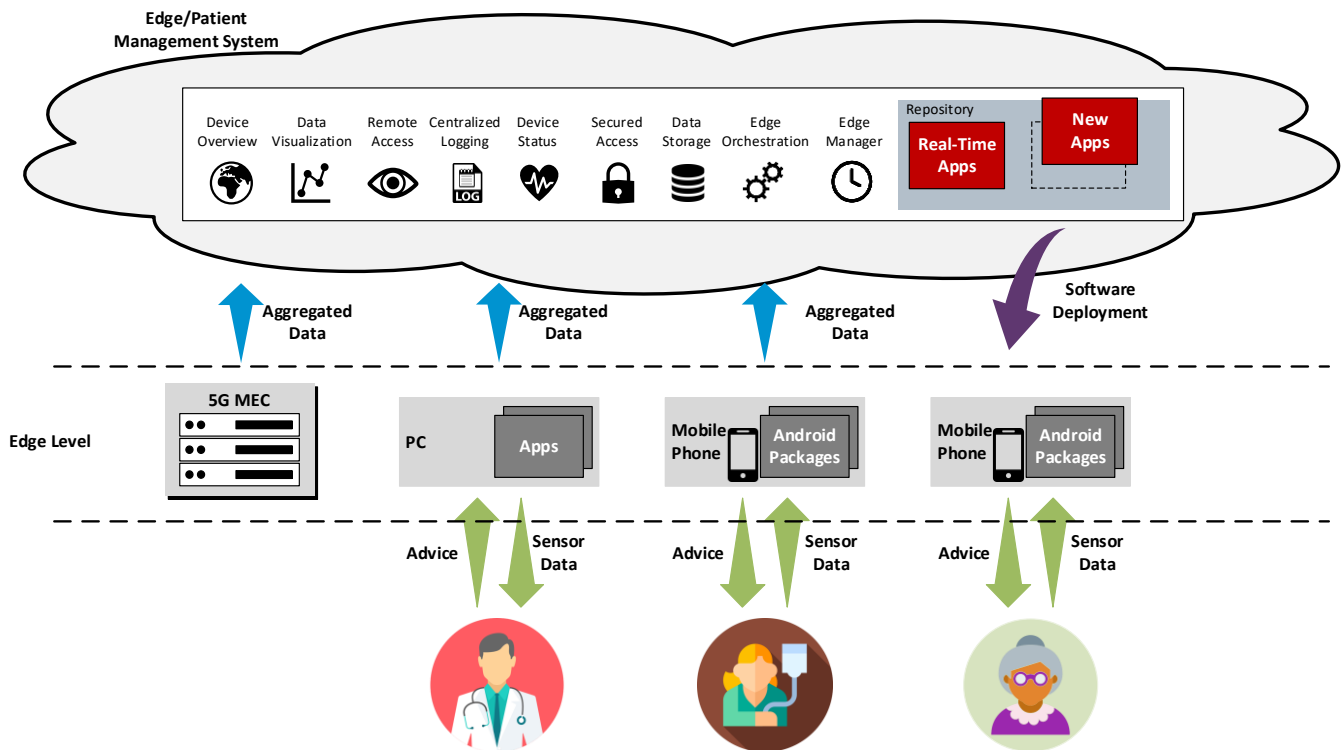


Figure 3: Edge-based infrastructure for Healthcare

The end users of the edge devices are in this situation human users, be it either patients, elderly people, or physicians. The patients or elderly people will have the mobile phones as edge devices with them and they collect the personal data from them. Additionally, through these devices, they will receive input from the physician regarding their medical situation and what is required of them.

The physician does not have an edge device but a PC at its office, where the data is being analyzed by the physician and advice can be send to the individual patients. Nevertheless, in Figure 3, the PC is included in the edge level, as similar applications can be run on it.

Functionalities that are offered at the edge management level are mainly the secure access to the data (logging and storing) and visualization of the data. The orchestration and management of the different edge devices can be of interest to the physician or the operator of the system, as updates of the devices can be required (faulty or outdated software, so new software can be pushed automatically from a central repository).

### 2.1.3 Manufacturing – Industrial Edge Devices

The basis for edge computing can be found in the automation pyramid, which has been common in many factory processes, where on the lowest level there were the actual shopfloor with the sensors and actuators and on the highest level the Enterprise Resource Planning. This classical approach poses some restrictions. The lower levels rely more on Operations Technology (OT) and the upper levels more on Information Technology (IT). The change of trying to close the gap between IT and OT has introduced the concept of edge computing in the industrial domain, resulting in an edge-based infrastructure as depicted in Figure 4.

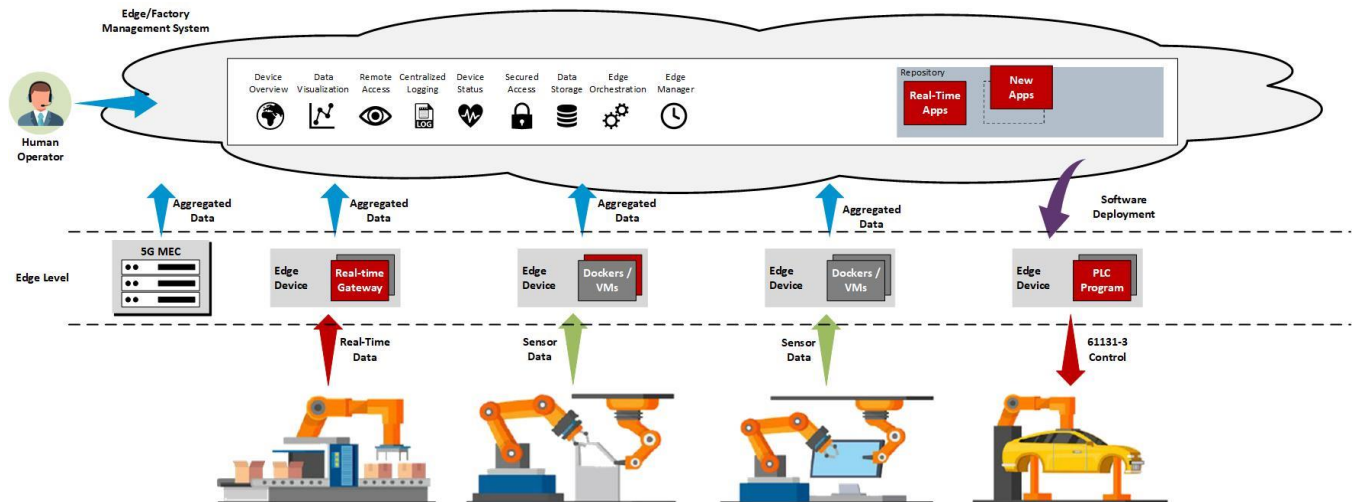


Figure 4: Edge-based infrastructure for Manufacturing

Although the infrastructure has many similarities with the two previous domains, agriculture and healthcare, there are some significant differences that can be mentioned. One of the biggest differences is that in the manufacturing domain, there is much more reliable communication available between the different edge devices, as they are typically indoor environments. Many systems are connected through wired communications, enabling a much safer and secure communication concept. Furthermore, on the shopfloor, computing power is often not so constrained as in other environments. Within the agricultural and healthcare domain, the available computing power is much more limited. Compare a smart phone with an industrial PC, or in a lesser form an ECU on an agricultural vehicle to an industrial PC. Additionally, one edge device can easily handle multiple machines, like e.g., multiple industrial robots or additionally a conveyor belt or even a complete manufacturing cell.

Therefore, we will explore additional aspects of edge computing in the manufacturing domain, being the most representative and versatile for the scope of this Task. One of the interesting parts that will be investigated within this domain is the concept of the Edge Manager and the Edge Orchestrator, where the overall infrastructure will be able to identify the optimal distribution of the Edge Apps to the available nodes within the edge infrastructure. Reallocating apps to different resources could potentially be done in runtime, as all the edge devices are connected to the same network and can easily exchange data, applications, and resources. Compared to the previous two domains, this will

raise many new challenges that will need to be investigated and will additionally have impact on the final infrastructure required for this domain and solution.

## 2.2 Key Technologies for Edge-Based Infrastructure Platforms

Two key technologies are identified for edge computing and edge-based infrastructures, namely virtualization and (deterministic) networking. These two topics are shortly described in the following sections.

### 2.2.1 Virtualization

Computing hardware is becoming more and more powerful, which is for example established by increasing the number of CPU cores per chip. To exploit the available processing power, virtualization has been developed as a technique to share the available hardware between multiple applications with a guarantee of bounded interference of the different applications on each other. As presented in Figure 5, virtualization is commonly distinguished in two different types.

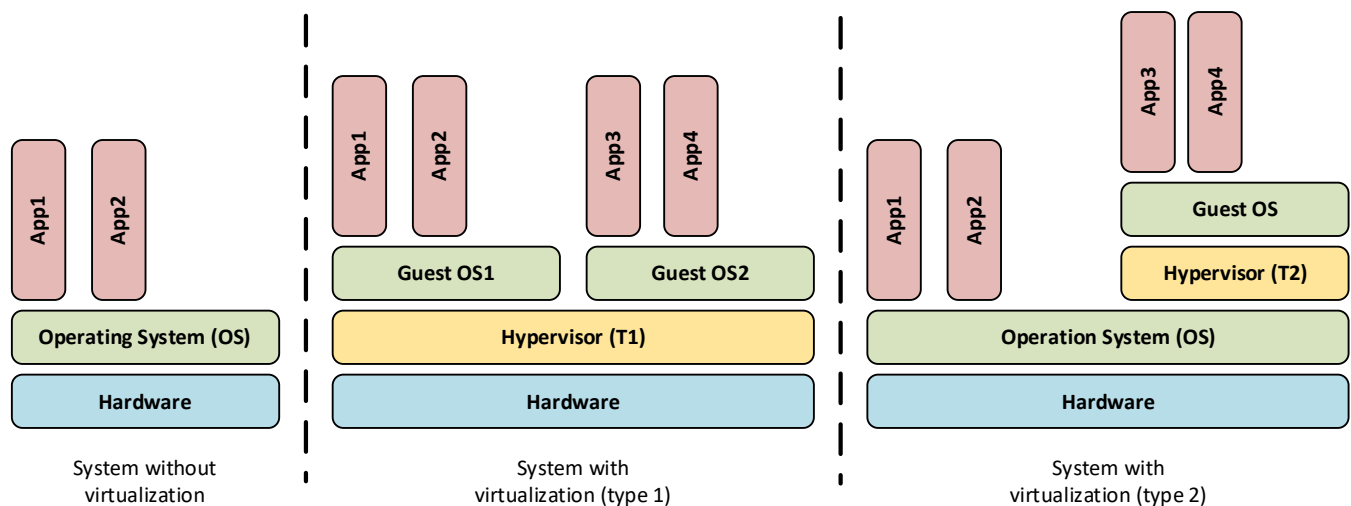


Figure 5: Infrastructures with and without virtualization

Type 1 (T1) hypervisors (also called native or bare-metal hypervisors) implement a software layer that is directly executed on the underlying hardware. This T1 hypervisor allows the implementation of one or multiple operating systems (OS), which are normally dubbed guest OS. The T1 hypervisor is normally configured in such a way that each guest OS only has a restricted access to the underlying hardware resources and the available memory. This means that each guest OS will be unaware of the hypervisor existence and will experience the restrictions of the hardware resources as the only hardware resources available to it. Hardware resources, like CPU or GPU cores, may also be shared between different guest OSs, in which case the hypervisor schedules the access to these shared resources.

Type 2 (T2) hypervisors (also called hosted hypervisors) form, as compared to T1 hypervisors, a software layer that is hosted on top of an operating system. This type of hypervisor is depicted in right part of Figure 5. Both solutions have their advantages and disadvantages for edge-based infrastructures, which also depends on the applications or domains that are being targeted. However, hypervisors are not always necessary, as concepts like dockerized applications can also be relevant for certain Edge Apps and can be hosted directly on the edge device itself, without the use of a hypervisor.

### 2.2.2 (Deterministic) Networking

As already mentioned in the subsections before, communication is of major importance to the functionality of the edge devices. There is the wireless communication between the edge devices and towards the cloud, which will be investigated within the project by deploying 5G communication infrastructures, enabling high volume data streams to

establish remote operation of agricultural vehicles in the field or remote control of robots on the shopfloor. The applicability of a 5G communication structure in these kinds of domains will be investigated more in detail in deliverable D4.2 – 5G Functionalities and D4.3 Dynamic Network Management

Next to the previous mentioned networking concepts, there is also the wired communication from the edge devices to the actual devices that are being controlled by the edge devices. As computation performance steadily increases of edge devices and they become more capable of directly controlling for example the devices on the shopfloor, or in the field, communication hardware must and is becoming more and more capable to perform real-time, deterministic communication that is needed to have a reliable, real-time communication. One of the communication technologies that has been steadily increasing its market share over the last decades is Ethernet, especially in the manufacturing domain, but also in other domains like automotive, off-highway and aerospace, to name a few. However, traditional Ethernet has been inferior for use in industrial applications as it lacks real-time response and reliability capabilities. Therefore, many new, real-time variants have been developed by the industry, like PROFINET, EtherCAT or TTEthernet, but these are all proprietary solutions, which causes the industry to use different solutions to cover all their needs.

To provide a consolidated version, in 2012 IEEE have started working on a solution to make a set of Ethernet technologies that will cover all the needs of the industry and is fit for use in OT operations. The Time-Sensitive Networking Task Group (IEEE 802.1 TSN) was founded with the goal to improve existing Ethernet standards and additionally to develop new standards to fulfill the deterministic requirements coming from the industry. This technology will be investigated in detail within the IntellioT project and more information about it can be found in deliverable D4.3 – Dynamic Network Management.

### 3 Optimized allocation of IoT application functions on edge resources

One goal in IntelloT is that functionality that cannot be provided by machines or devices itself is offloaded to remote computing resources. In IntelloT we are dealing foremost with computational resources located close to the clients at the edge of the network. Due to the proximity and the utilization of 5G and TSN, edge computing can benefit from short latency and high-capacity network links. In order to exploit the full performance, the landside infrastructure for computing and TSN networking must be highly optimized. For this reason, we introduce the Edge Orchestrator. This component is optimally allocating requested offloading services to computing resources inside the IntelloT edge infrastructure. It further guarantees optimality by dynamically reallocating resources. This also includes the enforcement of quality of service by reservation of resources inside the 5G and TSN network infrastructure.

#### 3.1 Edge Architecture

The edge architecture was introduced in 2.1. This section puts a close look on the architecture of the edge infrastructure with its components. Further, the basic concepts are described how the edge computing infrastructure is interfacing with other IntelloT components.

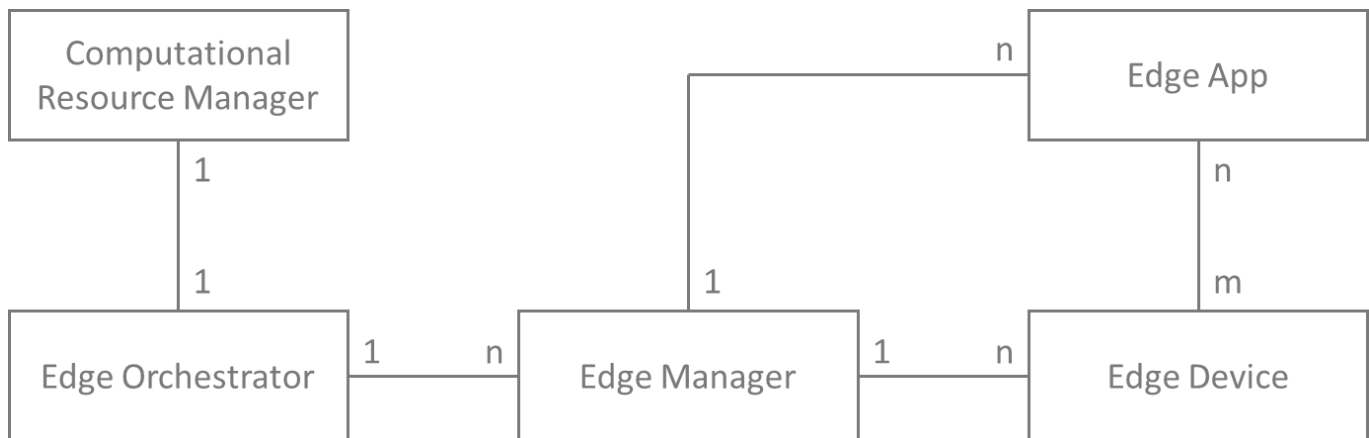


Figure 6: Main components of the edge infrastructure with cardinality

Figure 6 shows the relations of the main components of the edge infrastructure with their cardinalities.

The Edge Orchestrator manages edge resources on the granularity level of Edge Apps. Offloaded services are encapsulated in Edge Apps. It uses the computational resource manager to determine an optimal allocation based on metrics received from other IntelloT components. An allocation includes the concrete point in time to start and stop an Edge App on a specific edge device. To apply an allocation, the Edge Orchestrator updates the run state of an Edge App at the exact moment in time via a request to the Edge Manager.

The Edge Manager is responsible for the lifecycle management of Edge Apps as well as edge devices. The lifecycle management of Edge Apps includes onboarding, start, stop, install, remove operations. Lifecycle operations for devices include onboarding, restart, and shutdown operations. Device operations are not further discussed in this document as they are not an integral part of the dynamic activities of the edge orchestration. It is assumed that lifecycle operations on devices are part of the regular maintenance activities.

The components Edge Manager, Edge Device and Edge App in Figure 6 can be seen as abstract. Implementations of those components are complementing each other to form an Edge Management System. In IntellioT we support different types of Edge Management Systems. The Edge Orchestrator can manage different types of edge ecosystems as long as they support the same methodology with basic atomic lifecycle operations. However, it is not planned to mix Edge Manager, Edge Device and Edge Apps across different Edge Management Systems. For each supported edge environment, the Edge Orchestrator implementation requires a specific adapter, to map the requested operations on concrete request syntax of the edge environment. Currently we support the Siemens Industrial Edge ecosystem. The Industrial Edge Management (IEM), part of the Siemens Industrial Edge product line, will be the component acting as an Edge Manager. Figure 6 also includes the cardinality of the components. Our architecture foresees exactly one Edge Orchestrator using one Computational Resource Manager. However, the Edge Orchestrator can support  $n$  Edge Managers, which in turn manage  $n$  Edge Devices. An Edge App is managed by exactly one Edge Manager and can be deployed on  $m$  Edge Devices. However, an Edge App typically cannot be instantiated more than once on the same Edge Device. Here, Edge Apps are analogous to apps on a smartphone and not to applications on a PC. If multiple instances are required, Edge Apps must be provisioned multiple times with new identities.

In the following, for simplification we differentiate between the software and the hardware only if necessary as they can be assumed to be inseparably tight together. The edge runtime is explicitly identifying the software part of the Edge Device. For simplification purposes again, the Edge Device is synonymously identifying the hardware and the software part. All hardware components are multi-purpose industrial PCs (IPC).

The Edge Orchestrator is the central component. It automatically decides at which point in time on which Edge Device an app is or will be deployed. Accordingly, the Edge Manager is just executing those requests. Until now, the app deployment has been a manual operation triggered by a human user directly at the Edge Manager.

The Edge Device is the actual computing environment for Edge Apps. Edge devices differentiate themselves in the computation capacity, in the form factor, in the supported environmental conditions and the availability of hardware extensions, e.g., GPUs or security elements.

The hierarchy between Edge Manager, Edge Device and Edge App is a universal concept, which applies in all Edge Management Systems. However, the focus of the Edge Orchestrator lies on the orchestration of Edge Apps executed in a Siemens Industrial Edge environment. Therefore, the concepts are explained based on that example, although, as mentioned above, other types of edge environments might be supported by the Edge Orchestrator as well.

## 3.2 Edge Manager

As stated above, the Edge Manager is a standard component, which manages Edge Apps and Edge Devices. It supports atomic lifecycle operations. The Edge Manager in our architecture is a COTS, so we use it as it is and to modify it to get integrated. This gives us the freedom, that our Edge Infrastructure can support multiple Edge Managers from different vendors. As stated above, it is assumed that the Edge Manager offers an interface to manage Edge Apps and Edge Devices. The Edge Orchestrator will use this interface accordingly to dynamically relocate apps.

## 3.3 Edge Apps

Edge app is an umbrella term for applications executed in an edge computing environment. The term app implies that the application is isolated from its execution environment, so that it compromises neither the execution environment nor other parallel running apps. Such an isolation requires defined interface and advanced software concepts. Typically, Edge Apps are realized via containerization technologies like Docker. If the Edge App follows a micro service principle, multiple containers might form an app. Edge Apps cannot be instantiated more than once per device. This is due to the fact that the linking of the app to the outside world with routes and ports is statically defined on engineering time because Edge Apps typically provide defined services (e.g., with fixed ports).

In the example of the Industrial Edge, running Edge Apps are consisting of one or multiple docker containers, organized by Docker Compose [3]. They run on an Edge Device in an isolated environment. Containers of an Edge App can expose

ports on the host system. They can also run further isolated behind a nginx reverse proxy [4]. In the latter case the connectivity is via HTTP on standard ports. Depending on requirements, Edge Apps can run in privileged mode. For example, if access to network interfaces is required.

Edge apps are also isolated from each other. They run in their own networks and can communicate either via standard ports or with an internal communication system based on MQTT.

Edge apps are specified by a docker-compose.yml file and an app description. The docker-compose.yml describes the orchestration of the container compliant to Docker Compose. It corresponds to the regular specification with some limitations. For example, Docker volumes are not supported, and containers cannot be built by docker-compose. The app description includes meta data required for the provisioning on Industrial Edge plus meta data required for services within IntellioT, which will be added in cycle 2. The QoS specification will be part of the app description.

### 3.4 Edge Orchestrator

In cycle 1, we defined the edge infrastructure and the integration of the components of IntellioT. In the first iteration, the Edge Orchestrator will support static allocations and the integration with the HIL Service, the Communication Resource Manager and the HyperMAS. These services will be introduced in more detail in Section 3.5.

#### 3.4.1 Architecture

The Edge Orchestrator can apply its orchestration on different types of Edge Management Systems. As each Edge Management System has its own data model and service interface, an integration layer abstracts the specialties of each platform and transforms their API calls and data structure in a uniform data format on which the Edge Orchestrator operates. The basic concept is shown in Figure 7, the Edge Orchestrator receives data models and metrics from the edge managements system and vice versa sends command actions for lifecycle operations. It is agnostic to the internal organization of the Edge Management System. Apps are completely remaining in the original repository. The same applies to the execution environment of Edge Apps on the Edge Devices. Whether Edge Apps are containerized or bundled on other mechanisms is up to the Edge Management System.

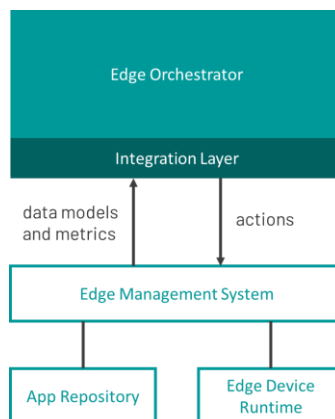


Figure 7: Edge Orchestrator with Integration Layer

As said, each Edge Management System has its own data model, however the model in most systems deals with entities like apps, app instances, devices as first-class elements. The Edge Orchestrator data model is aligned with the model of Industrial Edge, which follows the same paradigm.

As the Edge Orchestrator goes beyond the capabilities of a traditional Edge Management System, it augments the data model with specific information for orchestration. The data model in Figure 8 shows how the data model of the Edge Orchestrator references to the model of the Edge Management Systems. Original entities are referenced, so that they can be used for the interaction between Edge Orchestrator and Edge Management System.

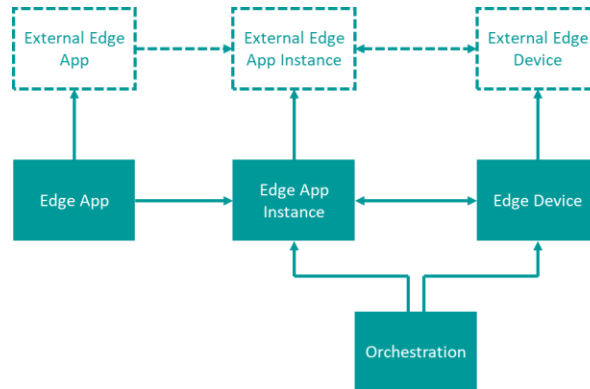


Figure 8: Edge Orchestrator basic data model

### 3.4.2 Web API

The Edge Orchestrator exposes multiple web APIs for services within IntellioT as well as for internal frontend applications, e.g., for web-UI-based configurations. The function set comprises functions specific for the orchestration as well as the alignment of Edge Orchestrator with connected Edge Management Systems. The concept explicitly includes configuration steps at the Edge Management System. For example, onboarding of Edge Apps or Edge Devices must happen in the Edge Orchestrator as well as in the respective Edge Management System. This is because, the low-level management of apps remains in the Edge Management System. It is not intended to expose the whole app and device lifecycle management on Edge Orchestrator level.

#### Target Edge Management API Configure the target edge management system ^

GET	<b>/targets</b> Get all edge management targets	▼
POST	<b>/targets</b> Add a new edge management target	▼
DELETE	<b>/targets/{target}</b> Delete a target edge management system	▼
POST	<b>/targets/{target}/connect</b> Connect to edge management system	▼
POST	<b>/targets/{target}/disconnect</b> Connect to edge management system	▼
GET	<b>/targets/{target}/status</b> Get status information to the edge management system	▼

Figure 9: Target Edge Management API

The Target Edge Management API in Figure 9 manages the registrations of Edge Management Systems as targets for the edge orchestration. With this API, connections to Edge Management Systems can be established and released. The target id of an Edge Management System will be used for some web service functions to identify the Edge Management System.

## App API Manage apps



GET	/apps	Get all apps	▼
POST	/apps	Onboard a new app	▼ ↵
GET	/apps/{appId}	Get app per id	▼
DELETE	/apps/{appId}	Delete app per id	▼

Figure 10: App API

The App API (Figure 10) is used to manage Edge Apps. It includes functions for listing, onboarding, and deleting of apps. An app on Edge Orchestrator comprises a reference to its equivalent in the target system and is augmented with meta information for IntellioT.

The onboarding of Edge Apps is an example, where not the complete function set of the target Edge Management System is exposed in Edge Orchestrator level. Before an Edge App can be onboarded with the App API of the Edge Orchestrator, it must be onboarded and uploaded to the app repo with the original Edge Management System.

## Device API Manages devices



GET	/devices	Get all devices	▼
POST	/devices	Add a device	▼
GET	/devices/{deviceId}	Get device per id	▼
DELETE	/devices/{deviceId}	Delete device per id	▼

Figure 11: Device API

The Device API is used for the onboarding of devices. Again, devices must prior be onboarded on the target Edge Management System side and referenced here. This API gives an overview of devices, which can be selected as targets for orchestrations.

## Orchestration API Manages orchestration



GET	/orchestrations	Get all orchestrations	▼
POST	/orchestrations	Add an orchestration	▼
GET	/orchestrations/{orchestrationId}	Get orchestration per id	▼
PUT	/orchestrations/{orchestrationId}	Update an orchestration	▼
DELETE	/orchestrations/{orchestrationId}	Delete orchestration	▼
POST	/orchestrations/{orchestrationId}/activate	Activate orchestration	▼
POST	/orchestrations/{orchestrationId}/suspend	Suspend orchestration	▼
POST	/orchestrations/{orchestrationId}/resume	Resume a suspended orchestration	▼
POST	/orchestrations/{orchestrationId}/terminate	Terminate orchestration	▼

Figure 12: Orchestration API

The Orchestration API provides functions for the creation and management of jobs to orchestrate Edge Apps. Edge Apps are instantiated based on orchestration jobs. An orchestration job includes the app plus additional information, which are required for the orchestration, e.g., service levels.

Orchestration jobs are inactive on creation. To start the orchestration, the "Activate Orchestration" command must be sent. With "Suspend Orchestration", an orchestration can be paused. Edge apps will be stopped, accordingly. With "Resume Orchestration", a suspended orchestration can be continued. "Terminate Orchestration" deletes an orchestration and release the resources on the Edge Device accordingly.

The format for specifications for new orchestration jobs is shown in Figure 13. It includes the ids of the apps, which should be part of the orchestration plus the devices on which orchestration might be dynamically deployed. The mode of the orchestration can be automatic or static. In the static mode, the apps are statically assigned to target Edge Devices. In the dynamic mode, Edge Apps are orchestrated based on optimization constraints. To meet different availability goals, preparation steps can be specified to reduce the handover delay when an Edge App is moved to another Edge Device. Furthermore, QoS requirements can be defined. Those include maximal delays, jitter, and minimal throughput, which will be guaranteed end-to-end between client and Edge App by the Edge Orchestrator. These requirements are translated out into concrete reservations, which are defined for network elements on the path between clients and Edge Apps.

```
{  
  "appIds": [  
    "string"  
  ],  
}
```

```
"targetDeviceIds": [  
  "string"  
],  
"mode": "static" or "dynamic",  
"availability": "basic", "preload" or "prestart",  
"qosRequirements": {  
  "e2eLatencyMillis": number,  
  "e2eJitterMillis": number,  
  "e2eThroughputMbps": number  
}  
}
```

Figure 13: Orchestration specification format

Status messages for Orchestration jobs, which are already running, include a richer data set. In Figure 14, the additional data fields are depicted: It includes the list of running instances of the specified Edge Apps. These could be used, e.g., to get information about the health. Starting and stopping will happen automatically as part of the orchestration.

```
{  
  # include all fields from Orchestration Specification  
  "orchestrationId": "string",  
  "state": "inactive", "active", "suspended", "terminated" or "failed"  
  "instanceIds": [  
    "string"  
  ],  
  "createdAt": "date time string",  
  "updatedAt": "date time string",  
  "reservations": [  
    {  
      "linkType": "5G" or "TSN",  
      "clientReference": "string",  
      "referenceType": "ipAddress" or "macAddress",  
      "maxLatencyMillis": number,  
      "maxJitterMillis": number,  
      "minThroughputMbps": number  
    }  
  ]  
}
```

Figure 14: Orchestration status message

## Instance API Manage app instances



GET	/instances	Get all app instances	∨
GET	/instances/{instanceId}	Get app instance	∨

Figure 15: Instance API

The Instance API (Figure 15) gives insights on running apps. As the lifecycle of Edge Apps are managed on the basis of orchestration, no lifecycle operations are exposed in this API.

### 3.4.3 Computational Resource Manager

The Computational Resource Manager will be defined in Cycle 2. Its purpose is to determine optimal allocations for Edge Apps meeting QoS requirements. It will take as input live metrics from the 5G system and from the TSN controller.

### 3.5 Interaction Edge Orchestrator with other IntellioT services

As the Edge Orchestrator is part of the IntellioT system, it interacts with various other IntellioT service like the HyperMAS Infrastructure, Agents that run within that infrastructure, the HIL Service, the 5G MEC and the Communication Resource Manager.

#### 3.5.1 HyperMAS Infrastructure and Agents

The HyperMAS Infrastructure interacts with the Edge Orchestrator to learn about the availability of Edge Apps with their deployment vectors. Furthermore, it initiates the orchestration on demand, e.g., because the specific service is required for a production goal. In the diagram in Figure 16 is shown, how the HyperMAS Infrastructure and the Edge Orchestrator are interacting in the example of the Engraver service instantiation.

In the first step, the Edge Orchestrator registers at the HyperMAS Infrastructure, that a specific Edge App is available, but not yet instantiated. It sends a W3C WoT Thing Description Template that includes the input- and output-schemas but not form bindings to the HyperMAS Infrastructure. At run time of the system, Agents that run in the HyperMAS Infrastructure request the instantiation by calling the Activate Orchestration service function (step 2). In response, the Edge Orchestrator instantiates and executes the Edge App (see Orchestration API in Section 3.4.2; Step 4) and then sends a W3C WoT TD that corresponds to the instantiated and deployed Edge App (i.e., it contains a concrete form binding; Step 3). The engraving job is then directly executed in step 5 on the instance of the Engraver app. This deliverable includes the basic concept for the protocol and will be further refined and demonstrated in Cycle 2. For example, the decommissioning of apps will be added.

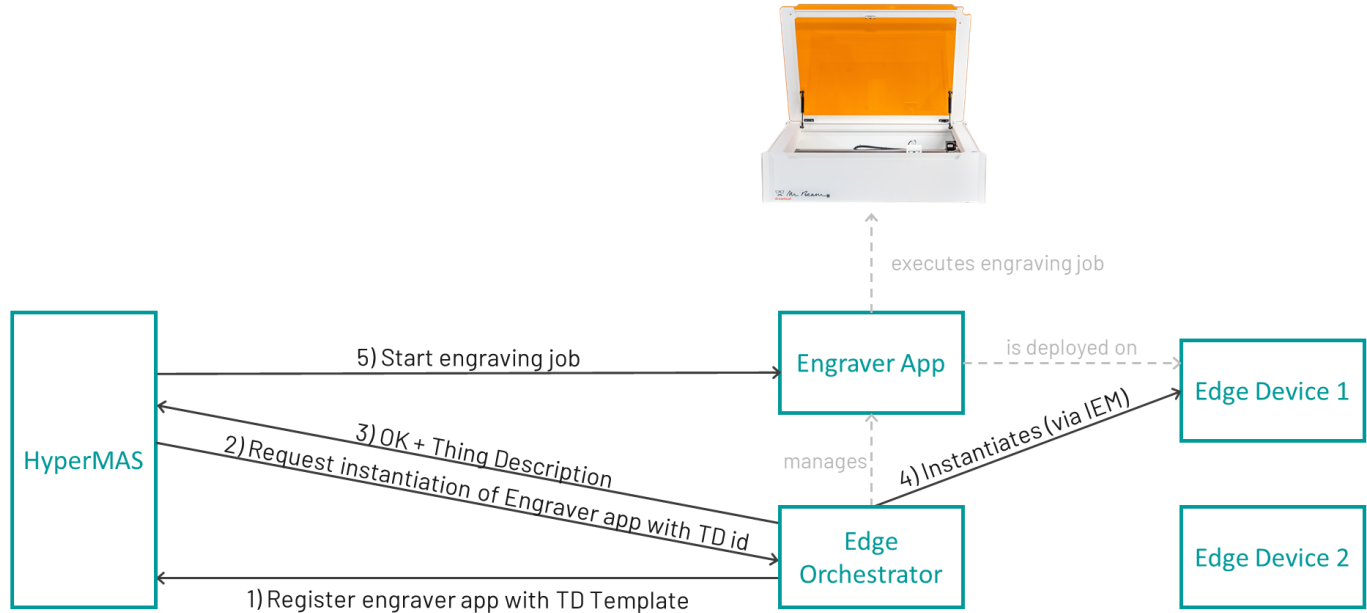


Figure 16: Interaction Edge Infrastructure with HyperMAS

#### 3.5.2 Wired and wireless network resource reservation

The Edge Orchestrator reserves resources in the network to meet the real-time requirements of Edge Apps. These resources are reserved per Edge App for the network path between Edge App and clients. If the Edge App is relocated

or new clients appear, the resource reservation will be updated accordingly. In the example of the manufacturing use case, the network path can include 5G and/or TSN links, i.e., wireless, and wired resources. Reservations for 5G links are sent to the Communication Resource Manager, those for TSN links are sent to the TSN Controller. Both software components are further described in D4.3 "Dynamic Network Management". Reservation requests include high-level QoS parameters like throughput, latency, or priority. The dynamic reservation of communication resources will be part of cycle 2.

### 3.5.3 HIL Service

The HIL service is summoned on the event of the involvement of a human operator. If the production process supporting AI cannot proceed in the manufacturing use case, for example, the HIL service requests the instantiation of the HIL application at the Edge Orchestrator together with the reservation of communication resources to establish a real-time communication link between service operator, HIL application and the machine (in this case the robot). The Edge Orchestrator, in turn, will instantiate the requested services and contact the Communication Resource Manager to execute the reservations. The amount of communication resources to be reserved will be defined by the specification of the Edge App. It depends on the characteristic of the application. In Figure 17 the interaction is depicted.

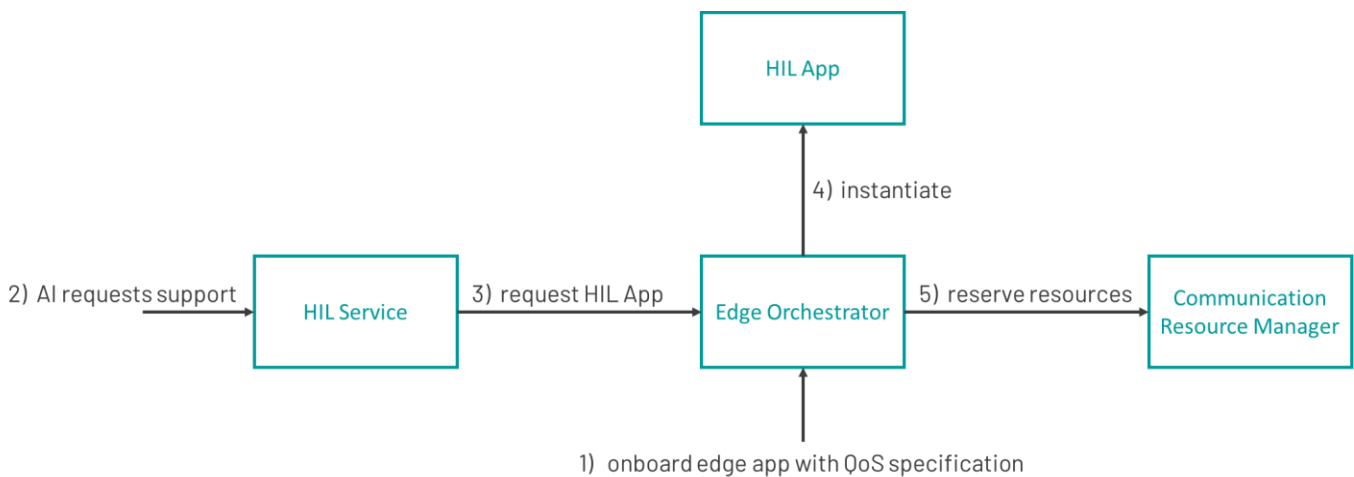


Figure 17: Interaction HIL Service with Edge Orchestrator

### 3.5.4 5G MEC

The services of the 5G MEC will be used to retrieve live performance and status metrics from the 5G radio system. They are required as input values for the Computational Resource Manager. More details regarding the 5G MEC will be provided in Section 4.

## 4 5G MEC

Multi-access Edge Computing (MEC), which was formerly known under the name of Mobile Edge Computing (MEC), is a network paradigm and set of APIs defined by the ETSI to provide 5G network and 5G application services closer to the consumer. In a 5G architecture, a MEC is associated with one or multiple 5G gNB and has a very similar deployment and lifecycle mechanisms as edge computing. MEC can therefore be considered as a specific case of edge computing when applied to 5G networks, yet with specific services, such as Radio Network Information Service (RNIS), 5G traffic re-routing or 5G traffic shaping. RNIS provides 5G network information to external entities, which could be used for instance to optimize the deployment of Edge Apps in 5G-connected end units. The traffic re-routing service is a native service, which re-routes 5G traffic originally targeted to a public/private internet to a local MEC app. Finally, the traffic shaping service aims at changing and adapting traffic stream encoding to support the 5G network quality.

In IntellioT, the objective of the 5G MEC is to orchestrate the hosting and lifecycle of low latency Edge Apps for the 5G network, defined as MEC apps. Various Edge App will be hosted in a 5G MEC, such as a DLT agent component, the 5G Communication Manager component or the IAKM server component. According to the ETSI MEC architecture, MEC apps are deployed by a MEC orchestrator. However, in IntellioT, the Edge Orchestrator will also be in charge of deploying MEC apps.

### 4.1 MEC microservice architecture definition

The ETSI ISG MEC defined a MEC ecosystem composed for different entities as depicted on Figure 18. We next describe each of them, as well as the interfaces (reference points), and explain how this is mapped to and utilized in the IntellioT framework.

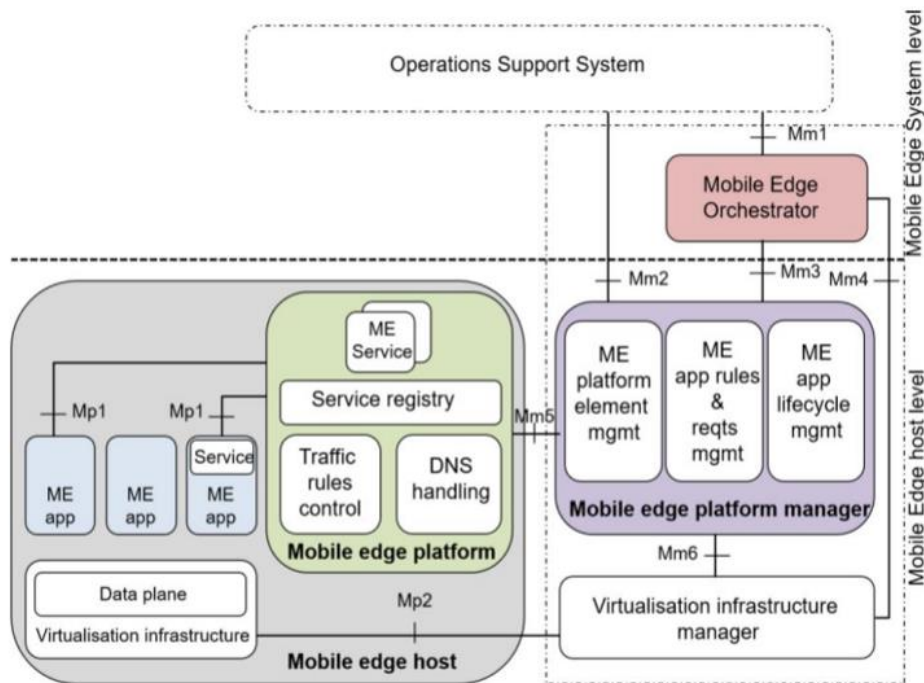


Figure 18: ETSI ISG MEC

- **Operations Support System (OSS)** - The Operations Support System may be considered as an operator. It receives requests via a customer facing service portal (CFS) and from device applications for instantiation or termination of applications and decides on the granting of these requests. Granted requests are forwarded to

the MEC orchestrator for further processing. In IntellioT, the industrial Edge Orchestrator would act as an OSS, receiving Edge apps requests from external entities (e.g., HyperMAS).

- **MEC Host** - The MEC host is an entity that contains the MEC platform and a Virtualisation infrastructure which provides compute, storage, and network resources for the MEC applications.
- **Virtualization Infrastructure & Manager<sup>1</sup>** - The Virtualisation infrastructure manager allocates, manages and releases virtualized (compute, storage, and networking) resources of the virtualization infrastructure (e.g., deploy a software image). The virtualization infrastructure acts as a Network Virtualization Function Infrastructure, and may execute the traffic rules received by the MEC platform and routes the traffic among applications, services, DNS server/proxy, 3GPP network, other access networks, local networks, and external networks.
- **MEC Platform** - The MEC platform is responsible for offering an environment, where MEC applications can discover, advertise, consume, and offer MEC services. The MEC platform also offers access to persistent storage as well as MEC services. The Mosaic5G MEC platform will be used in IntellioT, and Edge Apps will use its MEC services.
- **MEC Platform Manager** - The MEC platform manager is responsible for managing the life cycle of MEC applications, as well as managing application rules, such as authorization, traffic rules or DNS. Its role will be taken by the IntellioT Industrial Edge Manager.
- **MEC Orchestrator** - The MEC orchestrator has critical roles to the MEC ecosystem, as it selects the appropriate MEC host for specific MEC applications and is also in charge of triggering the deployment and displacement of MEC applications between MEC hosts. The MEC orchestrator primarily connects to the MEC platform manager. In IntellioT, the MEC orchestrator has similar functions as the Edge Orchestrator component, accordingly, the IntellioT Edge Orchestrator will act as MEC orchestrator.
- **MEC Applications** - MEC application runs as a containerised application and can interact with the MEC platform to consume and provide MEC services. In IntellioT, MEC applications have similar functions as Edge Apps.

## 4.2 MEC Service API definition

The MEC architecture described in Figure 18 shows several API, also called reference points, which provide standardized structure to exchange information between the various entities. The scope of the reference points depends on the target entity.

Reference points to a single entity MEC Platform:

- **Mp1** - The Mp1 (Mec Platform 1) reference point between the MEC platform and the MEC applications provides service registration, service discovery, and communication support for services. It also provides other functionality such as application availability, session state relocation support procedures, traffic rules and DNS rules activation, access to persistent storage and time of day information, etc. This reference point can be used for consuming services and providing service specific functionality.

---

<sup>1</sup>In this current version, IntellioT will not use virtualization infrastructure in the UC demonstrators.

- **Mp2** - The Mp2 (Mec Platform 2) reference point between the MEC platform and the data plane of the Virtualisation infrastructure is used to instruct the data plane on how to route traffic among applications, networks, services, etc. This reference point is not specified by the MEC specification.

In IntelloT, as only one MEC platform is used and no virtualization platform is employed, only the Mp1 endpoint will be developed.

Reference points to the MEC Management:

- **Mm1** - The Mm1 reference point between the multi-access Edge Orchestrator and the OSS is used for triggering the instantiation and the termination of MEC applications in the MEC system.
- **Mm2** - The Mm2 reference point between the OSS and the MEC platform manager is used for the MEC platform configuration, fault, and performance management.
- **Mm3** - The Mm3 reference point between the multi-access Edge Orchestrator and the MEC platform manager is used for the management of the application lifecycle, application rules and requirements and keeping track of available MEC services.
- **Mm4** - The Mm4 reference point between the multi-access Edge Orchestrator and the Virtualisation infrastructure manager is used to manage Virtualised resources of the MEC host, including keeping track of available resource capacity, and to manage application images.
- **Mm5** - The Mm5 reference point between the MEC platform manager and the MEC platform is used to perform platform configuration, configuration of the application rules and requirements, application lifecycle support procedures, management of application relocation, etc. This reference point is not specified by the ETSI MEC.

In IntelloT, as virtualization platforms will not be used, Mm4 will not be developed. Although not being identical, the Mm1-Mm3 endpoints correspond to similar APIs as for the Edge Platform. Accordingly, IntelloT will employ the Edge Computing Platform corresponding APIs for the Mm1, Mm2 and Mm3 endpoints. Finally, the Mm5 endpoint is important for the interaction between the Edge Computing Platform and the MEC platform. As the Mm5 is not standardized by ETSI MEC, IntelloT will propose an API for Mm5.

### 4.3 Edge Orchestrator integration

Considering that the management side of the MEC architecture has similar objectives and functions as an Edge Computing architecture, IntelloT will rely on the Edge Orchestrator as well as Edge Management functions for the management aspect of the 5G MEC. Moreover, the MEC applications will be considered as Edge Apps, although each Edge App requiring to be deployed on a MEC platform will need to implement the Mp1 endpoint. Accordingly, the integration of the MEC architecture within IntelloT Edge Computing architecture is depicted on Figure 19.



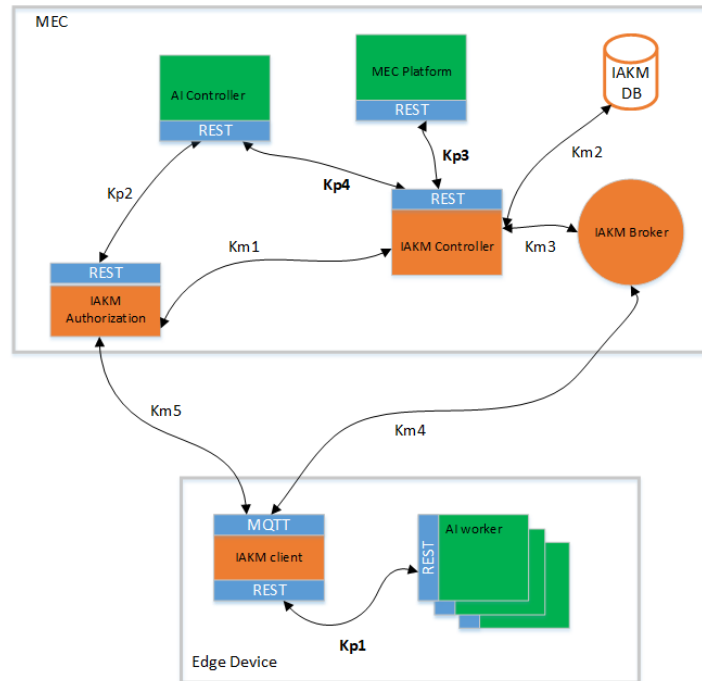


Figure 20: IAKM architecture

The green components are external to the IAKM component, which is composed of 4 major sub-components:

- **IAKM Authorization** – This module grants authorizations to IAKM clients as well as any external components to connect to the IAKM. This includes external IAKM instances, such as MEC-to-MEC or MEC-to-Cloud IAKM data connection.
- **IAKM Controller** – This module handles knowledge flow within the IAKM as well as providing external services to the MEC platform or to other MEC applications.
- **IAKM DB** – This module is a database, responsible for storing knowledge and providing querying mechanisms to extract the appropriate knowledge according to a described context.
- **IAKM Broker** – This module is a data broker to deliver knowledge to subscribed and registered IAKM clients.

In IntelliIoT, the IAKM component will interact with at least the MEC platform, the AI components (controllers/workers) and potentially also the DLT edge component. All interactions are based on REST interfaces.

The IAKM architecture contains four platform-type REST APIs:

- **Kp1** – Provides REST services to subscribe for using or training AI models.
- **Kp2** – Provides REST services for an external entity to authenticate with the IAKM.
- **Kp3** – Interacts on REST with the MEC platform. This corresponds to the MEC Mp1 endpoint.
- **Kp4** – Provides REST services to interact with MEC applications.

The IAKM architecture also describes five management-type APIs:

- **Km1** – Receive and validate the authorization to access knowledge.
- **Km2** – Responsible to store knowledge by the IAKM.

- **Km3** – Notifies the IAKM server of subscriptions to specific services as well as send knowledge to the subscribed IAKM agents.
- **Km4** – Data exchange API between the IAKM server and client components
- **Km5** – Authentication API between the IAKM client and the IAKM server components.

The set of IAKM core components will be deployed as a MEC application and will be orchestrated by the Edge Orchestrator.

## 5 Remote Rendering with AR/VR Visualization at the Edge

AR/VR visualization is an example for an application at the edge. AR/VR renderings are mostly more functional than impressive compared to the sharp ones produced by modern PCs that have the capability to deliver high graphics within milliseconds thanks to their powerful GPUs and CPUs. A self-sufficient XR data goggle can only fluently visualize up to 1,000,000 polygons. Therefore, when the application's requirements pass that range (i.e., UC1 & UC3), the user must sacrifice his user experience (drop in quality) to continue a normal functionality for the application. To overcome this limitation, the key lies in the usage of external resources through remote rendering.

### 5.1 AR/VR Remote Rendering

To address the limited computing resources of mobile hardware devices we use ISAR. ISAR allows to outsource the rendering process to an external server (or Hypervisor) for potentially unlimited performance. It permits the user to decide how much resources you want to deploy while enabling the visualization of GPU-intensive graphics effects and interaction with highly polygonal, data-intensive content.

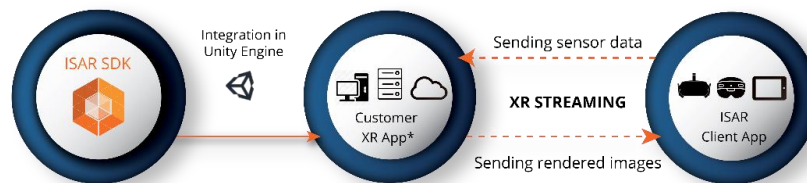


Figure 21: High-level view on system components

The ISAR SDK is added to the Customer's XR Application through the Unity Engine. The application could be hosted anywhere (Cloud, VM or Bare metal) to utilize the resources provided. Then, using the ISAR Client App (on multiple devices), the signalling between Client & Server starts and once successful, a peer-to-peer connection is established where the Server side would stream rendered images to the Client which in return sends back sensory data.

## 6 Conclusions & Outlook

In this deliverable D4.1 we firstly defined the terms around *computational* side of the IntellioT infrastructure and their understanding among the involved partners and we discussed the first development cycle of implemented components. We introduced the now established basis for our IoT/edge computing infrastructure management and its alignment with the 5G system and its MEC management, which are detailed in D4.2. Through the central component of the Edge Orchestrator, we have now the ability to provide agile application and service orchestration that can be triggered by the HyperMAS, developed in Task 3.1. As an example application, we further introduced the remote rendering of an AR/VR visualizations to underline the needs for edge computing facilities.

In the Cycle 2 of the IntellioT project, we will build up on this infrastructure management and we will focus on dynamically adjusting this infrastructure. Thereby, a key component will be the optimized allocation of workload to computing resources (i.e., mapping of HyperMAS artifacts of an IoT application to devices), which will be realized by the Computation Resource Manager that is directly linked with the Edge Orchestrator. Building up on prior work [9], we will develop a flexible algorithmic framework to calculate the allocations at runtime and able to dynamically adjust to changes of the network. There are multiple existing allocation strategies, e.g., [5, 6, 7, 8], which we will investigate to develop a suitable solution. As the response time of an application is not only determined by computation time, but also by network delay, we will develop mechanisms to transmit high-level application requirements [10] to the network management solutions, developed in Task 4.3, in order to adapt to the (executed/planned) applications. Hence, our planning and execution system will thus be integrated with the network management, and the network will be adjusted based on (changing) IoT application demands. Thereby, these two sides of the infrastructure management will act in a closed loop: (1) An optimized deployment to IoT/Edge Devices is computed in-line with network characteristics and a network configuration (network slices, bandwidth reservation, etc.) is determined. (2) The network dynamically reconfigures itself in response to deployed applications (as part of Obj. 2<sup>2</sup>). The updated deliverable, D4.5, will then report on these developments regarding the efficient and dynamic management of the computing infrastructure.

---

<sup>2</sup> Objective 2: Enable ultra-reliable low-latency communication over heterogeneous networks to enable tactile (real-time) and contextual (adaptive) interaction between IoT devices, humans, and services.

## 7 Bibliography

- [1] Siemens, "Industrial Edge: Bring IT to the field level – easily, flexibly, and securely," 2019. [Online]. Available: <https://new.siemens.com/global/en/products/automation/topic-areas/industrial-edge.html>. [Accessed 29 11 2019].
- [2] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, pp. 2961-2991, 2018.
- [3] docker-compose. [Online]. Available: <https://docs.docker.com/compose/>.
- [4] nginx, "<https://www.nginx.com/>," [Online].
- [5] A. M. Haubenwaller and K. Vandikas, "Computations on the edge in the internet of things," *Procedia Computer Science*, vol. 52, pp. 29-34, 2015.
- [6] S. Sardellitti, G. Scutari and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, pp. 89-103, 2015.
- [7] N. Mohan and J. Kangasharju, "Edge-Fog cloud: A distributed cloud for Internet of Things computations," in *2016 Cloudification of the Internet of Things (CloT)*, 2016.
- [8] V. Cardellini, V. Grassi, F. Lo Presti and M. Nardelli, "Optimal operator placement for distributed stream processing applications," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, 2016.
- [9] J. Seeger, A. Bröring and G. Carle, "Optimally Self-Healing IoT Choreographies," *ACM Transactions on Internet Technology (TOIT)*; *arXiv preprint arXiv:1907.04611*, 2019; to be published.
- [10] J. Seeger, A. Bröring, M.-O. Pahl and E. Sakic, "Rule-Based Translation of Application-Level QoS Constraints into SDN Configurations for the IoT," in *EuCNC 2019, Valencia, Spain*.