DETERMINISTIC6G

Report on Dependable Service Design

D1.3

The DETERMINISTIC6G project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no 1010965604.

Report on Dependable Service Design

Grant agreement number:	101096504
Project title:	Deterministic E2E communication with 6G
Project acronym:	DETERMINISTIC6G
Project website:	Deterministic6g.eu
Programme:	EU JU SNS Phase 1
Deliverable type:	Report
Deliverable reference number:	D1.3
Contributing workpackages:	WP1
Dissemination level:	PUBLIC
Due date:	M24
Actual submission date:	30-01-2025
Responsible organization:	B&R-AT
Editor(s):	Oliver Höftberger
Version number:	V1.0
Status:	Final
Short abstract:	This report introduces a value-driven approach to service design with focus on 6G networks, aligning technical capabilities with industrial needs and societal priorities such as productivity, sustainability, and worker safety. It defines a framework for adaptive and dependable services and introduces concepts like modes and levels of operation to enable dynamic service adaptation, ensuring optimized performance and reliability for critical applications. Practical use cases, including XR, exoskeletons, adaptive manufacturing, and smart farming, demonstrate how the proposed framework maximizes value for industries and communities alike.
Keywords:	service description, dependable services, modes of operation, levels of operation, service modelling, application value, service request, service optimization, dependable communication, dependable computation, time synchronization service, cybersecurity service, 6G, time-sensitive networking (TSN), adaptive applications, industrial use cases

Contributor(s):	Joachim Sachs (EDD) Edgardo Montes de Oca (MI) Huu-Nghia Nguyen (MI) Mahin Ahmed (SAL) Jose Costa-Requena (CMC)

	János Harmatos (ETH) Dávid Jocha (ETH) Leefke Grosjean (EAB) Christer Holmberg (LMF) Inés Álvarez (ABB) Francesco Giovacchini (IUVO) Filippo Dell'Agnello (IUVO) Emilio Trigili (SSSA)

Reviewers:	Frank Dürr (USTUTT) Saeid Karamzadeh (SAL) Norbert Reider (ETH)

Revision History

V0.1	First draft
V0.2	Draft for review
V1.0	Final version

Disclaimer

Executive summary

This report outlines a comprehensive framework for dependable and adaptive service design in 6G networks. It presents a value-driven approach that aligns technological innovations with industrial and societal priorities such as sustainability, productivity, and worker safety. By addressing the inherent challenges of 6G's dynamic and stochastic environments, the report establishes foundational principles for designing robust services that meet the stringent demands of next-generation applications.

The report introduces key concepts, including modes of operation (driven by application needs) and levels of operation (infrastructure-driven changes), to enable dynamic service adaptation as enabler for dependable applications. These mechanisms ensure optimized performance and reliability across diverse use cases. Core technical contributions include dependable modeling of communication, computation, time synchronization, and cybersecurity services, along with concepts for their integration into 6G mobile networks.

Practical use cases demonstrate the applicability of this framework. Examples such as exoskeleton and extended reality-based industrial worker support, adaptive manufacturing systems, and smart farming illustrate how dependable services can enhance productivity, reduce waste, and support societal well-being, while economic value provision can be optimized at the same time. These scenarios underscore the focus of DETERMINISTIC6G on maximizing value through value-driven adaptivity. This document offers a structured methodology to enable reliable, adaptive, and efficient 6G systems and wired industrial TSN networks. By integrating dependability principles with adaptive capabilities, it contributes to advancing Industry 5.0's goals of inclusivity, sustainability, and human-centric innovation.

Contents

1 Introduction

The economic and social prosperity of modern society increasingly depends on technological advancements driven by digitalization. Innovations such as extended reality (XR), digital twins, and cyber-physical systems (CPSs) are revolutionizing interactions between people, machines, and autonomous systems. These technologies enable new possibilities for productivity, sustainability, and inclusivity while reshaping industries and societal functions. The next generation of mobile communication, 6G, is positioned as a key enabler to support these advancements by bridging digital and physical worlds in mobile and dynamic scenarios. Cloud and edge computing further amplify the benefits by providing scalable and efficient platforms for computation and data management. Together, these advancements form the foundation of Industry 5.0 [LSW+22], which prioritizes human inclusion, system dependability, and environmental sustainability.

The societal value of these advancements lies in their ability to address critical needs: enhancing worker safety, optimizing resource utilization, reducing waste, and enabling new forms of collaboration between humans and machines. For instance, adaptive manufacturing minimizes material surplus and environmental impact, XR enhances workforce training and productivity, and autonomous systems improve safety and operational efficiency in industries like farming and logistics. However, realizing these societal benefits hinges on the ability to deliver dependable and adaptable services that can meet the stringent demands of future applications.

Emerging applications in domains such as adaptive manufacturing, industrial automation, and XR-driven interfaces require ultra-low latency, high availability, and deterministic performance [DET23-D11]. These time-critical services are characterized by their dependence on reliable communication and computation across highly dynamic and interconnected environments. Traditional systems, which are designed for static, localized use, are inadequate for these scenarios. The stochastic nature of wireless communication, variability in computational resources, and the need for real-time (RT) adaptation present significant challenges, particularly for safety-critical and business-critical operations.

To overcome these challenges and unlock the societal value of future applications, a paradigm shift is required in how applications and services are designed and delivered. Future systems must integrate deterministic and adaptive capabilities, ensuring dependable operation while accommodating inherent variabilities in communication and computation. Dependable service provisioning plays a central role in achieving these objectives by enabling consistent performance and reliability even under dynamic conditions. This also underpins applications that improve sustainability, enhance worker well-being, and enable inclusive growth. By addressing these challenges, it is ensured that technological advancements not just translate into economic growth, but also into meaningful societal benefits, paving the way for a more prosperous and equitable future.

1.1 DETERMINISTIC6G Approach

Digital transformation of industries and society is resulting in the emergence of a larger family of time-critical services with unique requirements distinct from traditional Internet applications like video streaming or web browsing. These services demand stringent guarantees, such as ultra-low latency,

high availability, and seamless operation across diverse domains, including industrial automation, extended reality, and autonomous systems.

Traditional time-critical communication systems, like those employing programmable logic controllers (PLCs) and wired protocols such as Powerlink and EtherCAT, have been limited to static and isolated configurations tailored for specific local applications. Recent advancements in standardization, such as Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet), have extended deterministic capabilities to Ethernet and IP networks. These advancements have enabled managed infrastructures with consistent performance, zero packet loss, and guaranteed low-latency communication. However, assumptions underlying these traditional systems — such as tightly bounded timing behavior and minimal stochastic variability — are increasingly challenged by the adoption of wireless technologies, cloud computing, and dynamic application demands.

DETERMINISTIC6G represents a revolutionary approach to time-critical communication networks. It integrates stochastic and dynamic elements into the deterministic paradigm, ensuring dependable performance despite inherent uncertainties. The objective of DETERMINISTIC6G is to design, plan, and operate scalable, multi-domain infrastructures capable of supporting diverse time-critical services while addressing the following challenges:

1. **Integration of Stochastic Elements:** DETERMINISTIC6G embraces the stochastic behavior of wireless links and computational resources by modeling them through envelopes of variability (short-term or long-term). Monitoring and prediction mechanisms are employed to quantify key performance indicators (KPIs) such as latency and reliability, enabling deterministic-like planning even with stochastic variances.
2. **Management of End-to-End Interaction Loops:** The approach extends deterministic control to encompass the entire end-to-end (E2E) loop (e.g., from sensor to controller to actuator), including interactions with computational elements. This ensures comprehensive management of time-critical operations despite the dynamic nature of the underlying network and compute elements.
3. **Adaptation of Applications:** Recognizing the inevitability of performance degradation due to stochastic factors, DETERMINISTIC6G enables applications to dynamically adjust their end-to-end KPI requirements at runtime. This includes harmonizing application demands with network conditions, allowing for scalable and resilient operation under variable circumstances.

DETERMINISTIC6G also emphasizes time-awareness by ensuring precise synchronization across all network components. Furthermore, it incorporates security-by-design principles to safeguard dependable communication, essential for applications where high reliability and safety are paramount.

To achieve its objectives, DETERMINISTIC6G utilizes novel methodologies and technologies, including algorithms for dynamic E2E schedule optimization that ensure robustness and scalability under varying conditions. Through these innovations, DETERMINISTIC6G facilitates the deployment of converged and scalable network infrastructures that meet the demands of future 6G applications and beyond, enabling dependable time-critical communication across diverse and dynamic domains.

1.2 Objective of the Document

This report aims to detail the service design and service description frameworks developed within the project, focusing on achieving high dependability of service provision and automatic optimization of system performance to maximize the value provided to the different stakeholders. By addressing the unique requirements of industrial applications, the report establishes a foundation for reliable, adaptive, and efficient 6G and wired TSN systems.

The report advances deterministic service design by leveraging the OPC Unified Architecture (OPC UA) Field eXchange (FX) model of automation components to generalize industrial use cases and describe dependable services. This framework enables the abstraction of assets, functions, and connections, allowing seamless integration of critical infrastructure services such as communication, computation, time synchronization, and security. By employing this unified approach, the report provides a scalable methodology to address the diverse requirements of industrial applications.

The report emphasizes the value-driven approach to service design, linking technical specifications to tangible industrial and societal benefits. Key societal value indicators (KVIs), such as environmental sustainability, worker well-being, and inclusivity, are integrated into the service value descriptions. For example, adaptive manufacturing reduces waste, while XR and exoskeleton applications enhance worker productivity and safety. These value descriptions align industrial priorities with broader societal objectives.

To ensure adaptivity and dependability, the report introduces distinct concepts of modes of operation (application-driven changes) and levels of operation (infrastructure-driven changes) as supported by the corresponding applications. This dual framework enables dynamic adaptation of services, optimizing resource allocation while maintaining reliability for critical applications. Moreover, the report explores how prioritized services ensure dependability for high-value applications, safeguarding mission critical applications under changing conditions. To this end, a framework of co-design between applications, the communication system, and a compute platform, which is executing the applications, is presented. There are mutual dependencies among those domains for the provisioning of the application service and by sharing insights among those domains an improved service performance and better resource utilization can be achieved.

Finally, the report illustrates its findings through practical industrial use cases, including XR-enhanced manufacturing, exoskeletons in the industrial context, adaptive production lines, and smart farming. These scenarios demonstrate the applicability of the proposed methodologies, highlighting how service design can optimize performance and dependability while addressing societal needs in real-world industrial settings.

1.3 Relation to other Work Packages and Deliverables

This document is part of work package (WP) 1 and establishes a foundation for other technical work packages within the DETERMINISTIC6G project. It builds upon the use cases outlined in deliverable D1.1 [DET23-D11], using them as input to define a generalized use case model. These use cases also describe applications with specific dependability requirements regarding the services they rely on.

Figure 1.1 illustrates the relation between this and other WPs within the DETERMINISTIC6G project. This document incorporates the findings from WP2 and WP3 to describe concepts and practices that

foster the dependability of the described services, including communication, computation, time synchronization, and security.

The findings of this document concerning service description will be used in WP2 and WP3 to define mappings that transform application's service requirements – functional and non-functional – into network capabilities and resource allocation strategies. Furthermore, the service descriptions and service requests from this document will be fed into WP4 to evaluate the network capability mappings and the resource allocation algorithms.

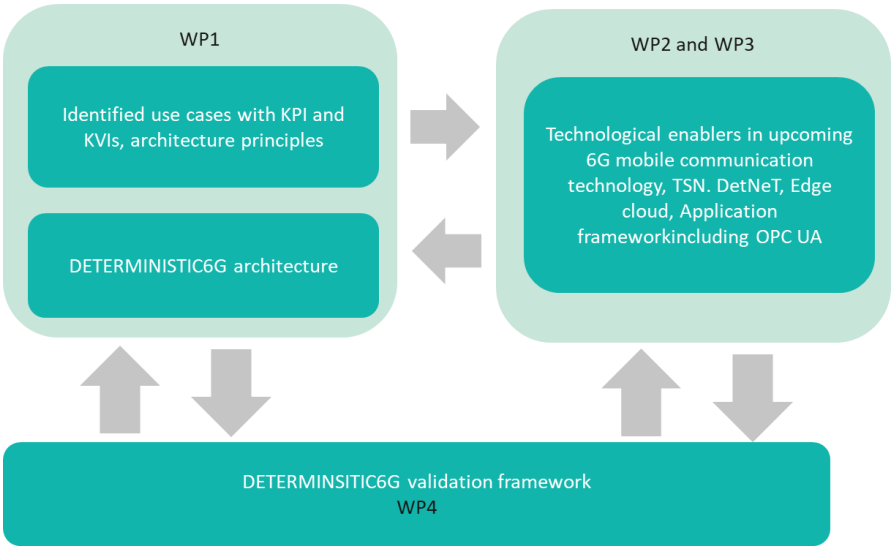

Figure 1.1: Relation between work packages in DETERMINISTIC6G

In summary, this document not only serves as a pivotal input for WP2 and WP3 but also provides crucial insights for the validation framework in WP4, thereby enhancing the overall determinism and dependability of the 6G E2E communication architecture.

1.4 Structure and Scope of the Document

After this introduction, the document continues with an overview of industrial use cases, such as extended reality and exoskeletons in the industrial context, adaptive manufacturing, and smart farming, which illustrate the importance of service adaptivity and dependability. A review of related work in Section 3 highlights existing gaps addressed by the proposed methodologies, while the subsequent introduction of a generalized use case model in Section 4 provides a unified framework for abstracting the application structures and service dependencies from the use cases.

The core sections of the document delve into dependable service design, examining how service descriptions enable adaptive applications and discussing key concepts like value and cost functions for application prioritization in Section 5. In Section 5.3, the definition of application services, and the modelling and known key concepts of the dependable platform subservices in the focus of DETERMINISTIC6G, as well as their integration into higher-level applications – as constitutional part of

the use cases – is explored. Alongside, in Section 7, mechanisms for dynamic adaptation of application services and dependable platform services, to maintain system performance in a dynamically changing environment, are presented. Architectural cybersecurity aspects to safeguard dependable services and system integrity are addressed in Section 8. In Section 9, the guiding use cases are revisited, demonstrating the practical application of the core service description concepts introduced in this document. Finally, the document concludes in Section 10 by summarizing the role of service descriptions in achieving adaptable and reliable 6G networks.

2 Use Case Overview

This section illustrates how a proper service description is key to support the future industrial applications targeted by the DETERMINISTIC6G project, by delving into specific scenarios of the use cases described in Deliverable D1.1 [DET23-D11]. In particular, this section describes one scenario per use case which illustrate how the service delivered by the application, and by the DETERMINISTIC6G system itself, can change during runtime to provide value to the stakeholders (i.e., end users of an application, the owners of the DETERMINISTIC6G system setup, and in some cases also the society as a whole). In this context, an ***application*** can be any combination of software (i.e., programs) and hardware (i.e., controller devices, sensors, actuators) that leverages the capabilities of the DETERMINISTIC6G system to provide a defined service (i.e., the obtained and intended added value) to the user of the application. The DETERMINISTIC6G ***system*** (or also simply referred to in this document as system) is the infrastructure on which the applications are running. It provides different types of services to the applications, like wireless and/or wired communication, computation, time synchronization or security. In case an application depends on the reliable provision of such service, it is also referred to as ***dependable service***. Similarly, if the user of an application relies on the provision of the intended application service, it is denoted as ***dependable application***.

The selected use case scenarios highlight the need for adequate service descriptions at every level of the DETERMINISTIC6G architecture to maintain the intended operation of the application and the system in a dependable and safe manner throughout the whole system operation, even in the presence of changes. As further detailed in Section 7.1, such changes might be intentional and anticipated adaptations of the application or reactions to changing environmental conditions. The ability of the system to perform these adaptations will be referred to as ***adaptivity***. Furthermore, the scenarios also aim at highlighting the importance of the value provided by the application and by the DETERMINISTIC6G architecture, as providing added value to different stakeholders is the ultimate goal of any application and system infrastructure.

The following descriptions start with a brief summary of the use cases from the Deliverable D1.1 and the corresponding value provided by each of them, followed by a more concrete scenario that illustrates possible modes of operation and corresponding levels of operation. In this document, a ***mode of operation*** of an application is defined as *a specific configuration and/or state in which the application operates to achieve its intended performance and reliability objectives.* As described later in this document, the mode of operation has a significant impact on the service (i.e., the value) that is provided by the application to its stakeholders, and in many cases, the provision of the service also requires the execution of a sequence of modes of operation. This may also mean that different types of services are provided in the distinguished modes. For example, an autonomous transport vehicle

might distinguish the modes *loading*, *driving* and *battery charging*. In this example, one could consider the safe transport of objects as a type of service that is provided.

In contrast, ***levels of operation*** are referred to as *the distinct hierarchical stages or tiers at which an application functions to meet specific performance, reliability, and quality of service (QoS) requirements. Each level of operation encompasses a unique set of operational parameters and behaviors tailored to ensure the application's dependability.* Like with modes of operation, the provided service of the application differs at the distinguished levels. The levels of operation are intended to define multiple performance levels for a given type of service, or to focus on a specific aspect of the provided service (e.g., the safety aspect as mentioned above). For instance, in the *driving* mode of operation of the autonomous vehicle, the levels of operation *fast driving*, *slow driving* or *safe stop* may be distinguished.

Within the service definition of this document, modes of operation are controlled by the application (or the user) itself, while the active levels of operation are selected by the DETERMINISTIC6G infrastructure by changing the platform services (i.e., communication, computation, time synchronization services, or security services) that are provided towards the running applications, and the resulting reaction of the applications. Finally, the presented use cases also show the relationship between changes in the mode of operation and the level of operation, that is, the use cases reflect how a change in the mode of operation of an application may imply changes in the level of operation supported by the infrastructure; and vice versa, how changes in the level of operation of the infrastructure may have an impact on the mode of operation of the application and, ultimately, in the value provided by it. More details on the concepts of modes of operation, as well as the levels of operation will be discussed in Sections 6.1 and 7.2.

2.1 Extended Reality (XR)

2.1.1 Use Case Description

Professional workers in the industrial domain face the challenge of an ever-evolving factory environment with new tools and machines being introduced, and processes becoming more and more complex. This trend is expected to continue. XR is expected to be instrumental in supporting professional workers in the industrial domain in the area of learning, upskilling, and efficient execution of tasks (see Figure 2.1). For instance, the possibility to have information overlaid on the real world while simultaneously having your hands free has been shown to increase worker efficiency dramatically. Furthermore, XR provides possibilities for human-centric design, meeting the industrial worker at his or her individual development level, overlaying/showing only the content that is relevant for the individual worker.

Figure 2.1: An industrial worker accessing shopfloor information via XR (source: [GMH+23])

In this section, XR use cases in the industrial domain are explored via different scenarios that highlight the need for dependable service descriptions and service adaptivity.

Scenario description:

- **Context:** When setting up or optimizing a new production line involving a number of machines, assets, etc., it needs to be carefully checked that the production process is working as designed and if necessary, adjustments need to be made. For this task, a human worker is instructed to monitor and if necessary, perform smaller adjustments to the process where needed. By enhancing the worker with XR devices (like e.g. XR glasses) and collecting and providing relevant information from the digital twin of the production line, the task can be done more efficiently. For example, via the XR device the worker can receive input about e.g., parameters of the machines, output rate of the process, or adjustment suggestions. Moreover, via an interface to the machine displayed on the XR devices the worker may also set parameters of involved machines, change positions, etc.
- **Value discussion:** The ambition is to perform all adjustments while the industrial process is already up and running to always ensure the highest currently possible production output. Depending on the adjustments to be made, the worker may have to intervene/steer the process leading to a lower production rate. A well-adjusted production line is in the longer run expected to produce less faulty products, reduce potential re-work time, and increase the quality of the products. Moreover, with the information being made available to a human worker, the training times for human workers decrease, which is even more important as production system develop towards higher flexibility, where the production system is constantly changing. Besides benefits in reduced training times, XR-supported upskilling and re-skilling may also provide access to a larger work force.

Sustainability angles in this scenario are the following:

- Reduced scrap and reduced discarded low-quality output contribute to environmental sustainability. Prioritizing advanced real-time fault search at times may contribute to this.
- To ensure the well-being of the worker, the system may want to ensure that the worker is not over-using and getting stressed by high-speed data processing, e.g., during real-time fault search at normal process speed. See also [GMH+23].

The value of the industrial service “process monitoring/adjustments via XR” depend on factors like acceptable quality, acceptable production output, acceptable environmental sustainability impact, acceptable worker well-being impact, etc. With increased adaptivity implemented in the system, the factory owner can steer and optimize the value of the industrial service by adjusting priorities of the parameters according to his/her needs at hand. The choice has a direct impact on the need for communication and compute resources of all devices involved in the system.

Figure 2.2 shows the technical realization of the use case. In order to reduce the weight, battery consumption and heat in the head-mounted XR display, compute-intensive functions are offloaded from the XR device to a spatial compute function in the edge cloud, as described in Deliverable D1.1 [DET23-D11]. In the edge cloud, also the spatial compute is located, where virtual objects are merged into the local vision of the XR headset. Depending on the application, the information that is merged into the local vision can be process or machine information (received from the factory digital twin) for process analysis, maintenance, and repair; it can also be information received from a remote expert or a co-worker. After the information has been merged in the spatial compute function in the edge cloud, it is rendered and transferred to the XR headset, where it is visualized.

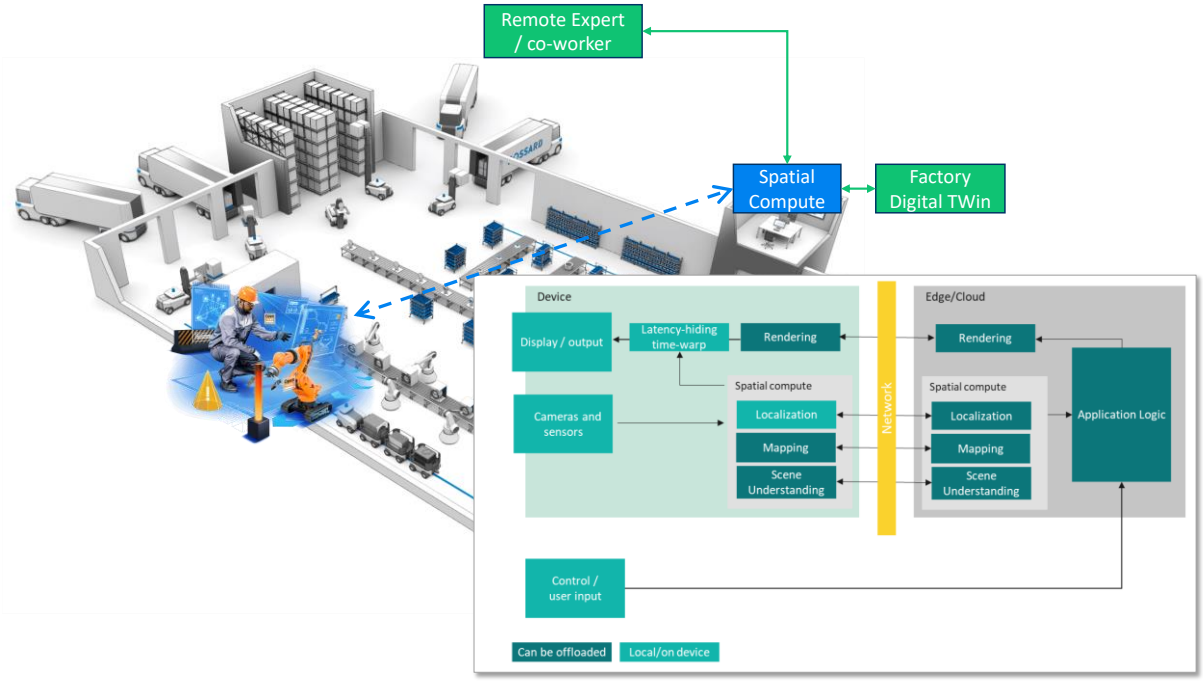

Figure 2.2: XR use case for an industrial worker

2.1.2 Levels and Modes of Operation

In the above scenario, the XR device will have to adjust the content and the way information is shown in response to different factors, such as the type of operation being performed by the worker, the context the worker is in, individual information relevant for ways to execute a task, etc.

The below *modes of operation* further highlight the need/potential for adaptivity.

- During normal production operation, the worker is observing the industrial process, receiving high-level information from the digital twin run in the edge cloud about machines and parameters that are in his/her field of vision. Communication resources need to be available for the industrial process and the XR device at the specific location of the human worker.
- During real-time fault search, the worker may want to retrieve and compare information about parts of the process (zoom-in) in real-time. Real-time fault search can be selected during normal operation and is triggered by the worker, or the digital twin related to the industrial process. This mode has more stringent requirements towards communication and compute from the XR device side, e.g. more communication resources need to be available for the XR device at the specific location of the human worker as well as devices in the field of view of the worker. During advanced real-time fault-search, the worker may want to slow down the process and run it at e.g. half the normal speed to understand where the erroneous behavior is happening. In this setting, the information displayed on the device may be much more detailed than during normal mode, potentially using advanced AI algorithms in the cloud to support for fault search. This mode has more stringent requirements towards communication and compute from the XR device side but potentially less stringent requirements towards the connected devices/machines related to the industrial process. This needs to be reflected in the service definition to request communication and compute services.

As outlined, the different modes the industrial worker can utilize depend on the availability of communication and compute resources. In the case resources are limited, depending on the mode of operation, the system may make use of increased adaptivity so that for instance resources may be switched from belonging mainly to the industrial process, to being consumed to a larger extent by the XR device instead at times. In all the modes of operation of the XR use case, there are some levels of operation, which define the quality of the XR experience. By allowing XR to operate at a lower resolution and/or lower frame rate, the required communication and compute resources can be reduced at the price of some quality reduction of the XR perception. This may be acceptable for the operation of the use case, but it may also lead to some slowdown of the use case, where the tasks are performed at a lower pace.

2.2 Exoskeleton in Industrial Context

2.2.1 Use Case Description

Occupational exoskeletons are proving to be effective tools that companies are adopting to help reduce the physical strain on workers involved in physically demanding tasks. This use case envisions the deployment of exoskeletons in industrial settings, relying on future 6G networks to enhance the synergy between exoskeletons and human workers. Next, some relevant points in the use case are summarized starting from an example of application, followed by a proper dependable service description capable of supporting the generalized use case societal values. A practical example, where

a worker wears an exoskeleton that supports moving, lifting and handling loads in the warehouse, helping to mitigate the risk of injury during demanding tasks, is shown in Figure 2.3. In-depth details regarding this use case which involves a lumbar active occupational exoskeleton (OE) are described in Deliverable D1.1 [DET23-D11].

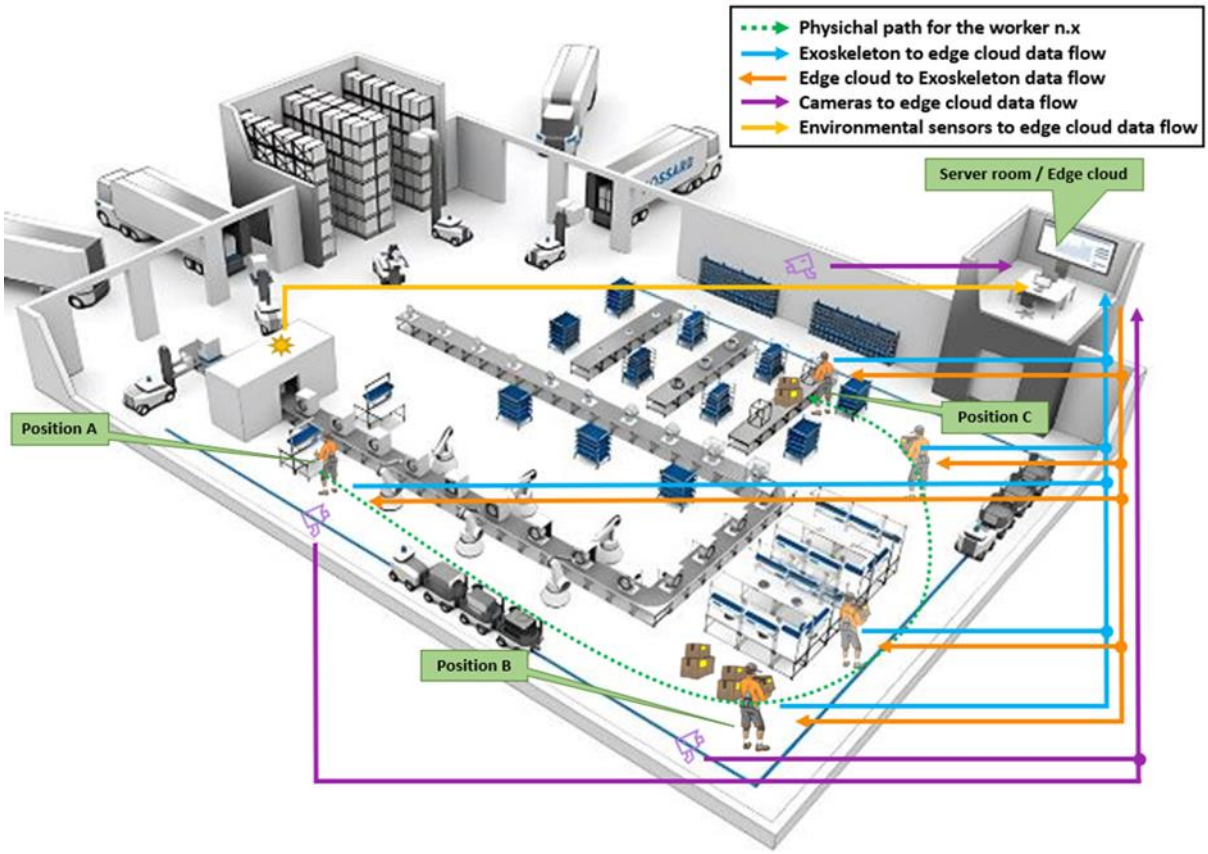

Figure 2.3: Example of exoskeleton use case in a shopfloor to assist walking and lifting tasks of a worker. In the example depicted the generic n.x worker monitors the automatic assembly line at position A, then walks to position B and lifts a small box to complete assembly at position C. The green dashed line represents the physical path for the worker while the straight lines highlight the data flow between the edge cloud, environmental sensors, cameras, and exoskeleton.

Scenario Description:

- **Context:** In this use case, exoskeletons will adjust their behavior in response to various factors, such as the user's condition, the type of task being performed, and other context-specific elements. These exoskeletons will be part of a connected network, exchanging data with a centralized edge cloud system and interacting with other modules in the industrial ecosystem such as cameras and environmental sensors.
- **Value discussion:** The use case aims to boost the maturity of active exoskeletons by leveraging the potential of the 6G network to offload the control of the wearable robots, as well as the hardware components on-board the exoskeleton, thus reducing power consumption. A delocalized dependable control architecture enables computation of complex artificial intelligence-based assistive strategies that consider a huge amount of kinesthetic, physiological, and environmental information and to improve the overall benefit and safety of exoskeleton devices. Exoskeletons worn by industrial workers to enhance strength and

safety demand accurate time synchronization. For preserving the safety of the user, time synchronization redundancy is desirable. At the same time the latency and jitter in the communication have to comply with the overall implemented control architecture to guarantee the required performances for a wearable assistive device.

2.2.2 Levels and Modes of Operation

As envisioned by this use case, the worker using the lumbar exoskeleton both (i) receives support while lifting loads, to reduce the risk of injuries, and (ii) receives assistance while walking, to reduce the effort of carrying loads when moving in the warehouse. This makes it clear that different *modes of operation* can be identified:

- (MO1) the worker walks in the warehouse, aided by the assistance provided by the exoskeleton;
- (MO2) the worker either stops walking to lift/leave down a load or starts walking after lifting/leaving down a load;
- (MO3) the worker lifts/leaves down a load.

During each of these modes of operation, the exoskeleton sends the data acquired by its on-board sensors to the edge cloud (light blue arrows in Figure 2.3). The edge cloud elaborates these data, together with the data gathered by cameras and other environmental sensors (purple and yellow arrows in Figure 2.3), computes the value of the support that the exoskeleton must provide based on the task the worker is accomplishing, and transmits this information back to the exoskeleton (orange arrows in Figure 2.3).

In MO2, the frequency of data transmission, both from the exoskeleton and environmental sensors to the edge cloud and from the edge cloud back to the exoskeleton, should be higher than in the other two modes of operation, because during transitions between different actions, the movements performed by the worker are neither periodic, as walking, nor “stereotypical” as lifting/leaving loads down. The increase in data transmission frequency of course would imply a change in the *level of operation* of the 6G network, specifically an increase in the required 6G network bandwidth.

However, since in all the described modes of operation the worker wearing the exoskeleton is in the loop and the support provided by the exoskeleton depends on his/her actions, and being human movements characterized by unpredictability, it is crucial that the most stringent requirements, i.e. those of MO2, are maintained also during MO1 and MO3. This is fundamental to preserve the safety of the worker and ensure that the exoskeleton adapts its assistance whenever the worker unexpectedly changes the task, he/she is accomplishing or the way he/she is doing it.

Each of the three described modes of operation can be executed with two different *levels of operation* according to the current network performance, namely the Standard Level and the Safe Level, described in the following.

- *Standard Level*: this level of operation represents the default and standard operation of the use case, executed when the infrastructure services performance is optimal. In this level, the full set of sensors is available and utilized, both embedded and external, to gather information about the user and the surrounding environment. The control system optimizes the interaction torque and the target assistance level based on these diverse inputs, providing adaptivity and responsiveness tailored to the user's needs.

- *Safe Level*: this level of operation is designed to always ensure user safety and maintain basic functionality in situations where the infrastructure services performance is partially compromised. In this level, the exoskeleton relies primarily on the highest-priority data (i.e., the encoder readings from the embedded sensors used to guarantee a safe operation measuring the interaction torques with the user). Other non-essential information and external sensor inputs are temporarily suspended. The exoskeleton operates in a transparent manner (i.e., not providing assistance), zeroing the interaction torque with the user to prevent any inadvertent movements or actions. This level ensures a safe and reliable operation until infrastructure services full performance is restored.

In order to guarantee an acceptable user experience, the infrastructure should have such adaptivity as to make the switch between the two levels effortless and efficient without any noticeable gaps, delays, or inconsistencies in the functioning of the device. In the event of network shutdown and when infrastructure services are no longer available, a protection level is implemented directly in the exoskeleton logic to mitigate the overall risk for the user.

2.3 Factory Automation: Adaptive Manufacturing

2.3.1 Use Case Description

Deliverable [DET23-D11] describes in detail a manufacturing use case, which has adaptivity at the core of its design principles. Adaptive manufacturing systems can lead to improved sustainability, as adaptivity allows to respond to the market needs in a faster and more efficient manner, allowing to change the product that is being produced depending on the need. Furthermore, it can also support customization, prioritizing customers' needs over the production of large batches of standard products. These are just two examples on how adaptive manufacturing can result in a reduction of surplus products, and therefore a reduction in waste. But another way in which adaptive manufacturing can improve sustainability (as well as productivity) is by reducing the number of equipment deployed in manufacturing sites. This can be achieved by introducing mobile processing modules (MPMs) that can be used in different parts of the factory depending on the production needs.

2.3.2 Levels and Modes of Operation

In order to support customization of products, production lines (PLs) need to support several *modes of operation*, meaning that they need to be able to change the way they operate. In this scenario, we assume we have two different modes of operation:

- MO1: the production line builds a simple product, with no extras.
- MO2: the production line builds the same product as in MO1, but now with an extra feature that requires one (or several) new MPM(s) with different tools to be built in.

An example of this could be a PL for mobile phones, in which each mobile phone model has several versions, and each version has more added features than the previous, e.g., an extra camera. In order to add the new feature, not only we need new MPM(s) with extra tools, but also the already existing processing modules need to modify their operation, for example, change the placement of the components they embed in the phone to make space to add the extra camera. Therefore, the whole production line needs to be adapted, which implies a change in the mode of operation.

Figure 2.4 depicts an example of a production line that is changing its mode of operation from MO1 to MO2. Specifically, we see a PL composed by two static processing modules and one MPM (MPM1), which collaborate to work in MO1. Note that the processing modules exchange information constantly during the building process, and this information is represented with the orange solid line. We can also see a second MPM (MPM2) that is approaching the PL in order to change to MO2. MPM2 needs to exchange registration information with the PL to join the line, and this information is represented with the blue dashed lines.

Furthermore, in this scenario, MPM1 and MPM2 are different, as they have different tools but also, they have different capabilities. Specifically, we assume that MPM2 is an older model that does not support all the functionalities MPM1 does, for example, highly accurate synchronization. We reflect this difference in the figure by coloring MPM2 in gray. All these changes in the mode of operation of the PL imply changes at the infrastructure level too, e.g., the number of communication streams, the number of publishers and subscribers to those streams, or the accuracy in the synchronization. This is what we call *levels of operation*.

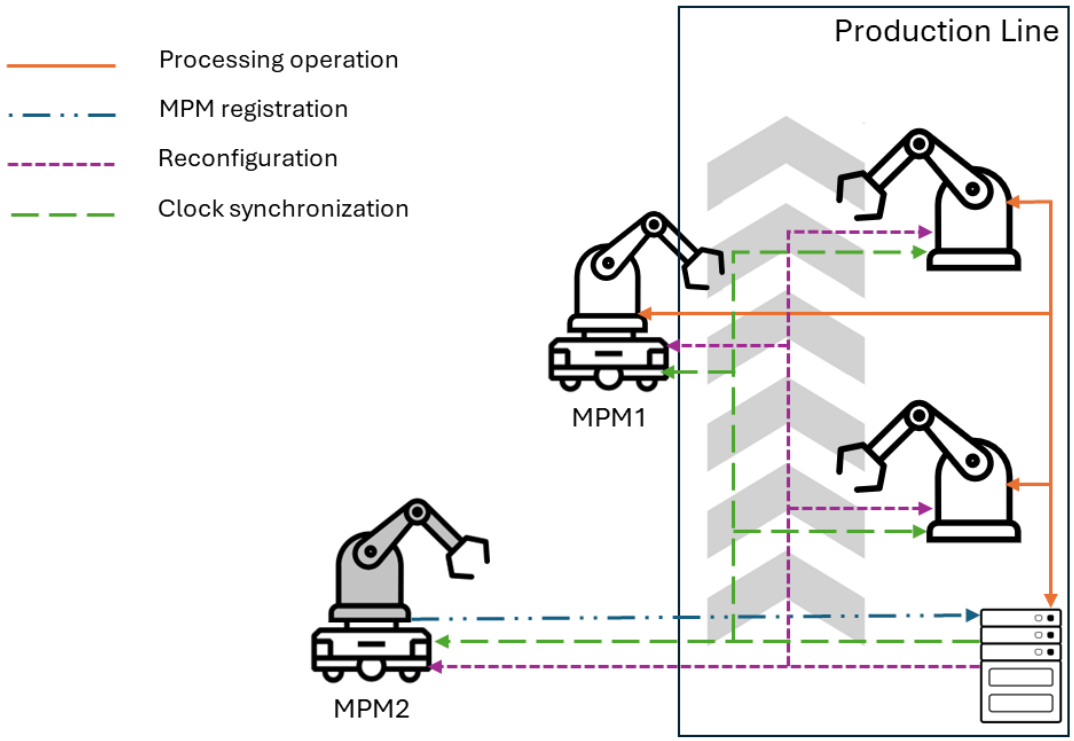

Figure 2.4: Adaptive manufacturing example of a change in the production line

To operate correctly, all the processing modules that collaborate to perform a task need to be coordinated and synchronized. When introducing MPM2 with lower synchronization accuracy capabilities, the operation of the whole production line needs to adapt to this limitation to avoid possible accidents or malfunction of the production line. For example, it may be necessary to reduce the speed at which the line moves, to avoid collisions between MPM2 and the static processing modules due to the lack of synchronization between the two modules. Thus, all processing modules need to change the level of operation and use the same synchronization scheme. Based on this example, we can identify two different levels of operation when it comes to synchronization:

- Accurate synchronization: used when all the devices in the production line support the most up to date synchronization protocol.
- Degraded synchronization: used when one or more devices in the production line lack the necessary support in hardware/software for accurate synchronization.

This is a reasonable scenario, as factories often have novel equipment deployed together with legacy one. Having proper support for changing the level of operation of the system allows to use novel features when available, increasing the productivity and efficiency, and using a degraded mode when forced to, ensuring continuous production.

2.4 Mobile Automation: Smart Farming

2.4.1 Use Case Description

As introduced in deliverable D1.1 [DET23-D11], agriculture is an essential domain to consider for guaranteeing a sustainable future with a fair access to food. Unfortunately, the climate crisis poses complicated challenges to the agriculture sector, which will need to adapt to guarantee a reliable food supply, as well as to improve the quality of life of farmers. The automatization of the farming process, together with more sustainable production practices and the protection of the environment, are key to ensure the transition to a more sustainable food production system.

In deliverable D1.1 [DET23-D11], a description of the smart farming use case has been provided. Based on that use case, this section adds more details on an example scenario, which supports the definition of a dependable service description in the rest of this document. Specifically, in this example the focus lies on highlighting the need for a proper service description in order to achieve an optimized smart farming process, while the system is capable of automatically reacting to changing conditions. Diverse optimization criteria may be considered that increase the corresponding achievable value. To demonstrate that not only economic value is driving optimization, within this scenario, automated sustainability is put in the foreground. This may be achieved by making the protection of wildlife part of the application, as well as the automated resource allocation process.

2.4.2 Levels and Modes of Operation

Figure 2.5 shows a scenario where a harvester is collecting crops, which are then moved to a trolley that moves next to the harvester in a synchronized manner. The orange solid line represents the exchange of information between the harvester and the trolley (or the tractor pulling the trolley). The figure also shows a drone which flies ahead of the harvester and takes images to monitor the status of the field. Whenever the drone takes an image, it sends it to the farming safety application that is running on the Edge. The transmission of images is represented by the blue dashed line. When the farming safety application receives an image, it processes the image to detect potential obstacles – mainly wild animals. Finally, the safety application sends safety information to the unmanned ground vehicles (UGVs), represented with the green dashed line. In this example the assumption is, that the harvester and trolley are moving at maximum speed (6 km/h), as the conditions allow it, maximizing productivity. In this scenario the drone moves at the same speed as the vehicles (6 km/h), in the same direction and slightly ahead of the harvester.

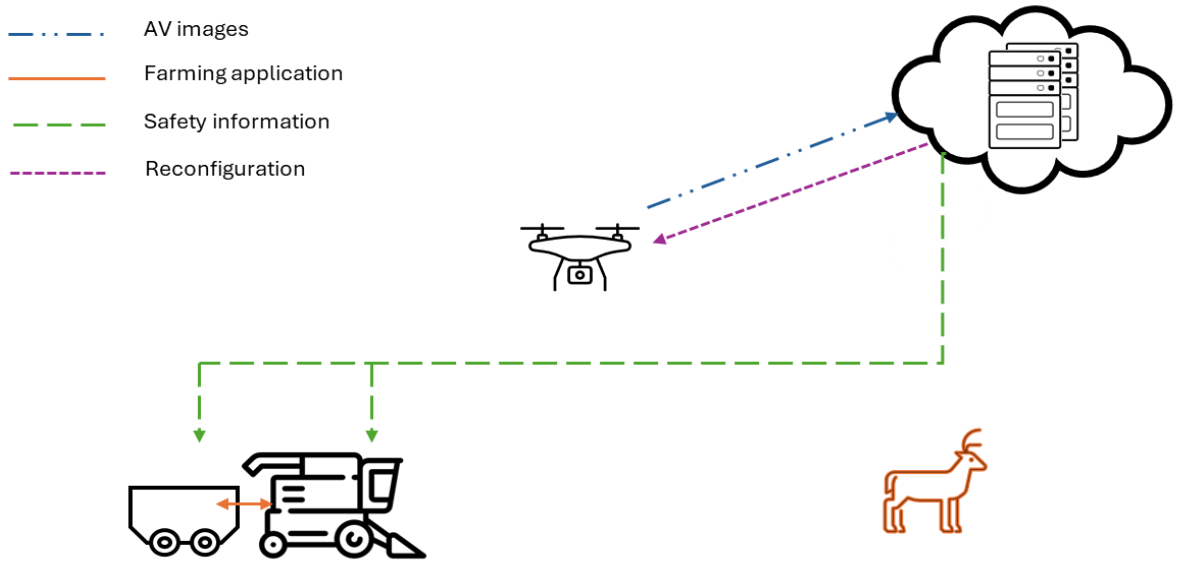

Figure 2.5: Illustration of smart farming use case

It is further assumed that a deer is running into the field. Now, when the drone captures the next image and sends it to the farming safety application on the edge, the application will detect the animal, and will send new safety information to all vehicles, indicating that they should slow down. The UGVs and the drone will slow down (3 km/h) according to the safety information, but the drone will also be configured to capture and send images more frequently, to monitor the movements of the deer. This reconfiguration information is depicted using the purple dashed line. This change of the *mode of operation* of the involved vehicles is triggered from the safety application (i.e., due to the detection of the deer), and it denotes one of the dedicated modes of operation that has been implemented for safety reasons. Another example of safety-related mode of operation would be to fully stop the UGVs if the animal does not move out of the way. From this description we can identify three modes of operation:

- MO1: regular harvesting operation.
- MO2: slow harvesting operation due to animal detection in the field.
- MO3: full stop of the harvesting operation due to continuous presence of an animal in the field. Continue supervision by the drone.

So far, we have described changes in the mode of operation from the application point of view, but these changes in the application also imply changes at the infrastructure level, which we call changes in the *level of operation*. As we have described, there is a constant exchange of information between the actors of the system (represented with the colored lines in the figure). This exchange of information takes place through the deterministic 6G network, and it requires communication resources. Furthermore, there are applications running in the vehicles as well as in the edge, which require computational resources. On top of that, to ensure that all the actors operate in a coordinated manner, we also need synchronization services. Finally, the execution of the application and the exchange of information must be secure, to avoid possible hazards introduced through cyberattacks, which could manipulate the vehicles or the images to cause a malfunction of the system.

Therefore, we need to keep in mind that when we change the way the application behaves (the mode of operation), we are potentially changing the system too (the level of operation). For example, when the drone is configured to capture and send more images per unit of time, it implies an increase in the bandwidth consumed in the 6G network, as well as an increase in the computation resources required by the safety application in the edge, which now needs to process more images per unit of time. Thus, we could differentiate these levels of operation:

- Standard level: includes the minimum amount of network, computation and synchronization resources needed to carry out the harvesting operation.
- Increased monitoring level: includes the resources from standard level, plus additional network bandwidth and computation resources to process the increased volume of traffic.
- Safety level: includes only the resources required to ensure the operation of the safety application when the harvesting operations stops.

For the system to be able to support this dynamicity and to do so in a dependable manner, it must be modelled and designed to be adaptive. In the next section we discuss previous works on modelling of future industrial systems, to later discuss the Deterministic 6G vision on the application service model.

3 Related Work

3.1 Service Architectures

The term service has been extensively used in the literature to refer to different aspects and functionalities of structures in general. A common definition of a service, which we use in this document, is that of something that serves a need [Gra23], or provides value [Kow06]. This definition can be applied at any level, from service-based business models, processes and structures; to the manufacturing infrastructure [PGK+24]. This document addresses the description of the DETERMINISTIC6G architecture as a service that provides value to the industrial applications that use it. To that aim, this document provides a model of the different services from the DETERMINISTIC6G architecture and describes how these models can be used to provide the expected value.

There are different efforts in the literature to provide models for information and communication technology (ICT) services. One example is Service-Oriented Architectures (SOAs) [OG09], which focus on the design of software architectures dividing the different functionalities into building blocks called services. SOAs are characterized by the modularity of the services, i.e., each service can be defined, developed and maintained independently of the rest, while ensuring proper interoperability through well-defined interfaces. The main limitation of a SOA is that it only concerns the software, and thus does not allow to model other relevant aspects of ICT services, such as computation or communication.

Another relevant initiative is the Topology and Orchestration for Cloud Applications (TOSCA) standard [OASIS13] developed by the OASIS Open consortium. TOSCA aims at describing cloud-based services, components and relationships, by means of templates. It also defines the process models that are used to create and terminate a service as well as to manage a service during its whole lifetime. The limitation of TOSCA is that it only allows to model the cloud applications, and not the communication services.

On the network side, we find service-level specifications (SLSs), technical specifications of service-level agreements (SLAs). An SLA is a contract between provider and consumer, which covers many aspects of their relationship, from legal and economic aspects to the service that is to be provided. The SLS is the part of the SLA that outlines which services should be provided and with which performance they should operate, as well as which techniques will be used to monitor the service. There are efforts to define SLS for industrial 5G services [5GA21a], nonetheless, these efforts do not cover the definition of interfaces between the network and the other elements of the architecture.

In this document we focus on the definition of the services and in the specification of methods, tools and frameworks that enable an industrial application to present its needs, and the underlaying infrastructure to present its capabilities in order to support dependable changes in the application and the infrastructure in real-time. This includes services such as computation, communication, time synchronization and security, and the means to enable their interoperation. One example of such effort is OPC UA FX, a standard that provides information models and protocol definitions to enable such holistic integration of services in a flexible manner. Some concepts of OPC UA FX that are relevant within this document are described in the next section.

3.2 OPC UA FX AutomationComponent Model

OPC UA [OPC-10000-1] defines a standardized information model and protocol. The information model is used to describe industrial entities, processes and functions, and the relationship between those. The protocol is used by entities that support it to communicate with each other, e.g., to register and discover entities, to exchange data and invoke method calls between entities. OPC UA is hosted by the OPC Foundation (OPCF), which coordinates the work on the base specifications. Information models for different verticals are often defined by third-party organizations, in cooperation with the OPCF.

OPC UA FX [OPC-10000-81], or UAFX, is an OPC UA extension that defines an information model and connection model for AutomationComponents. An AutomationComponent (see Figure 3.1) is an entity that performs one or more automation functions and provides connection capabilities. UAFX defines the AutomationComponentType to model AutomationComponents. An AutomationComponent can represent a device, a controller, or a function within an edge- or cloud server.

An AutomationComponent contains two main sub-models: Assets and Functions. An Asset typically describes a physical item, but can also describe non-physical assets (e.g., firmware or licenses). An example of an Asset is a sensor connected to an industrial robot. UAFX defines the FxAssetType to model Assets. A Function describes logical functionality, including functions or drivers associated with Assets (e.g., IO module functionality, motor and actuator functionality, sensor functionality), but also more complex functionalities (e.g., image recognition). UAFX defines the FunctionalEntityType to model Functions.

To model the interactions and data exchange between AutomationComponents, UAFX defines Connections. A Connection is a logical relationship between Functions, associated with different AutomationComponents. UAFX defines a ConnectionManager function, which is responsible for creating and terminating Connections. A Connection can be established between two Functions, or between multiple Functions. UAFX defines the ConnectionEndpointType and ConnectionConfigurationSetType to model the representation and configuration of a Connection

endpoint. As a Connection is a logical relationship between Functions, it is separate from the physical network used to transport the bits and data associated with the Connection.

In addition to defining the types mentioned above, UAFX also uses existing OPC UA mechanisms and information models. UAFX uses the publish/subscribe traffic pattern to exchange data. The OPC UA Publish/Subscribe information model defines types to describe the publish/subscribe specifics, including IP addresses.

UAFX classifies physical network entities. A Bridge Component interconnects two or more network segments and forwards packets between them, while an End Station represents a source or destination of network traffic. An Industrial Automation (IA) Station consists of one or more End Station Components, and optionally one or more Bridge Components. In addition, an IA Station shall support topology discovery through the Link Layer Discovery Protocol (LLDP), and may support e.g., remote management through the Network Configuration Protocol (NETCONF) and time synchronization.

An UAXF Station is an IA Station that supports UAFX. It must support priority mapping configuration and network interface representation, as defined in the OPC UA Base Network Model information model. The OPC UA Base Network information model defines types to describe network interfaces, TSN components, etc.

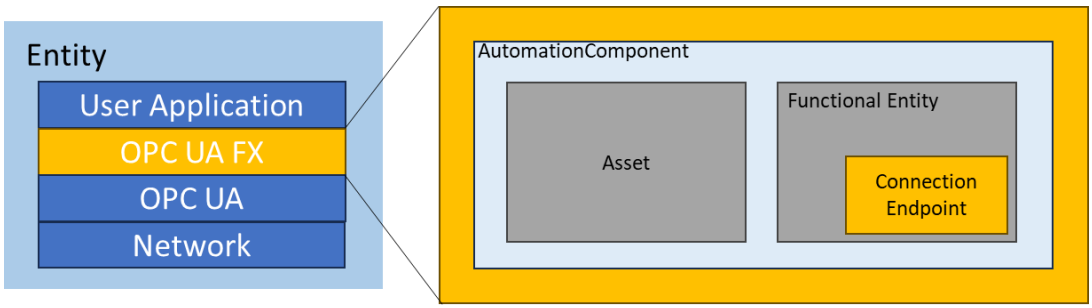

Figure 3.1: AutomationComponent model of OPC UA FX

4 Generalized Use Case Model

This section describes the model of generalizing industrial use cases within this document, and thus, provides a unified framework for the subsequent sections. While it is presented with individual references to the use cases of Section 2, it may also be applied for mapping many other use cases as well. The purpose of this generalized use case model is to abstract from the details within the use cases, their scenarios and applications. This enables to introduce and discuss the dependable service models and the concept of service adaptivity based on a cohesive structure, facilitating easier understanding of the core concepts.

4.1 Principle of Use Case Generalization

When analyzing the use cases in the focus of DETERMINISTIC6G, it can be seen that a few simple generalized concepts are sufficient to describe their essential structure, application and information flow models. Based on that, requirements on the services an application depends on can be defined. As the generalization of whole use case domains has been performed in many projects and standardization bodies before, an existing model has been chosen for this document. However, the reader is free to choose any other generalized use case model for its purposes and apply the findings about dependable services onto that model.

The model chosen in this document is based on the OPC UA FX AutomationComponent model introduced in Section 3.2. As introduced, there are different mechanisms for modelling assets, functions and connections. OPC UA is a widely popular open framework, used within a wide area of industries, that allows abstract modelling of the above, and that fits each use case described within this document. For that reason, the OPC UA FX extension for field-level communication, as specified by the corresponding OPC Foundation working groups, has been chosen to show how use cases can be modelled in an abstract way.

4.2 Dependable Service Modelling using OPC UA FX

In this section, certain aspects of the dependable subservices, which are in the focus of DETERMINISTIC6G, are mapped into the generalized use case model. An application of a certain use case may require different types of services from the 6G system or wired TSN infrastructure to operate correctly. While in some cases, these services are independent from each other, e.g., if simple best-effort communication is needed, no time synchronization may be required; in other cases, the service requested from the system may be a combination of multiple of these services. For instance, setting up a distributed application that involves computation and communication, where additionally a bounded E2E latency is important, would require a complex service that consists of computation, communication, and time synchronization subservices. In the subsequent parts of this document, a subservice is considered a service that can be specified and used independently from other subservices. However, at a higher level, combinations of such subservices may be requested by an application.

Further details about the dependability of the individual subservices will be presented in Sections 5.3 and 7.

4.2.1 Generalized Communication Modelling

For components to be able to exchange data with each other, they need to establish communication channels. Depending on the purpose of the data exchange communication channels can have different QoS parameters. associated with them.

UAFX defines Connections, which are logical communication relationships between two or more FunctionalEntities (see Figure 4.1). The UAFX Connection is separated from the physical network used to transport the data. A ConnectionEndpoint is used to model an endpoint associated with a Connection between FunctionalEntities. It provides information, e.g., regarding the input- and output data that is exchanged on the Connection. A FunctionalEntity might have multiple Connections and hence multiple ConnectionEndpoints.

Even if in UAFX data is often exchanged on Connections using the publish/subscribe traffic pattern, a more generic interpretation of ConnectionEndpoints and Connections is followed within this document. These concepts are used to describe the relevant characteristics of the dependable subservice that is required by an application.

Some types of traffic may require dedicated assets to be used or be present on the connected automation components. For instance, in order to use time-aware shapers for communication as defined by TSN standards, a reliable synchronized notion of time needs to be established, which requires the corresponding clock assets (cf. Section 4.2.2).

Further details on the communication subservice can be found in Sections 6.2.1 and 7.3.1.

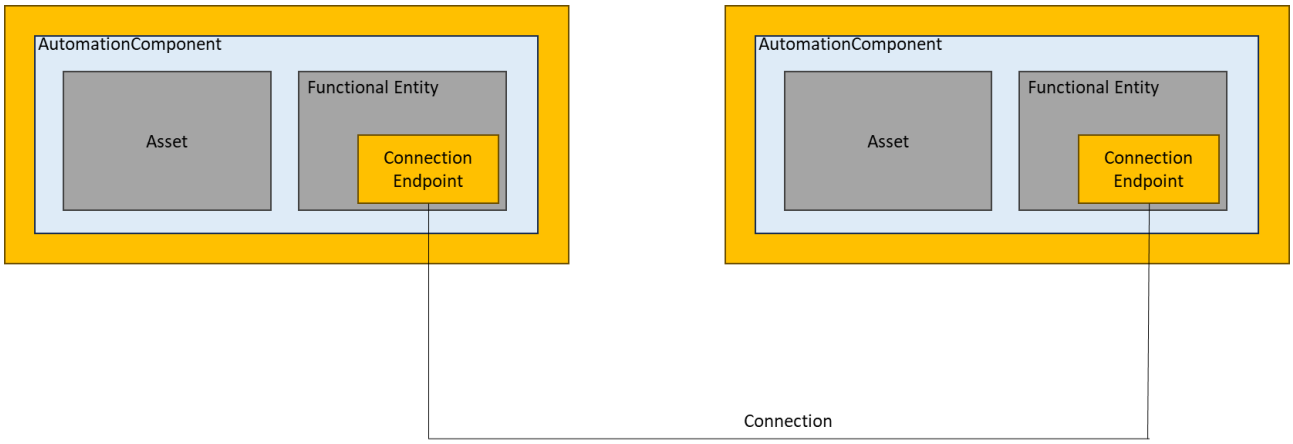

Figure 4.1: Model of connections between AutomationComponents based on OPC UA FX

4.2.2 Generalized Computation Modelling

The use cases described in this document require different types of computation subservices to realize certain functions. Such functions can be used, e.g., for object recognition, or to calculate movements of mobile equipment. Compute functions might be located in the equipment itself, or for instance, within an edge server, depending on the tasks that need to be performed by the compute function.

Firstly, the compute functions have to be abstracted in order to allow mapping into generic hardware resources and find the best matching supporting devices. Within this document, compute functions are mapped to FunctionalEntities within an AutomationComponent (cf. Figure 3.1). As the compute functions described by use cases are part of a distributed application (e.g., control of an exoskeleton), the corresponding FunctionalEntities come with a defined input/output behavior (see Figure 4.2). This can be described as ConnectionEndpoint that receives input data and provides output data. In addition, certain types of physical or logical elements are needed to execute the functions required by an application. These elements are denoted as Assets and typically include, for instance, CPU, memory, and data storage. In case of distributed computing, the functionality of an application can be described using multiple FunctionalEntities, associated with multiple AutomationComponents.

Based on this abstract modelling, a service description shall enable the specification of assets, capabilities and properties required to dependably execute the compute functions of an application.

This may include RT versus non-real-time (NO-RT) computing, local versus remote storage system, compute data access throughput and speed which determines what storage technology should be used, for example, in-memory, caches, hash-based storage, etc. Finally, the processing requirements in terms of processing speed and data processing throughput need to be given. More details on the computation subservice can be found in Sections 6.2.2 and 7.3.2.

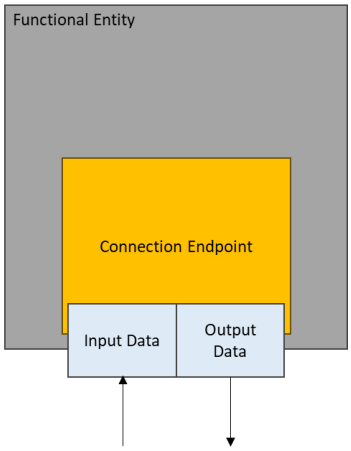

Figure 4.2: Input/output data model of a FunctionalEntity based on OPC UA FX

4.2.3 Generalized Time Synchronization Modelling

Time synchronization is essential to provide distributed systems with a global and shared notion of time. By using time synchronization, an AutomationComponent gets access to a shared perception of a common reference time, to ensure that the corresponding clocks are synchronized. The Precision Time Protocol (PTP) protocol [IEEE1588] [IEEE20-802.1AS] is typically used between Automation Components (i.e., time-aware IA stations), but other protocols (e.g., the Network Time Protocol (NTP) [RFC5905]) may be applied as well. PTP defines three different clock roles: Grand Master Clock (GMC), Ordinary Clock (OC), and Boundary Clock (BC).

UAFX also defines the usage of a clock and a time synchronization protocol. A clock can be modelled as an UAFX Asset[1], and would, e.g., contain information about whether an instance is acting as a GMC, which PTP domain (if multiple) the clock serves, etc. The protocol implementation can be modeled as a FunctionalEntity establishing a connection between the synchronized AutomationComponents.

4.2.4 Generalized Security Modelling

Security plays a vital role in industrial applications. Beside the application of generic secure-by-design principles and architectural security mechanisms, individual applications may require dedicated security services that can be provided by the infrastructure. Such services could, for instance, be hardware or software based encryption, authentication, synchronization of cryptographic keys, etc.

[1] At the time of producing this document, the OPC Foundation has not yet defined a clock type.

Modelling security with UAFX can be done by describing required hardware or software artefacts (e.g., encryption hardware, trusted platform modules, certificates, etc.) as an UAFX Asset and other security related functionalities requested from or provided by the infrastructure as FunctionalEntities.

5 Dependable Service Design

5.1 Service Description for Adaptive Applications

The use cases described in Section 2 highlight the need of dependable services provided by a DETERMINISTIC6G system as the basis for dependable applications running on top of it. Furthermore, the use cases also show how each individual application may specify different requirements on the capabilities, resources to be allocated and the reliability of these dependable system services. In a predefined and static system environment (e.g., a hard-wired production system executing fixed control algorithms) these requirements can be evaluated offline, while considering all relevant interdependencies between applications. If necessary, additional measures can be implemented (e.g., redundant communication or computation) to ensure the required level of dependability.

However, contemporary industrial applications come with increasing demands for flexibility, and 6G and TSN systems even need to deal with an additional level of uncertainty introduced by the environment. To account for this development, industrial applications, as well as the systems on which they are executed (i.e., a system implementing the DETERMINISTIC6G architecture), need to dynamically react to changing conditions and requirements. Thus, it becomes apparent that dependability of applications cannot be considered in isolation, but it has to be seen in combination with the DETERMINISTIC6G system, other applications executed in parallel, and the environment in which they operate. This is only possible if both – i.e., applications and the system – implement the right interfaces and mechanisms for managing dynamic behavior. In the case of dependable applications this implies that the application must be capable of describing the services it requires from the system platform, while the system needs a way to feedback its ability to provide a requested service. Furthermore, since multiple applications may compete for services and resources (e.g., like the bandwidth available in 6G and TSN networks), a management process has to be in place that ensures dependable and optimized system performance.

The management of complex and dynamic systems, like large wireless and/or wired networks, typically requires many decisions to be made to find good operating conditions and the deployment of corresponding configurations to all affected system components. Manually elaborating all these decisions and providing an appropriate configuration to be deployed requires deep knowledge about the whole system, the executed applications, as well as the optimization target. Therefore, it quickly becomes infeasible – already for smaller system setups. In addition, the system setup may be highly dynamic, with a frequently changing physical and/or logical structure. Examples are, added or removed hardware components like HMI devices, changed physical cabling, modified logical connections, and activated or deactivated software applications. Handling such dynamic behavior introduces high additional efforts that soon become unmanageable for human operators.

However, decision processes often follow a recurring pattern of steps that may be automated. An abstract sequence of such steps is

1. determine (possibly changed) constraints and conditions of system operation,
2. find possible solutions (e.g., target configurations),
3. rank the found solutions,
4. select the best possible solution, and
5. apply required actions to implement the targeted solution.

Automating these steps is only possible if these solutions can be described in a machine-readable format and the corresponding description can be translated into a value rank. An adequate service request of an application has to describe the functional and non-functional requirements of the requested (dependable) service(s), and it shall also provide the necessary input to rank them. The description of functional and non-functional requirements consists of a service specific listing of the minimum resources (e.g., communication bandwidth), necessary capabilities (e.g., available time synchronization), and prerequisite properties (e.g., achievable reliability) that are necessary for the successful operation of an application. Consequently, if these requirements cannot be fulfilled, the application cannot be put into operation and provide its intended value. However, in many cases, the same application could also be operated with less demanding requirements, but then, the resulting application value would be reduced. For instance, an obstacle detection application may be operated with a lower image rate, leading to an increased maximum obstacle detection time. As a further consequence, a vehicle that operates based on this obstacle detection application would have to reduce its maximum speed.

In order to enable the operation of an application at different performance levels, the application has to provide information about the supported modes and levels of operation. The implementation of such modes and levels of operation can range from switching between multiple sets of parameters within an algorithm of a functional entity (e.g., a control algorithm that supports multiple cycle times of the control loop), to completely changing the way in which a functional entity works, including a different input/output behavior (e.g., switching from image-based obstacle detection to laser-based detection). Obviously, each of these levels will come with an individual set of requirements that need to be fulfilled. An application itself can then handle the selection of the best possible level of operation by repeating service requests with a lower level, as long as the response to a higher-level request returned a negative result (e.g., due to high requested communication bandwidth). However, this can lead to a high number of service requests, which consume resources. Also, the right time to upgrade the level of operation is difficult to determine by applications, as they do not have a complete view of the system.

In contrast, it is also possible that an application provides all the different supported levels of operation to the DETERMINISTIC6G system at once. The system is then able to determine, which levels' requirements can be satisfied, and to report the selected level of operation back to the application. This additionally enables the system to automatically consider changing the level of operation of active applications, when operating conditions change (e.g., in phases with high radio interference) or during dynamic changes within the system (e.g., adding or removing of automation components or applications). For instance, such system capability may be used to automatically protect applications with high requirements on service dependability from failure, by degrading the level of operation of other less critical applications (e.g., slow down production process in order to keep safety relevant applications alive). Another major benefit of this may be the possibility to automatically upgrade and optimize the overall system performance if more resources become available or environmental conditions improve significantly.

To achieve a coordinated automated switch between different levels of operation, the service request of an application shall also consider the conditions for a seamless transition. This includes, for example, the time to handle the switch from one active level to another one.

5.2 Value and Cost Function Concepts

A description of the requested services and information about transition requirements is not enough to achieve a dependable operation of the system under changing conditions. As it can be derived from the list of steps above, in order to automate the system management process, it is necessary to introduce a method to determine the value provided by each solution in order to rank them and select the best among them. Furthermore, the transition from the currently active state to a target state (i.e., a target system configuration), by applying the selected solution, usually comes with certain costs in terms of resources, transition time, and probably time of unavailability. A positive relation between the expected added value of operating the system in a new target configuration and the corresponding cost of applying that target state may be referred to as a gain. On the contrary, a negative relation between the value and the cost of a change may be referred to as loss. In case of loss, i.e., in case moving to a new state has a higher cost than the expected added value that will be provided by implementing the change, such adaptation may be refused. Furthermore, considering gains and losses is particularly important for highly dynamic systems (with frequent changes), where the new target state is soon going to be replaced by a subsequent state. In other words, it would not make sense to change the system configuration often if the frequent changes do not come with a positive impact on the overall system performance. More details on the cost of changing from one state to a different one can be found in Section 7.

Hence, key to automating adaptivity is the provision of value and cost functions that enable the automated ranking of solutions. Note that, even though it is described here what the value and cost function consist of, and what the benefits of using them are, no specific function is proposed in this document. This is because these functions heavily depend on the application, i.e., the way industrial services provide value to the stakeholders varies greatly from area to area and the specific applications. The diversity of example fields of application is illustrated in Section 2. In this context it is also worth noting that the provided value is not only impacted by technical and economic factors but may also be influenced by the environment (e.g., the country) within which the system is operated or by societal needs. For instance, in Deliverable D1.1, KVIs have been identified that determine gains and losses of use case services when it comes to sustainability. The value and cost concept described here, could potentially include an evaluation towards KVIs in an abstract way as well, where KVI-gains could be interpreted as value and KVI-losses as costs.

Thus, besides the above-mentioned requirements, an adequate service request shall contain details about the value provided by an application that is executed with a defined level of operation. In the simplest case this can be a numeric value defining the hierarchy between the levels of operation, where the system tries to fulfill the service requirements of the level with the highest value rank. However, in a complex – and probably open – system that operates many different applications (cf. smart farming use case in Section 2.4), such simple ranking will not be sufficient. In case of limited resources, the management system has to decide which of the applications need to be prioritized (e.g., safety critical control systems) and which ones will not get any resources at all. This can be solved

by a more sophisticated value definition of the applications, that may also be composed of multiple factors. Examples may be:

- Importance classifications assigning a discrete category to the different types of applications (e.g., safety critical, high priority, medium priority, low priority)
- Economic value per unit of time (e.g., 500 €/hour)
- Societal value rank, which allows to prioritize applications, and their level of operation based on certain KVIs (cf. definition of KVIs in Deliverable D1.1 [DET23-D11]). Such ranking can be expressed in form of equivalence classes (e.g., high probability of detecting wildlife, medium probability or low probability), but also with numeric values describing gains or losses on a time basis (e.g., 100 kW power consumption)

Obviously, these values do not easily fit together. This introduces the need for value functions capable of combining the available information in order to create a prioritization among the applications and their corresponding supported levels of operation. The value functions are highly domain specific and, therefore, they may be adapted to individual installations (e.g., having different value functions in similar systems in multiple factories or when using smart farming in different countries). It may, for instance, assign fixed priorities to class-based value descriptions and give weight to numeric parameters, where the weights can be adjusted by the owner of the system.

For example, when implementing the smart farming use case in a specific country, safety critical applications will always receive highest priority. Among the applications with high priority, those with higher economic value will be considered first. The last factor would be the societal value classification. However, when operating the same system in a different country, the societal impact may be ranked higher than economic value (e.g., wildlife protection is more important than making more money).

Furthermore, the value function could also use the description of requested services as input, e.g., the communication, or computation subservices. This becomes especially important when the subservices are provided by an operator or third party, in a way that more resources consumed lead to a higher cost. This is common for cloud computing or telecommunication services. In this case, it is of utmost importance to properly reflect the value provided by the subservice, to decide whether the cost is worth it or not. Having a clear ranking of applications and their levels of operations enables to incrementally assign resources to these applications until no more resources are available. Finally, if resources need to be freed to preserve the dependability of safety critical applications, the applications with the lowest rank will be degraded.

5.3 Communication-Compute-Control Co-Design

One common application area for adaptivity is networked (i.e., distributed) control, leveraging cloud technologies for executing the control workload. Both, communication and compute, are resource constrained systems, where performance is impacted by the communication or compute loads. Communication-compute-control (3Cs) co-design is an application or system design method that understands and makes use of the relations among the following three entities:

1. (wireless) communication,
2. (cloud) compute, and
3. (control) applications.

The purpose of co-design is to enhance application and/or overall system performance. By exploiting the relationships between these three entities, the ambition of co-design is to find technical solutions that achieve improved resource efficiency, increased robustness, and advanced performance compared to traditional approaches. This may also lead to potentially relaxed requirements on each of the 3Cs components compared to independent design. In turn, this would then enable applications that would otherwise (a) not be feasible to implement due to too stringent requirements or (b) require a very costly infrastructure. Figure 5.1 illustrates the traditional approach and the ambition with the co-design approach where in addition to sending requests and providing services, insights from the compute, and communication entities are also shared in response to the requested service.

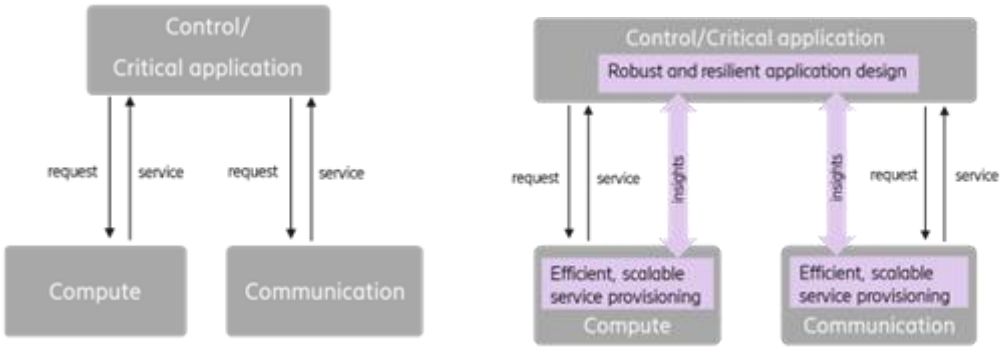

Figure 5.1: Communication-compute-control co-design (right) in contrast to a traditional design (left)

In Figure 5.2, a framework is introduced that provides a structured approach towards the adoption of co-design principles. The framework groups application design approaches according to their needs on dependability and relate these to the knowledge, insights and interactions with the communication and computation infrastructure that could support enhanced application design. Five different degrees are described that range from the lowest degree (with no insights and no interaction between the 3Cs entities) to highest degree (with very advanced insights and interactions). In-between these extremes, with every degree the detail of knowledge/insights available to the application about network/compute infrastructure increases (general awareness, specific knowledge, etc.) by using increasingly sophisticated interaction mechanisms. At the same time, with every degree, the levels of dependability the application can guarantee increases together with increasing levels of resource efficiency.

Degree	Description of application design approach	Type of application
0	Unawareness Control applications with no awareness of network and computing resources.	Non-critical applications with no requirements towards dependability nor resource efficiency.
1	Co-design with awareness Control applications with awareness on general characteristics of network and compute.	Non-critical applications with limited requirements towards dependability and resource efficiency.
2	Co-design with interfacing for information sharing Control applications designed using exposure interfaces towards network and/or compute.	Non-critical/critical applications with moderate requirements towards dependability and resource efficiency.
3	Co-design with interaction for configuration Control applications designed using exposure interfaces as well as making service requests towards network and/or compute.	Critical applications with stringent requirements towards dependability and resource efficiency.
4	Co-design with negotiation Control applications designed to negotiate with network and compute infrastructure to achieve a common overarching goal.	Critical applications with stringent requirements towards dependability and resource efficiency optimized in a larger context.

Figure 5.2: Degrees for communication-compute-control co-design

The service design developed in this report builds on a co-design approach of degree 2 or above.

6 Model of Dependable Application Service

Dependable industrial applications are vital for providing not only economic value but also societal and environmental benefits. These applications are often distributed and executed on wireless and wired communication networks. Robust subservices of the platform build the basis, where each of them is meticulously designed to ensure reliability and performance. This section offers a generic view on dependable industrial application services – which can be extended to be applied in non-industrial application domains – and outlines how this view contributes to an increased overall system performance. Afterwards, a broad overview of what makes subservices of the DETERMINISTIC6G architecture dependable is provided, highlighting key parameters and properties essential for specific applications. By understanding these elements, developers can create applications that are both resilient and beneficial to society and the environment.

6.1 General Definition of Dependable Application Service

Within this document, an application service is considered as the processes that support customers' needs and activities, so that value for the customer is created in those processes. In many processes, mobile communication is an essential building block in the service realization – and increasingly so with advancing digitalization and usage of cyber-physical system design. The application services that are investigated within DETERMINISTIC6G are industrial services focusing on industrial production, and smart agriculture targeting efficient farming and food production. But the same approach is applicable to many other application domains. One of the cornerstones addressed in DETERMINISTIC6G for such use cases is dependability: the operation and functioning of application

processes rely on, that the combination of subservices they are built upon (e.g., communication, computation, etc.) is available with a given minimum performance. Dependability can be defined as a property of systems that ensures they can be trusted to deliver correct services with a high probability. Therefore, a *dependable industrial service* is one that consistently provides the desired value with high reliability.

The International Electrotechnical Commission (IEC) defines *dependability* as the "ability to perform as and when required" [IEC23-1920122]. Dependability is a collective term that comprises multiple time-related quality characteristics and is discussed in several fora [IEC23-1920122] [IEC09-61907] [5GA23] [IR17] [3GPP23-22104]. It comprises characteristics of *availability*, *reliability*, *maintainability* and is sometimes also associated with *safety*, *security,* and *integrity*.

Dependability implies that the following is fulfilled:

- a required (*service*) ***availability*** is met, which means that the (communication) system is in a state to perform as required [IEC23-1920123] (i.e., delivering a specified performance) for a specified relative amount of the time during which the (communication) system is expected to provide the service.
- a required (*service*) ***reliability*** is met, which means that the (communication) service is performed as required [IEC23-1920124] (i.e., delivering a specified performance) in an uninterrupted manner for a given time interval under given conditions.
- the system is ***maintainable***, which means it has the ability to be retained in or restored to a state to perform as required (i.e., meeting the requirements on (communication) service availability and reliability), under given conditions of use and maintenance [IEC23-1920127] [5GA23].

This can be summarized to the definition, that a **dependable network** can quantitatively ascertain that it will deliver the required service performance for the network services according to the performance level(s) that were agreed upon.

The owner and/or user of a dependable application needs to define the purpose of usage and the value that is provided. On the one hand, this value is clearly related to the dependability measures associated with the application. In other words, the value provision depends on the correct operation of the application. And on the other hand, the value defines whether it is reasonable to execute an application at all. Especially, in a system with multiple applications competing for the same restricted resources (e.g., communication bandwidth, CPU cores, etc.), the set of applications, which is providing the highest combined value, shall get their required resources. Finding the best set of applications and the corresponding resource allocation is a complex challenge that requires a clear definition of the achievable value, which finally allows to determine an absolute rank for each application. As discussed in Section 5, the value may consist of combinations of gains out of multiple independent focus areas (e.g., economical, societal or environmental gains).

Typically, resource and service requirements of industrial applications are defined such that a dependable operation can be assured. Otherwise, if the execution of an application is disrupted unexpectedly, the desired value cannot be provided. And even worse, in case the application is there to guarantee functional safety, the overall safety of the system can be compromised. However, as the same application is often instantiated over many different system deployments (e.g., in many different factory floors or smart farms) the mentioned requirements are selected to fit for all of these

installations. In a highly dynamic environment, like a 6G system, this may mean that dependable execution is sometimes easily achieved, while in other situations it is almost impossible to keep the application operational. Fitting the application for a broad range of customers requires more conservative configurations, which may waste valuable resources in less demanding environments.

A novel approach to solve this issue can be the introduction of multiple ***levels of operation*** of a dependable application, as anticipated in Section 2. Such a level defines a set of operating conditions (i.e., including required resources and services) under which an application is capable of providing some specific value. The relation between these required conditions and the provided value heavily depends on the type and purpose of the application. The value provided by each level does not have to consist of the same combination of gains as other similar levels, and in some cases, the value of a level is focused on a single topic. For instance, a system may offer a degraded level of operation which only purpose is to maintain safe operation until a problematic situation is overcome. An automated management system (or the system operator) may then decide, which level of operation provides the best contribution to the overall system performance. Examples of levels could be *'high speed'*, *'medium speed'*, *'low speed'* and *'safe stop'* of an UGV (e.g., the harvester) in the smart farming use case, where the selection of the achievable speed depends on the precision of the obstacle detection. Obviously, the operational speed of such a vehicle has a significant impact on the provided economic value.

In contrast, many applications consist of multiple relatively independent program segments within which the application behavior changes significantly (e.g., warmup, production, and maintenance phases of a plastics injection molding machine). This is, for instance, often implemented as a state machine that switches between states depending on predefined conditions or due to interactions with the physical environment. Such state machines can be used to describe possible algorithmic sequences to control a physical process. In this document, these distinctive parts of an application are referred to as ***modes of operation,*** as introduced in Section 2. Each mode may require different types and amounts of resources. If a mode is expected to be operational for a significant duration, freeing previously used resources and requesting new ones can increase the efficiency and optimize the achieved overall system value.

6.2 Service Descriptions of Dependable Subservices

In the rapidly evolving domains of 6G systems and industrial TSN networks, ensuring the dependability of various subservices is crucial for the seamless operation of complex automation systems. This section provides a comprehensive overview of individual dependable subservices, encompassing many known concepts and parameters pertinent to their dependability. Rather than delving into specific solutions for distinct problems, the focus is on presenting a broad spectrum of relevant aspects in the context of the generalized use case model, supplemented with links to related concepts and technologies.

This section aims to guide readers in making well-informed decisions for their specific applications by addressing key questions such as the scope and purpose of each subservice, the technical components involved, and critical parameters that influence QoS and dependability. Additionally, it explores various factors that may impact these parameters, providing a holistic view of what contributes to the reliability and efficiency of the subservice.

Each subsection ends with an example list of parameters that may be part of a corresponding dependable subservice definition. However, depending on the application domain, the relevant parameters may change significantly. A set of such parameters can then be used by an application to specify the minimum requirements for the subservices, such that a defined level of service of the application can be achieved.

By the end of this section, readers will have a clear understanding of the foundational elements that underpin the dependability of subservices in 6G and TSN communication, empowering them to implement robust and resilient solutions in their systems. It shall be noted that the range of dependable subservices is not limited to the four subservices described within this document. Other dependable subservices, which still may have to be developed, can be treated in a similar manner. An example of such extension could be a dependable localization service that provides information about the location (e.g., as GPS coordinates) of a mobile device with a given accuracy.

6.2.1 Communication Subservice

The purpose of the communication subservice is to establish a connection between two or more functional entities in a system and allow for an information exchange. The communication subservice is equally relevant for wired and wireless communication; however, in this section it is described based on wireless transmission and in particular cellular communication, due to the influence of the stochastic behavior onto dependability. Nevertheless, most of the presented content can be applied similarly for wired communication.

“Communication service” is a very broad term that can be used to describe communication services on very different levels, e.g.: on a system-level the communication service could describe the different entities in the system that need to communicate with each other, and the characteristics related to these. On a user equipment (UE) level, a communication service could describe the different types of communication that all applications on the UE require, including their characteristics. On an application-level, a communication service could describe the different communication streams that the application is using and what communication performance those require. Finally, a communication service can be described on stream-level. The description of the communication service below is on stream-level but points to other levels when needed. We assume two communication endpoints, connected to an application, and between which information is exchanged in messages or bursts of messages as one stream or flow of data.

A communication service can be characterized in different dimensions with respect to the traffic to be supported:

- **Performance parameters** for defined traffic characteristics, which define the required QoS, e.g. in terms of data rate, latency (see e.g. [3GPP23-22261] [5GA23]).
- Provided **availability** (as defined above) of the communication service. Communication service availability can be conditioned both in time as well as geographical area.
- Provided **reliability** (as defined above) of the communication service. It can be conditioned both in time as well as geographical area.

Communication service availability and reliability are tightly coupled and conditions on time validity, geographical validity can both apply, defining where and when the service is required. Another important characteristic of communication services is their maintainability (as defined above), i.e. their capability to retain the service delivery or restore it.

For **dependable communication**, reliability, availability, and maintainability are crucial elements that need to be quantifiable in order to be able to ascertain a guaranteed communication service performance. A dependable communication service can only be ascertained in connection with a specific application and its traffic characteristics. In an industrial environment, multiple categories of traffic characteristics are distinguished, which are also referred to as traffic types [5GA21b] [IIC19] [IEC24]. Traffic types provide an abstraction from the characteristics and requirements of individual connections between functional entities of an application. Examples are isochronous, cyclic, event-based, configuration, or best-effort traffic. Based on such predefined traffic types, the system may determine the most suitable QoS mechanisms that enable the dependable communication.

The network providing the communication services needs to be able to observe the performance that it provides. Thus, the network can enable preventive service assurance measures (like adapting the priority for the traffic stream or reserving additional resources) in case that the delivered performance is close to (or even below) the guaranteed service performance and there exist mechanisms to enable performance accountability on the performance of the communication service that has been delivered.

The network may specify different types/profiles of dependable communication services. These may differ in their supported communication service performance characterization. As an example, an advanced dependable communication service may give high guarantees on providing communication with low latency variations for medium-data-rate-traffic while another more relaxed dependable communication service may give lower guarantees on providing the same communication service. Another communication service could provide high guarantees on its performance, but the performance characteristics may be less strict, e.g., allowing a larger range of acceptable latencies, etc. The desired service characteristics to be enabled by the network, need to be already considered and be accounted for during the network deployment, to ensure that robust coverage and capacity is available.

Moreover, the network may support certain dependable communication service profiles only in certain areas or during certain times. Figure 6.1 illustrates a factory floor where different example communication services are depicted. Here, in the office area, no dependable communication service is needed. In the production area a dependable communication service A is ensured and for the critical production area a dependable communication service B with even more stringent requirements is deployed.

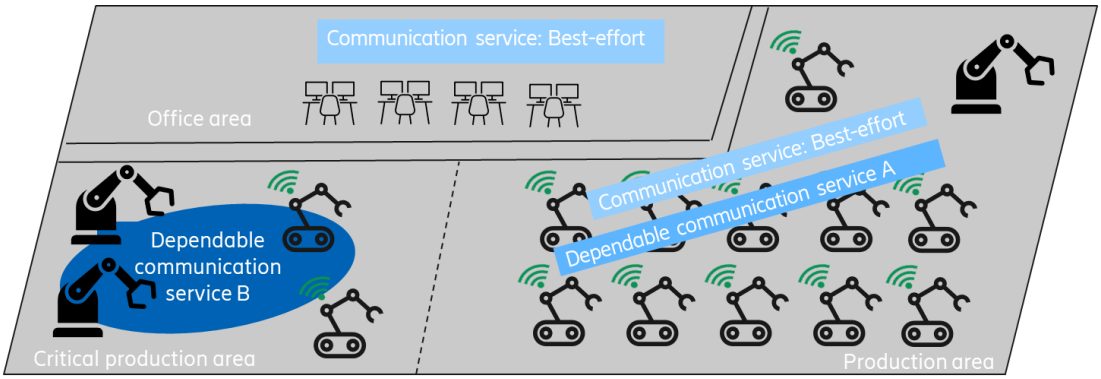

Figure 6.1: Factory floor with different communication services (Source: Ericsson)

When it comes to dependability, additional information from traffic flow/application/system side is desirable for dependable communication. For instance, a dependable communication service would benefit from an understanding of relations to other traffic streams, which are sharing the same communication resources, and their relative priority. In a congested network, priorities decide on what communication services will continue to be supported, and which is faced with an increased risk of being discontinued. A crucial difference from a dependable communication service compared to a best-effort service is, with respect to maintainability, that a dependable communication service should notify the involved entities when the service is (or is foreseen to be) no longer supported. Enriching a communication service request with an indication on what measures to take in case the first-choice communication service request can no longer be supported is of great value. This could for instance be an indication on moving to a different dependable communication service profile in case the originally requested profile is lost, thereby allowing service continuity, even though with changed terms. Another example of enriching information is that the communication service request could contain a desire to flexibly switch to a more advanced communication service profile whenever possible. Additional information could also indicate preferences when it comes to energy efficiency or other sustainability parameters. In the following, an example for requesting a communication service is outlined together with the information provided by the application/stream during service inquiry and service operation. An entity requests a communication subservice, providing information on e.g., source and destination, time to start the service, duration, etc. as well as specifics on the traffic flows and their characteristics. The network posts its generally supported communication service portfolio. The entity determines a dependable communication service profile based on knowledge about its own traffic characteristics and needs towards performance and dependability parameters. The entity provides additional information on the priority of the traffic stream in relation to other streams as well as how to respond to changes in the provided communication service. If the dependable communication service is available in the area and time interval requested, the service is provided with a high level of guarantee, and the communication channel is established.

Example selection of parameters for service description

Next, an exemplary subset of parameters is listed that could be used in the communication subservice description. This is not intended as an exhaustive list, but as a set of examples that can be extended in a real scenario.

Table 6.1: Example of parameters for the definition of the communication subservice

Parameter	Type	Description
Latency	Integer	Performance parameter for supported range of traffic characteristics
Data rate	Integer	Performance parameter for supported range of traffic characteristics
Packet delay variation, jitter	Integer	Performance parameter for supported range of traffic characteristics
Reliability	Decimal	Specified percentage, see definition above
Availability	Decimal	Specified percentage, see definition above
Maintainability	Integer	Average time required to repair a connection after a failure or needed for maintenance
Traffic type	Enumeration	Selection of the traffic type that shall be associated with the requested service (e.g., distinguishing cyclic or event-based communication)

6.2.2 Computation Subservice

The computation service offers the required resources and capabilities for performing computing tasks. Basically, the computing resources consist of memory, i.e., Random Access Memory (RAM), storage, i.e., disk or volatile memory, and computing power, which can be described as assets within an automation component. In virtualization platforms the computing power is abstracted as number of virtual CPUs (vCPU). However, it is difficult to generalize the concept of computing power since it depends on the processor architecture (i.e., RISC, x86), the processor cycles or speed, memory access speed, etc. The processor architecture determines the computing power, where Graphics Processing Units (GPUs) will excel in processing of parallel tasks compared to the generic x86 architecture.

Therefore, in addition to the amount of processing power (i.e., CPU, memory and storage), the computation service should include the following parameters to describe a dependable computation service that the consumer can select based on the required computational demands:

- RT or NO-RT:

  Real-Time Computing (RTC) requires a response time within a given target time. The system must provide a computing result to a request with a limited response time or before a given deadline. In Non-RTC the tasks scheduling optimizes the *average* response time for interactive tasks, e.g., round-robin scheduling and/or throughput of tasks (batch jobs). The Non-RTC cannot guarantee a deadline where scheduling algorithms like EDF or Rate Monotonic Scheduling with thorough scheduling analysis and acceptance tests are applied) . The RTC requires appropriate task scheduling algorithms and preemption of tasks depending on their priorities. The preemption will allocate the necessary CPU interruptions and processing cycles to the task that requires RT computing.

  RT tasks can further be classified into hard, firm, and soft RT tasks. A hard RT task must always be finished before the given deadline, as otherwise the application may fail to operate. Such tasks are usually found in highly safety-critical applications, like fly-by-wire systems. In contrast, firm RT tasks may tolerate missed deadlines to some extent, however, the value of the executed task cycle may be lost or degraded. Examples are video streaming applications where the quality of the video depends on the regular processing of frames, or continuous localization applications determining the position of an object. Finally, the correctness of soft RT tasks is usually not compromised by missed deadlines, but the user experience suffers from delays. For instance, a user interacting with the system expects a timely response. A properly designed system shall consider the different aspects of RT tasks in order to optimize the dependability of all applications that are executed.

- Volatile or Persistent storage (Storage access speed):

  Volatile storage will not persist during power cycles and a common example is the RAM versus mass storage that is persistent when powered off. Volatile storage provides faster access and is used as primary storage during computing processes. Persistent storage is used for maintaining information that would survive the power out events and allows replication across different computing platforms.

- Processing capacity (# CPU)

  Processing capacity measures the number of operations the processing unit can perform and measures the number of cycles per second. During each cycle the processing unit will perform arithmetic or logic operations as well as input/output operations to read or store values from memory either volatile or persistent as well as accessing other units such as networking interfaces.

- Processing type (CPU, GPU, DPU, IPU) and Hardware acceleration

  A CPU processing unit is a general-purpose asset for executing generic program instructions, as in contrast to a GPU, which is designed for graphics processing that requires parallel processing as well as geometric calculations required for graphics processing. Data Processing Unit (DPU) or Infrastructure Processing Unit (IPU) consist of enhanced CPUs integrated with network interfaces for dedicated processing of networking operations including transport protocols processing, encryption and decryption for private networks and firewall functions.

  Acceleration hardware consists of additional components designed to perform a specific task such as a DPU for infrastructure processing. The acceleration hardware might be integrated with a general-purpose CPU, GPU or DPU module for performing mathematical calculations required for encryption or decryption.

- Run time memory capacity (# MB RAM)

  The run time memory consists of RAM storage required for storing all the computing operations during the processing time. This storage would be used only during computing time and requires a high number of input/output operations and might consist of caching storage connected directly to the CPU in addition to RAM memory.

- Input/output capacity (# read/write operations/second)

  The input and output operations per second (IOPS) are used to measure the access to resources required for the computing. These IOPSes consist of interruptions that the CPU performs to access data for the processing tasks. The resources accessed during such IOPSes consist of storage memory, network interface access, peripherical devices such as screens, printers, etc.

- Static or autoscaling

  A computing task would be allocated a certain amount of assets, like processing capacity, memory, and storage to perform the tasks. The required processing and storage might increase during the computing tasks, which require the system to automatically allocate additional resources.

- Mobility (allow computing migration)

  The computing service is performed as a functional entity within an automation component that provides the required assets, i.e., the CPU, memory, and other required resources. Mobility allows the processing platform to migrate all the resources to another automation component to continue the computing without interruption. Mobility requires that all the state and external information required for

the computing service can be moved together with the resources to the new automation component.

- Type of computing platform (bare metal, virtual machine, container)
  The computing service can be performed in dedicated hardware or in a virtualized platform. Virtualization facilitates the mobility of the processing to a different platform since all the resources are contained in the virtual server that includes the operating system with the CPU, memory, and storage required for the processing. The virtualization platform can consist of an entire server or virtual machine that includes the low-level operating system and required modules equivalent to a physical hardware. The virtualization platform can be a container that includes only the resources for the processing provided by the platform.

For a dependable computation service, besides the typical parameters, also reliability and latency of computation should be described (the related enabler technologies are highlighted in D3.3 [DET23-D33]):

- CPU scheduling method (e.g., deadline scheduler), CPU pinning. RT or time-sensitive applications (including the ones with time-constrained networking support) require appropriate scheduling, which is to be described in the service request.
- Availability and reliability of the resource (e.g., 99.999%). Availability of a compute resource may depend on various implementation factors influencing its reliability or mean time between failures (e.g., HW quality, power supply, etc.), as well as the average time until a failure is detected and resolved – also known as mean time to repair.
- Network Interface Card (NIC) capabilities can be an important feature to support dependable networking. Requirements here may include a NIC with special hardware capabilities like virtualization or field programmable gate arrays (FPGAs). Hardware offloading of a specific task, e.g., precise hardware timestamping, is an example for an advanced NIC hardware. Smart NICs can incorporate additional features.
- Affinity and anti-affinity with other service components, which is already part of some service descriptors (see e.g., Kubernetes). For a dependable, time-critical compute service the components of the service may be desired to be placed as close as possible to each other (e.g., same blade or node) to reduce delay. In contrast, for safety reasons in other types of applications, it may also be desired that compute services are not executed on the same physical hardware, or even be executed on automation components with a minimum required physical distance (e.g., highly safety-critical applications like nuclear power plants).
- Time delay between service components may be specified in the service (graph) description. In this case it is up to the orchestration process to place these components to fulfill the delay requirements.

In case of strict timing requirements, like IEEE Std 802.1Qbv networking, or strict packet loss requirements, the compute service must also be able to support them.

- RT networking support of the compute resource (TSN/DetNet). On one hand, it is related to the usage of an appropriate NIC in the undelaying infrastructure, but on the other hand

appropriate software support is needed for it. This may include specific network driver software or kernel modules.

A common method for industrial applications – beside many others – to achieve the required dependability of the network is the usage of replicated communication. Depending on the network architecture, this may need to be supported by the compute service:

- For instance, in TSN networks, IEEE Std 802.1CB – Frame Replication and Elimination for Reliability (FRER) endpoint support may be needed in the compute service; in case the endpoint is located in the requested compute entity (container or virtual machine). This may be realized as a pre-deployed component, e.g., in the form of a network driver or kernel module.

Robustness is also a key aspect of ensuring dependable service deployment in the compute domain:

- One approach is to leverage the built-in features of the cloud management systems (e.g., Kubernetes) to ensure reliable application deployment. The most common option is to deploy multiple instances of the compute service. The number of the replicas can be specified in the service description.
- Depending on the deployment, one alternative is to apply load balancing between the replicas, in which case the traffic is automatically distributed across multiple instances by the cloud management, ensuring that no service instance becomes a bottleneck. Another approach is to forward the traffic simultaneously to all replicas.
- In the case of a complex application with multiple functional entities, the compute service description may specify the relationship between the functional entities from a reliability perspective. Specifically, the application and the relationship between the functional entities it is composed of can be described using a service graph. By leveraging affinity and anti-affinity rules, it can be specified which functional entities should be deployed using independent hardware and software resources to ensure robustness.
- If replicas are applied in the compute domain and IEEE Std 802.1CB FRER is used in the network domain, the compute service description needs to include deployment details for seamless FRER support. Cloudified FRER components can support a single active instance of a functional entity, or alternatively, multiple active replicas can also be served by multiple FRER components in a seamless manner.
- The use of a self-healing toolset can also be part of the service description, ensuring the automated re-start or re-deployment of service components (i.e., functional entities) to provide a reactive solution for handling failures.

The combined usage of reliable service deployment and self-healing offers the highest resilience against extraordinary events – leveraging the replicas to keep the service operational even in case of a failure, while the self-healing mechanism can be triggered to automatically restore the specified service robustness.

The computing subservice will expose the available resources and capabilities using this list of parameters to determine the computational requirements and determine whether the automation component, where the service is offered, can support the application demands. This information might need to be associated to the application when moving across different automation component.

Moreover, these parameters indicate the dependability of the application to concrete computing requirements that allow the application to migrate to a different automation component.

Example selection of parameters for service description

Below in Table 6.2, a subset of parameters is listed that could be used in the computation subservice description. This is not intended as an exhaustive list, but as a set of examples that can be extended in a real scenario.

Table 6.2: Example of parameters for the definition of the computation subservice

Parameter	Type	Description
CPU	Decimal	Computation resource in CPU equivalent
Memory	Decimal	Memory amount
Storage	Decimal	Storage amount
Virtualization type	Enumeration	Discrete selection of allowed virtualization types, like bare metal, virtual machine, or container
HW acceleration	List	Listing of required HW acceleration assets, e.g., GPU
Network capability	Decimal	Minimum network bandwidth, e.g., 10 Mbit/s
CPU scheduling	Enumeration	Discrete selection of CPU scheduling algorithm, e.g., deadline scheduler
Availability	Decimal	e.g., 99.999% of time
Nr. of replicas	Integer	Number of application instance replicas
RT networking support	List	Specifying the required RT networking capabilities, e.g., IEEE Std 802.1Qbv

6.2.3 Time Synchronization Subservice

In the evolving landscape of 6G and TSN networks, a reliable time synchronization service is essential for ensuring robust, high-quality communication and automation across a wide range of applications. It provides a reliable timing reference for distributed components within the 6G and TSN network. It orchestrates synchronization (aligning clocks to a common reference time) and syntonization (maintaining consistent frequency across clocks), providing a foundational element in maintaining precise timing standards required for the use cases presented in Section 2. This ensures that devices, sensors, and control systems operate in unison, reducing timing discrepancies that could lead to data loss, control errors, or performance degradation. By aligning clocks within microsecond precision, the subservice supports time-sensitive communication, enhances data integrity, and improves the overall QoS in applications such as autonomous vehicles, robotics, and industrial automation. In the next paragraphs a description of this time synchronization subservice in 6G and TSN networks is provided, highlighting key factors that influence its quality, dependability, and practical implementation.

Within the generalized use case model described in Section 4, the time synchronization subservice targets the maintenance of a common understanding of time in a defined set of automation components in a network. Time is represented within an automation component as the value of a physical or logical clock being an asset of the corresponding automation component.

While a time synchronization service in general can be implemented using different standardized protocols like the Network Time Protocol (NTP) [RFC5905], the remainder of this section is focusing on the PTP [IEEE1588]. It operates using a hierarchical clock structure, with a GMC acting as the timing source for all the other clocks in the network. This architecture is typically organized as a spanning tree, where synchronization signals propagate from the automation component acting as GMC to all the other automation components hosting their local clocks (also referred to as secondary clocks), traversing through intermediate nodes in a multi-hop topology. At each secondary clock, the **time offset to the GMC** is calculated, which also represents the most prominent performance indicator for time synchronization. Even if this offset is compensated at the secondary clocks (i.e., by using offset and frequency correction mechanisms), some clock error may remain that cannot be corrected due to imprecisions in measurement. The quality of the time synchronization subservice depends not only on its precision and accuracy but also on its resilience to failures or network changes.

Dependability measures for the time synchronization subservice

Reliable and accurate time reference source

Synchronization begins with a reliable and accurate time reference source also known as a grandmaster. The time synchronization subservice operates using a hierarchical clock structure, with a GMC acting as the timing source for all the other clocks in the network. For a clock to qualify as a GMC, three critical attributes come to play: the precision of its timing source (such as a quartz oscillator, atomic clock, or GPS), its granularity and the accuracy of its time. ***Precision*** refers to the consistency or repeatability of the clock's measurements. It indicates how much the clock's timekeeping deviates from itself over multiple samples or measurements. High precision means low variability between time outputs. ***Granularity*** refers to the smallest time interval that the clock can differentiate. It is essentially the resolution of the clock's time-keeping capability. If a grandmaster clock has a granularity of 1 nanosecond, it can distinguish time intervals as small as 1 nanosecond. If its granularity is only 1 millisecond, it can only differentiate times at that scale. ***Accuracy*** is the degree to which the clock's timekeeping matches the true or reference time, such as Coordinated Universal Time (UTC). It measures the correctness of the time reported by the clock. If a grandmaster clock is accurate, the time it displays or distributes will be very close to the actual UTC time. Inaccuracies might arise due to clock drift, network delays, or calibration issues. Traditional terrestrial networks have relied on Global Navigation Satellite System (GNSS) receivers to access highly accurate time. GNSS receivers derive their time from the atomic clocks onboard the satellites. In general, atomic clocks possess the capability to provide clock frequencies with unparalleled ***accuracy***, surpassing the capabilities of any other physical device, such as a quartz crystal oscillator [KO87]. On the one side, not every device can be connected to expensive GNSS receivers to allow access to accurate time given the high maintenance and deployment costs. Additionally, the performance of GNSS receivers is limited in indoor environments. On the other side, the clock granularity can be physically limited by its oscillator’s frequency.

The importance of these performance measures heavily depends on the intended use of time within an application. Many industrial applications need a timing source to drive cyclic control loops, where the accuracy of the clock value plays a subordinate role. There the precise length of the control cycle is of higher interest, especially if control loops involve multiple networked devices. Such control loops

can often be operated without specific interpretation of the absolute time value. However, a stable periodic trigger and precise relative time measurements need to be maintained, even across multiple devices within a network. To simplify the implementation and prevent discontinuities in the clock values, a dedicated monotonically increasing clock without predefined epoch (i.e., starting point of time measurement) can be used, which is also referred to as ***working clock***.

In contrast, many other applications need a timing source that allows interpreting the absolute time value, and therefore, can be related to global time scales like UTC or International Atomic Time (TAI). Examples are applications that generate timestamped data which is interpreted by humans (e.g., logging information) or applications that directly use the absolute time value within the algorithm (e.g., to distinguish between heating and cooling periods in winter and summer, to calculate sunrise and sunset, etc.). The corresponding clock is also referred to as ***global time*** or ***wall clock***.

A combination of working clock and global time semantics is also possible for multiple types of applications. However, special care has to be taken to correctly handle discontinuities in the global time standards (e.g., leap seconds) and to correctly manage situations where synchronization with the source of the global time (e.g., a GNSS) is (re-)established. Otherwise, unintended clock behavior may lead to critical situations.

Byzantine clock failures occur in distributed systems with clocks prone to transient failures, i.e., when some clocks in the network behave arbitrarily or maliciously, leading to faulty synchronization, conflicting time values, or failure to send updates. These failures pose challenges to achieving consensus because some nodes may report incorrect or inconsistent time information. To address this, Byzantine Fault Tolerant (BFT) algorithms ensure system reliability as long as fewer than one-third of the nodes are faulty [SW04]. Fault tolerant clock synchronization protocols, redundancy, majority voting mechanisms, and hybrid models combining hardware (e.g., secure GPS clocks) with software algorithms are commonly used to mitigate these issues and maintain consistent system operation.

Network topology

Apart from the clock's intrinsic time keeping properties, the synchronization quality also depends on the quality of the communication link between the local clock and the GMC. Wireless links are inherently more error prone as compared to wired links. Hence, the synchronization accuracy achieved at an end-station connected via a wireless link to its GMC is lower than for an end-station connected via a wired link to its GMC. As we move towards large scale industrial networks, the overall network topology and the location of an end-station in relation to its GMC also affects the synchronization quality in the network. The synchronization error accumulates over multiple hops. Hence, end-stations farther away from the GMC have worse time synchronization accuracy [GR03].

Timestamping mechanism used

The time synchronization subservice can be implemented using either hardware or software timestamping, impacting the subservice's performance in distinct ways [XYS+11]. Hardware-based time synchronization subservice is significantly more accurate due to hardware timestamping very close to the physical layer avoiding delays caused by the protocol stack and the operating system, thus

achieving synchronization in the nanoseconds range [LZZ+13]. However, automation components supporting hardware-based time synchronization require dedicated physical timestamping assets being attached to the network used for time synchronization. In contrast, a software-based time synchronization subservice relies on the reaction time and process priorities within the software system, allowing synchronization accuracies in the range of 10 to 100 microseconds. It does not require dedicated hardware and may be deployed to an automation component as software task with appropriate priority settings.

Fault Tolerance

In case of failure of the GMC, the choice for a new GMC is made using the Best TimeTransmitter Clock Algorithm (BTCA). BTCA evaluates all potential master clocks within the network and chooses dynamically the most suitable candidate based on a set of predefined criteria, including clock quality, priority, and identity. Each eligible (i.e., GMC capable) device periodically shares its clock properties in the form of Announce messages. The algorithm is executed locally at each automation component participating in time synchronization, and it ensures that only one device is designated as the GMC at any given time, while all others act as slaves. When a network or device boots up, the BTCA takes time to identify and select the best GMC. Network boot sequences may lead to undesirable GMC selections if a lower-quality clock boots faster than a higher-quality one or while parts of the network are isolated from each other. Additionally, during this period, devices might not have an accurate time source, leading to temporary time discrepancies. This delayed startup can be problematic in systems that rely on immediate synchronization for precise operations. If multiple devices are starting up or joining the network at the same time, the BTCA may experience fluctuations while trying to establish a consistent GMC. This may result in brief periods where no device acts as the GMC, causing synchronization lapses. Additionally, BTCA is unable to detect instability of a GMC, hence it may lead to a ping-pong effect to find the best GMC [DET23-D22]. To prevent unintended behavior, applications requiring a stable distributed common notion of time need additional means to ensure that all involved automation components are eventually synchronized to the same GMC.

Restricting the number of devices configured to act as GMC candidates can greatly reduce the likelihood of frequent GMC switches and simplify the BTCA decision process. By designating only one or two high-quality devices as GMC-capable, the network gains stability as there are fewer potential transitions between different GMCs.

Creating engineered synchronization trees, where the hierarchy and potential GMC roles are pre-configured, is especially beneficial for industrial automation applications. In such setups, automation components are organized in a structured way to ensure that the GMC selection is predictable and optimized for the network's operational requirements. External port configuration is already supported by several time synchronization protocols. Pre-determined paths help avoid unnecessary recalculations by BTCA and maintain synchronization stability. By engineering the tree to minimize hops and latency, synchronization can be made more efficient, whereas BTCA in its standardized implementation does not consider network topologies [SDL+21].

A standardized approach to deal with BTCA shortcomings, and to enable seamless switchover in case of a GMC fault, is the introduction of a hot standby GMC [IEEE24-802.1ASdm] in the network. The

standard defines two independent GMCs, which are termed as primary and hot standby GMC. Each of them uses a dedicated time domain to transmit their local notion of time. The hot standby GMC synchronizes itself with the primary GMC before transmitting timing messages within its domain. This procedure ensures that the time provided by both GMCs remains within a tolerance range, resulting in consistent time provision. The automation component designated as end station uses the primary time domain for its application as long as the hot standby system is in the redundant state (i.e., both time domains are available). Having a hot standby eliminates the time BTCA would need to identify and transition to a new GMC, and thus, providing near-instantaneous switchover.

Example selection of parameters for service description

In this section a subset of potential parameters is listed that could be used in a synchronization subservice description. This is not intended as an exhaustive list, but as a set of examples that can be adapted in a real scenario. An application can request a certain level of this subservice by selecting appropriate values for these (or similar) parameters. It shall be noted that the parameters define the worst-case values. As long as the system operates within the requested worst-case bounds, it can be considered dependable.

Table 6.3: Example of parameters for the definition of the synchronization subservice

Parameter	Type	Description
Maximum accuracy	Integer	Maximum offset between the clock of an automation component and the master clock.
Maximum precision	Integer	Maximum offset between the clocks of any two automation components that constitute an ensemble, i.e., which interact within an application.
Minimum hold-over time	Integer	Minimum time the system may remain operational after the synchronization to the master clock is lost.
BTCA allowed	Boolean	Some applications require fixed position of the GMC within the network for stability reasons, so BTCA is not allowed in such cases.
Time synchronization reliability mechanism	Enumeration	Discrete selection of required mechanism that translates to availability of the corresponding fault tolerance mechanisms. Examples are hot standby or BTCA.
Clock domain	List	Discrete selection used to indicate if the application requires synchronization with the working clock, the global time, or other clock domains. Requires a standardized method for identifying domains.

6.2.4 Security Subservice

Integration of time sensitive communication on the top of 6G networks broadens the attack surface, introducing a range of new vulnerabilities as TSN needs to evolve from its traditional deployment in closed, controlled environments to more open and heterogeneous wired and wireless communication links. For example, wireless communication lacks the inherent physical protections of wired systems, making it far more susceptible to interference, signal degradation, and other unpredictable environmental factors that can disrupt the deterministic nature of TSN. Attackers can exploit these vulnerabilities to launch targeted disruptions or degrade service quality, leveraging the stochastic

behavior of wireless channels to induce issues such as packet loss or increased jitter. Furthermore, GPS signals, often relied upon by grandmaster clocks for precise time synchronization, present a critical vulnerability, as they are prone to jamming and spoofing attacks, potentially causing widespread disruptions across the network. The risks are further amplified in inter-domain communication scenarios, where multiple operators and service providers are involved. Misaligned security policies or coordination gaps between domains can create exploitable weaknesses, allowing attackers to compromise the system. To provide secure dependable communication links on top of 6G networks, a security subservice must be developed to swiftly detect and mitigate these vulnerabilities while preserving the low-latency and high-reliability requirements. Effective countermeasures could include using advanced encryption protocols, secure synchronization techniques, and AI-based intrusion detection systems that are tailored to the unique challenges of deterministic wireless communication.

A security subservice ensures that the dependable application services remain available and resilient against potential disruptions or malicious attacks. It safeguards the integrity and confidentiality of data transmitted across dependable communication links, thereby preserving the trustworthiness of the system. To achieve this, the security subservice itself must demonstrate high reliability, performing its functions accurately and consistently over extended periods. Evaluating the efficiency and effectiveness of security mechanisms requires defined quality and performance measures tailored to the specific needs of the system. Some common metrics for assessing this subservice include – but are not limited to – latency, accuracy, availability, and resource efficiency.

Latency measures the time taken by security operations, such as encryption, authentication, or attack detection, and their effect on overall data rate. Excessive latency or reduced data rate can indicate bottlenecks within the security subservice, potentially impacting the system's RT responsiveness. The ***accuracy*** of the security subservice, particularly in threat detection, is often quantified by its false positive rate, where too many false alarms can lead to alert fatigue and diminished operator response effectiveness. ***Availability*** reflects the subservice's ability to deliver security services reliably, even under conditions of high demand or during an ongoing attack, ensuring continuity of protection. ***Resource efficiency*** measures the consumption of computational resources, such as processing power, memory, and bandwidth, by the security subservice.

For resource-constrained environments like drones or IoT devices, optimizing resource efficiency is essential to balance security with operational demands. A highly efficient security subservice not only protects the dependable application but also supports its overall performance without introducing undue strain on network or device resources. By achieving a balance across these metrics, the security subservice ensures robust protection while maintaining system dependability, particularly in scenarios requiring stringent performance and reliability.

To ensure the integrity and confidentiality of data transmitted across communication links, security subservices can leverage existing encryption and authentication protocols tailored to specific network layers. For instance, at Layer 2, IEEE Std 802.1AE MAC Security (MACsec) is widely used to secure TSN data plane frames, protecting against unauthorized access and tampering. At Layer 3 and above, protocols like Secure Shell (SSH) and Transport Layer Security (TLS) can be used to secure control plane communications, such as those transmitted via NETCONF, by encrypting and authenticating the data. For time synchronization protocols, PTPv2.1 enhances security in PTP implementations by supporting message authentication, effectively mitigating risks such as man-in-the-middle attacks, replay attacks, and malicious Master spoofing. However, the E2E protection of PTPv2.1 excludes from the network

any transparent clocks, where the correction field in PTP messages is updated, preventing the verification of message integrity. These examples illustrate the critical role of choosing appropriate security protocols to balance protection with functional requirements, particularly in systems requiring precise timing or deterministic behavior. Each protocol's capabilities and constraints must be carefully considered to ensure comprehensive protection without compromising the system's performance as it introduces additional latency and jitter, thus often making it unacceptable. By deploying such layered and context-aware security measures, communication networks can effectively safeguard data integrity and confidentiality while supporting robust and resilient operations.

In dependable communication links, not only data but time is also vulnerable. For example, a time-delay attack, which involves delaying either PTP *delay_request* or *sync* messages without updating its correction field, can disrupt the calculation of propagation delay and residence time, leading to inaccuracies in the PTP synchronization process. To address this kind of threat, the security subservice must incorporate a sophisticated Intrusion Prevention System (IPS) as an additional layer of defense. This IPS should continuously monitor network activity, detect anomalies indicative of time-delay attacks, and proactively mitigate their impact to preserve the integrity of the time synchronization process.

Typically, a security subservice consists of four key components: policy management, key and identity management, telemetry and monitoring, and a security network orchestrator, each serving a distinct role in safeguarding network operations. The policy component is responsible for defining, managing, and enforcing security policies across the network. These policies may cover access control rules, encryption protocols, and communication standards, ensuring that every device and connection adheres to the established security framework automatically. The key and identity management component handles the generation, storage, and distribution of cryptographic keys and identities, ensuring that only authenticated and authorized devices, users, and services can access network resources. This component ensures the integrity of authentication systems and provides secure means for key exchange, which is essential for maintaining trust across the network.

The telemetry and monitoring component plays a vital role in threat detection and system analysis by continuously gathering network behavior statistics. These collected statistics enable RT detection of anomalies, such as unauthorized access attempts or policy violations, allowing for rapid responses to emerging threats. Automated responses initiated by this component can contain potential breaches before they escalate, preserving system integrity and availability. At the core of these operations is the security network orchestrator, which ensures all components work cohesively. This orchestrator dynamically coordinates security policies, key and identity management, and monitoring activities to adapt to evolving threats and changes in network conditions.

The orchestrator also resolves inconsistencies between components, ensuring unified security operations even in complex and distributed environments. For example, if telemetry detects an anomaly, the orchestrator can automatically trigger policy updates or key revocation to address the threat in real-time. Additionally, it facilitates seamless integration of new devices or services into the network by coordinating policy enforcement and authentication processes, ensuring these additions meet existing security standards. This dynamic coordination reduces operational complexity and enhances the system's ability to adapt to new requirements without manual intervention. Together, these components create a comprehensive and resilient security subservice that not only protects

dependable application services but also ensures it remains flexible, reliable, and capable of supporting dynamic, high-demand environments.

Example selection of parameters for service description

Next, a subset of possible parameters is listed that could be used in the security subservice specification. This is not intended as an exhaustive list, but as a set of examples that can be extended in a real scenario. Specifically, for the security subservice it may be difficult to specify a set of parameters for the service request. However, a dependable application must be able to request security related capabilities and parameters required for its correct operation.

Table 6.4: Example of parameters for the definition of the security subservice

Parameter	Type	Description
Maximum latency	Integer	Maximum latency introduced by the applied security mechanisms
Accuracy	Decimal	Percentage of correctly reported threats
Allowed encryption algorithms	List	Listing of encryption algorithms that are allowed to be used
Minimum key length	Integer	Minimum number of bits used for cryptographic keys

7 Service Adaptivity

Dynamic dependable systems, like those in the focus of DETERMINISTIC6G, need to flexibly adapt to changing conditions in order to obtain high efficiency, ensure dependable operation of applications, and enable demand-based execution of applications contributing to individual use cases. If applied carefully, adaptation is a powerful tool that helps to continuously optimize the system behavior and performance.

This section introduces a novel concept for a system platform enabling to automatically adapt applications executed on top of it, such that the overall system performance and value provision is constantly adjusted. First, the need for adaptation is motivated by explaining what may cause the system to adapt and where the corresponding triggers may come from. Based on these triggers, the process of service adaptation is described, which considers different levels and modes of operation. A section on requirements for service adaptation discusses relevant general conditions that need to be considered when reconfiguring services and applications. Finally, important aspects of individual subservices are presented in Section 7.3.

7.1 Triggers of Adaptivity

Within this document, three different triggers for adaptivity are distinguished, namely the ***application***, the ***system infrastructure***, and the ***environment***. As discussed in Section 2, the industrial applications of the future are no longer expected to operate in a static manner for long periods of time. Instead, the aim is to automatize complex applications which either need to change to provide adequate value to the users, or which evolve over time due to technological advancements. In this vision, adaptivity can be triggered by the application itself upon startup or shutdown, or to continue to support its own operation, e.g., a harvester that is collecting the crops and when reaching the end of the field, will

need to turn around in order to continue to harvest the remaining crops. It can be deduced that the mode of operation of the harvesting application will not be the same when it goes straight collecting crops, as when it needs to turn around, when it will likely stop harvesting to just maneuver. In case a change of the mode of operation involves a different set of system services (e.g., computation or communication resources), a corresponding request has to be sent to change the level of operation too.

The second trigger for adaptivity, as already anticipated, is the system infrastructure. Similarly to what is happening at the application level, system infrastructures are no longer designed to be unchanged for years. Instead, infrastructures should evolve and improve continuously during the life of the system. In this new paradigm, changes are the norm, and there must be means to adapt to them in a dependable manner. Adaptation triggers of the system infrastructure may be caused by modifications of the physical and logical setup (i.e., hardware, software, or logical connections), like an increase of the data rate available for the communication due to a new base station being deployed in the area. But the system should also trigger adaptation, if the priorities and other configuration parameters have been changed. For instance, if the system owner decides to increase the price for certain services (e.g., reserved communication bandwidth or reliability levels) or the value function for calculating the overall system value changes (e.g., to increase the impact on sustainability).

Finally, the third trigger for adaptivity is the environment. The novel industrial applications targeted by DETERMINISTIC6G are characterized for interacting with the environment in which they are deployed to provide value. Traditionally, this interaction with the environment has heavily influenced the operation of the system, posing many constraints, such as RT requirements or dependability. But the future systems envisioned in this project should not just be constrained by the environment, but they should also be capable of adapting to it to increase the value provided, e.g., a mobile robot in a factory should be able to stop or change its course when humans are nearby. Specifically in the context of wireless communication, the environment plays an important role, as the characteristics of the communication channel are constantly changing. A promising example of a trigger originating from the environment is the communication latency prediction as described in Deliverable D2.1 [DET23-D21]. As the latency of relevant connections between functional entities is predicted, preventive measures can be applied to ensure that the dependability requirements of the corresponding application can be fulfilled. If the predicted latency increases or drops, the DETERMINISTIC6G system may calculate new resource assignments and schedules (cf. Deliverable D3.4 [DET24-D34]) to ensure high value applications to remain operational or to optimize the overall system value by also enabling less important applications.

7.2 Service Adaptation Process

Based on one (or multiple) triggers, the applications and/or the system platform initiate an adaptation process. This process may be divided into different stages that involve the system platform as well as a subset of applications that are operated. A management process of the system has to ensure that all of these stages are executed successfully and consistently, as otherwise the whole system stability could be compromised. Thus, also the adaptation management process of the system becomes a dependable service. The following subsections provide deeper insights into these steps.

Change Request

A change request can either be triggered on the application side or from within the system platform. For instance, an application may create an initial request when it is started (e.g., a contractor starts working on a crop field with its harvesting equipment) or an exit request that handles the shutdown of an application (e.g., the harvesting activities are completed). But also, dynamic situations during the operation can lead to a change request, like when the application switches the mode of operation. Once the need for a change is identified, a dedicated adaptation management logic, which is part of the application, has to compose a new service request that lists all possible levels of operation for the desired target mode of operation. For each level supported by the application, this list includes all requested subservices of the system platform – describing the subservice parameters are presented in Section 6.2 – as well as the expected value of the corresponding level (as discussed in Section 5.2). It shall be noted that in each level a subservice – as described in this document – may also appear multiple times (e.g., if multiple communication connections are needed or several computational entities), or not at all. However, it is always the whole set of requested services that need to be provided in order to satisfy one level. Any optional subservice parameter in the service request would denote a separate level of operation, where the option shall result in a clear distinction of the value description of the corresponding levels.

Along with the requested services and the expected values, the application also needs to formulate general conditions for the execution of a change. This will be, for example, the minimum lead time before the activation of a new level of operation – including a switched level due to a changed mode of operation. But also, other system dependent conditions could be possible, like maximum service rates or constraints on the daytime. To simplify the composition of a service request, an application may use predefined mappings of modes of operation and corresponding levels of operation.

Finally, when the service request from the application is ready, it is sent to the system platform, where an internal change request is triggered. The platform has to verify that the application possesses the required permissions for all parts of the service request, as otherwise the request is rejected immediately. After updating the service database – which holds the service requests of all active applications – the adaptation planning step is started. Change requests coming from the system platform itself (e.g., based on latency prediction functions), can directly go to the next step.

Adaptation Planning

This step determines a feasible target configuration and sequence of actions that need to be taken to execute the change. As this document proposes a value-driven approach, the goal of the adaptation planning is to find a solution that maximizes the sum of weighted values. Unless a simple definition of value has been chosen for the system, this is only possible with a sophisticated value function, as introduced in Section 5.

Adaptation planning considers the following basic sequence of actions (depending on the actual implementation):

1. *Calculation of weight values:* this shall be done for all levels of operation and all active applications. It shall be noted that highly important applications (like safety critical ones) will receive high weights, and thus, will receive all the required resources in the later steps.
2. *Ordering of levels of operation:* based on the weighted values, a total order of all known levels of operation is created – which implicitly creates an ordering of the applications as well.

3. *Resource allocation and scheduling:* starting with the highest ranked application and level of operation, if possible, the corresponding services requested (e.g., communication bandwidth, computation, etc.) are allocated and scheduled for the application. In case any item of the service request is not available anymore, the corresponding level of service cannot be provided dependably, and thus, the whole level of service of an application is ignored. On the other hand, if a level of service of an application with a higher rank has already been considered and resources have been allocated, all further levels of the corresponding application will be omitted. It shall also be noted that this activity includes any other relevant check, whether a level of service is eligible (e.g., if the resulting cost of service provision is within the bounds or daytime conditions are satisfied). This activity stops if either all applications have been handled or no more resources can be allocated. As this strategy might lead to a solution that is not equivalent to the global optimum, other implementations may be developed that optimize the outcome. However, this probably increases the computational complexity and required efforts.
4. *Deployment planning:* defines a strategy for deploying the new resource assignment and scheduling. This strategy heavily depends on the currently active configuration and the targeted resource usage. In a system that has been newly started, a transition in one step can be planned. In the opposite case, where the system is already under heavy load and a major reallocation is needed, stopping and restarting many applications will be hardly preventable. Where possible, a stepwise or incremental approach is suggested to enable seamless transitions. The result of the deployment planning is a sequence of actions containing precise information about when the corresponding action needs to be taken. Reconfigurations at the wrong point in time may cause a disruption of service provision and even a failure of the whole system. In contrast, if the sequence of actions also considers the internal dependencies of the automation components that form an application, the application may switch without any interruption. For example, in distributed control applications, the information typically flows from sensor devices to controllers executing the control algorithm, to actuator devices. Considering this dependency of information exchange allows to complete an older control cycle, while the sensor devices already switched to the new configuration.

Unless major configurations change, adaptation planning will not result in a complete reorganization of resources and service provisioning. In contrast, some applications (e.g., the safety-critical ones) may benefit permanent allocations. Incremental planning helps to keep the impact of service adaptation small.

Change Deployment

A plan that has been developed needs to be deployed to the affected automation components and service providers of the system platform. This requires an adaptation management instance within an application. Before the platform services are actually reallocated, the application is informed with at least the specified lead time. Consequently, the application acknowledges the new target level of operation and prepares for the actual switch. After all relevant parts have acknowledged, the newly calculated target service provision is activated, again in the intended sequence and at the defined points in time.

In order to prevent any instability of the system and the applications running on top of it, the service adaptation process needs to ensure a consistent deployment of the calculated target configuration. This includes the possibility for a rollback, if any intermediate step fails.

7.3 Adaptivity of Platform Subservices

This section explores the subservice-specific aspects of adaptivity, addressing key questions about what can be adapted, necessary processes for adaptation, and the cost and effort involved. Each subsection will provide detailed insights into the adaptable aspects of the subservice, the conditions under which adaptations make sense, and the associated trade-offs. Adaptations incur costs in terms of computation resources, bandwidth, and time, which may impact QoS during the transition. Thus, determining the value of a change versus its cost is crucial for decision-making.

By understanding these factors, developers and researchers can optimize the adaptivity of subservices, ensuring that the platform delivers the highest value while balancing flexibility and resource utilization.

7.3.1 Communication Subservice

A communication network is bounded by a certain capacity. Multiple users of the network share those network resources. At high load situations temporary scarcity of resources may occur, if too much traffic of different applications is contending for the same resources. In wireless communication networks there can be additional variations in the instantaneous network capacity and achievable performance on a wireless communication link. The wireless link is subject to radio propagation, which consists of multiple radio paths which – in addition to distance dependent attenuations – may be subject to signal attenuations or blockage, reflections, or interference. Even if many mechanisms are in place to manage link qualities, temporal performance variations of a wireless link may occur. Higher reliability of the transmission or improved performance (e.g. lower latency or higher data rate) are possible but come at the costs of additional resources. Furthermore, the traffic capacity of a mobile network can fluctuate depending on the environment and the user behavior. If a mobile network can support a certain amount of traffic (traffic capacity) at a certain performance, the traffic capacity will decrease if the radio links of the majority of users degrade. This means, that a mobile network has a higher capacity (e.g. in terms of aggregate data rate) when all users are at locations with good radio links, compared to when all users are located at positions with poor radio links. This can lead, e.g. in high load situations, to degradations in service performance for an application. The mobile network can protect critical traffic by e.g. reducing primarily resource allocations for applications with rate-adaptive or best-effort performance demands. Nevertheless, despite a range of service assurance mechanisms, a possibility remains that applications may occasionally obtain temporarily worse communication performance than requested, and the agreed communication service availability for the performance targets is at risk.

As explained above, many critical applications have mechanisms that allow for some adaptivity, and this is expected to increase in the future. The needs of an application can vary over time, location, and environment, and in these different phases the amount of communication resources that are needed to support the application can differ. This can be exploited if the application request is not only made for the worst-case situations. Applications are also able to change to other modes of operation, which require less load on the communication network. Even if this may sometimes come at a reduced operational performance of the application, it may be beneficial compared to an application failure at network overload. For example, a mobile robot that is maneuvered such that it maintains sufficient distance to obstacles, can operate with reduced control update cycles in an empty wide alley (and requiring less network resources), compared to when it is in a complex area with many obstacles and narrow passages. A mobile network in a certain operational state is ideally capable of predicting the

communication service performance into the future, including the likelihood to meet certain performance bounds. Methods for this are developed in DETERMINISTIC6G, see [DET23-D21]. If the network perceives an increased risk of not meeting a target performance level for an application, it can proactively trigger re-planning of schedules (see [DET24-D34]) and provide a corresponding notification to impacted application(s). The network may even consider, which applications allow adaptivity and to what level, if such information was provided in the service specification. In the sequence, a degraded mode of operation can be used for some applications – with reduced need for communication resources. Similarly, the network can observe when the load in the network decreases and better performance could be provided. In such cases, the network could notify applications that have indicated their adaptive capabilities, so that such applications may use another mode of operation with higher operational value.

Reasons that may trigger such behavior are manifold. A change in the environmental decision (mainly impacting the radio propagation environment and the distribution of radio link quality values) can change the resource costs of existing services. A reason can be that e.g. many blocking objects enter the environment, or that many UEs move to (radio-wise) unfavorable positions. It can also be caused by changes in the radio network infrastructure, like the addition of a new or the failure of an existing base station. Finally, the network load can trigger adaptivity from the network. In a resource-constrained system, a typical approach to provide performance guarantees as needed for a dependable service, is to make resource reservations for guaranteed services. If the network is highly loaded, and a large amount of resource have been reserved for a multitude of critical applications, the network will not accept further service requests. A similar situation can occur when a device moves into an area where the network is close to the capacity limit. By exploiting adaptivity of applications there is some flexibility in the capacity of the network. A new service request to a network that is already operating at the limit of reserved resources may trigger the network to notify already active applications, if degraded modes of operation are possible. If this is the case and acceptable to the ongoing applications, those can be transferred to a degraded mode of operation and thereby allowing for some more new service requests being admitted.

Nevertheless, despite a range of service assurance mechanisms, a possibility remains that applications may occasionally obtain temporarily worse communication performance than requested, and the agreed communication service availability for the performance targets is at risk. One dimension of adaptivity in a mobile network is to exploit different frequency carriers that are typically available in the network and can be used flexibly by UEs based on carrier aggregation [KK22]. Traffic steering among multiple carriers can be used [OFH+23] [Eri22] [FJN24] [ABB+22] smartly and bring benefits in capacity, coverage, and performance. Together with carrier activation / de-activation it is an important capability towards network energy saving [FJN24] [ABB+22]. This allows a flexible trade-off of the network between network energy savings and network capacity control; a valuable feature to adaptively dimension the 6G network to the service needs.

7.3.2 Computation Subservice

Adaptivity on the computing service requires calculating all the computing requirements in terms of CPU, storage, memory, and the dependability from the platform resources. Section 6.2.2 presented the features that characterize the computing service and those that determine the adaptivity of the computing service.

The first feature is whether the computing service is provided using bare metal or a virtualized platform. The type of virtualization whether it is a virtual server or container will restrict the capabilities of scaling the computing resources. If the computing service is provided as virtual machine (VM) or container, the adaptivity is managed by the platform and additional resources can be allocated if available in the underlying hardware.

The support for the autoscaling and mobility feature of the computing service indicates whether all the required resources can be isolated from the platform and allows migrating the computing to a different platform. Thereby, the computing service provides the CPU, memory, and other types of resources. The required resources can be measured and monitored when a computing service is deployed in a VM or container-based platform. Scalability determines whether the computing can be scaled to improve the performance without disrupting the service if the processing is not depending on physical platform resources.

The major dependability factor of computing service is hard real-time processing during mobility. The migration of services that require real-time computing is the major challenge since the schedulability of real-time tasks depends on whether the destination platform can allocate the necessary computing resources before the migration. This requires a new schedulability analysis before the migration happens. The scheduling needs to be analyzed and adapted before, during and after the migration based on computing resources in the destination platform and expected delay during the migration.

Therefore, the dependability of a real-time computing service would include the following factors:

i. the reservation of computing resources in the destination computing platform to support real-time tasks from the source computing platform,
ii. the dependability of the communication services to be able to be reconfigured during the migration to guarantee seamless real-time connectivity, and
iii. the speed of the migration process that needs to be time bound, so the scheduling needs to be adapted while the migration is happening.

Applying the concept of communication-compute-control co-design, as introduced in Section 5.3, may yield improved adaptability of the computation service.

7.3.3 Time Synchronization Subservice

A change in clock synchronization could be changing the grand master after a failure or because the device containing the GMC is no longer part of the network, also a mobile device that moves from one network domain to another one and needs to synchronize with the new network; but one could also think of more complex reasons to change (for example, changing the synchronization protocol used). Regardless of what triggers the change, there will be actions to be taken, which imply a cost in terms of resources, time, and maybe quality. For example, changing a GMC may require exchanging messages to agree on a new GMC, consuming network bandwidth, but also consuming computation resources to calculate which is the best GMC. Since the exchange of messages and the calculations are not instantaneous, the change also has a delay. Finally, accuracy of the synchronization could be degraded during the change, implying a cost in quality. The reason to explain this is to show that there is a tradeoff between flexibility and QoS and resources, so caution should be taken when deciding to change, to make sure that the change is worth the cost. Any other impacts or costs you can think of could also be reflected here.

In general, it is not expected that in a stable industrial environment the time synchronization subservice changes its properties and behavior very frequently. However, enabling the adaptivity of the time synchronization subservice is important to ensure its dependability and flexibility to support future industrial use cases as mentioned in Section 2. This adaptivity encompasses a range of scenarios that reflect the evolving nature of industrial applications, system infrastructures and their operational environments.

The need for adaptivity could arise from how future industrial applications are evolving with increased mobility and automation. Such application scenarios may trigger changes in the application requirements during its operation. As application requirements adapt to the changes in the different modes of operation, these changes might also ripple changes in the time synchronization hierarchy. For example, an autonomous harvester might require high-precision time synchronization while collecting crops, but a different time synchronization mode (e.g., with less demanding precision) when it is moving to and from the field.

Additionally, as industrial applications become more flexible, the network infrastructures are no longer static. These changes could include new devices in the network or replacing a part of the automation production process to incorporate on-demand production. For instance, replacing a device in the network, such as an industrial controller or sensor, introduces changes that may ripple through the synchronization hierarchy. On the one hand, a new device may have different characteristics, prompting the BTCA to re-evaluate and potentially select a new GMC. This reconfiguration must be handled to minimize disruption to the overall timing precision, as even brief inconsistencies can impact critical use cases. In such situations, having a hot standby GMC in the network would be highly beneficial to handle the issues related to identifying a new GMC within the maximum hold over time.

On the other hand, when parts of an automated production line are physically decoupled for maintenance or relocation, each section may maintain its own synchronization. In such a case, the split functionality introduced by the IEC/IEEE 60802 [IEC24] would provide the important adaptivity measure needed like splitting one clock domain into two or later merging multiple time domains. Such processes require not only accurate detection of changes within a domain but also mechanisms to ensure the smooth integration of separately synchronized systems, avoiding conflicts or extended downtimes.

Achieving this adaptivity requires robust strategies for transitioning between synchronization states. For example, if two machines with independent synchronization need to connect, one must be designated to synchronize with the other's GMC, or both must synchronize to a predesignated shared reference. This transition must occur with minimal latency and without degrading precision, ensuring that dependent applications continue to operate reliably. The process must also account for risks, such as temporary loss of synchronization or reduced timing accuracy during the transition, which could compromise processes requiring deterministic communication or high-precision coordination. Thus, it is important to carefully select the supported ways of adaptation or to prevent unintended disruptions by implementing means that ensure seamless transitions (e.g., implementing hot standby redundancy of the GMC). If changes in time synchronization are unavoidable, for instance, when independent production modules are coupled and decoupled during operation, the instant of transitioning to a different mode of time synchronization shall be closely synchronized with the

dependent application(s). This means that such a transition either happens while an application is not operational, or at the beginning of a new control loop cycle.

In general, the sensitivity of applications to changes in time synchronization heavily depends on the properties that an application requires from a synchronized clock (see Section 6.2.3). Besides the achieved precision, also the requirements on the absolute time value and the monotonicity of the clock value play an important role. For instance, many CPSs use cyclic control loops that require a precise relative time measurement, but do not use the absolute time value during computation (cf. concept of working clock as indicated in Section 6.2.3). In such cases, a smooth transition between different time domains may be implemented by switching the time before the next measurement cycle starts.

The cost of adapting to changes often manifests in the time taken to complete the transition and the resources consumed during the process. For example, network bandwidth may be utilized for synchronization message exchanges, while processing power is required to compute the new timing hierarchy. Protocol specific parameters, like default timeout values for Announce messages, may have a significant impact on the stability of the systems during the transition phase. Avoiding unnecessary or overly disruptive adaptations is a critical design consideration. Prediction mechanisms, such as anticipating likely points of failure, and planning of interruptions, like planned maintenance windows, can reduce the frequency and cost of such adaptations. Additionally, system redundancy, such as backup GMCs or a sorted list of potential GMCs or preconfigured synchronization paths, can ensure that the system remains operational even during significant changes.

By designing the time synchronization subservice that incorporates these adaptive capabilities, future 6G and TSN networks can maintain a high level of dependability. Such systems are prepared not only to handle inevitable changes in devices, infrastructure, or operational requirements but to do so in a manner that minimizes disruption, maximizes efficiency, and ensures continuity in time-critical operations.

7.3.4 Security Subservice

The adaptivity of a security subservice to respond to changes stemming from applications, infrastructure, or environmental factors introduces both benefits and costs that must be carefully considered. One significant cost is computational overhead, as dynamically adjusting security configurations, policies, or cryptographic operations requires additional processing power, which can impact system performance, particularly in resource-constrained environments like IoT or edge devices. Latency may also increase due to the time needed to reconfigure policies, redistribute cryptographic keys, or implement adaptive responses, potentially affecting time-sensitive applications. Additionally, the need for RT telemetry and monitoring to detect changes and trigger adaptive measures demands robust data collection, storage, and analysis capabilities, which can escalate resource usage and operational expenses.

The complexity of management is another factor, as adaptive systems often require advanced orchestration and coordination mechanisms to ensure consistency across all components, increasing both development and maintenance costs. There are also potential security trade-offs, as frequent changes in configurations or policies might introduce vulnerabilities if not executed and verified correctly. Interoperability challenges may arise in heterogeneous environments, where adapting the

security subservice to new devices, protocols, or services requires ensuring compatibility without disrupting existing operations.

The time-to-deploy (TTD) KPI serves as a principal metric to measure the responsiveness and efficiency of the security subservice in adapting to changes. TTD quantifies the duration required to deploy new security policies in response to shifts in applications, infrastructure, or environmental conditions. This KPI directly reflects the cost of adaptivity, as longer deployment times can delay critical security updates, leaving systems vulnerable, and potentially impacting time-sensitive operations. A low TTD indicates that the adaptive mechanisms are well-optimized, ensuring swift responses to emerging threats or changing requirements without significant disruptions. By evaluating TTD, stakeholders can assess the trade-off between the benefits of adaptivity and its operational costs, enabling better design and resource allocation for security systems that balance performance, reliability, and flexibility.

8 Cyber Security Aspects of DETERMINISTIC6G Service Modelling

The adoption of service description concepts in DETERMINISTIC6G to automate resource allocation and service adaptation introduces new security challenges that must be addressed to ensure the overall system's integrity and reliability. This section explores these implications, focusing on the risks posed by these service handling mechanisms and their potential treatments.

Service description concepts, such as machine-readable service specifications, enable applications to specify their performance, reliability, and security requirements in a structured format that the system can interpret automatically. This capability allows the system, e.g., via its control plane, to dynamically allocate resources and adapt to changing demands of applications without manual intervention. However, exposing such interfaces to applications without any restriction clearly introduces security risks, such as unauthorized access or manipulation of service requests. For example, arbitrary applications can inject malicious requests, over-allocate critical resources, or disrupt the whole system’s operations. To mitigate these risks, robust application authentication and privilege enforcement mechanisms must be integrated into the interface. Each application must be authenticated using secure methods, such as multi-factor authentication or certificate-based verification. Additionally, access control policies should limit the scope of requests based on the application's role and predefined privileges, ensuring that no single application can monopolize or disrupt critical network resources.

When an application or service requests network resources, the request is handled by control-plane entities such as the Centralized Network Controller (CNC) or TSN Application Function (TSN-AF). These entities authenticate the application or service before any data transfer occurs. For example, in the case of a TSN service, the TSN Centralized User Controller (CUC) would request a service setup from the CNC. The CNC, in turn, would coordinate with other network elements, including TSN bridges and the 6G network, to configure the necessary resources. Throughout this process, the CNC ensures that the requesting application or service is authorized to use the network.

The trust model in 3GPP networks typically assumes that entities like the CNC and TSN-AF are trusted. They are responsible for the management and orchestration of network resources and ensuring that only legitimate and authorized applications are granted access. When dealing with applications or

external functions that are not directly trusted, additional security measures such as policy enforcement, access control, and secure communication mechanisms must be implemented. These measures ensure that the network remains secure, and only properly authenticated and authorized applications can interact with network resources.

Thus, authentication and authorization in 6G networks will typically take place on the control plane and, in the case of TSN, will involve entities like the CNC and TSN-AF. In this way it is possible to ensure that applications are properly authorized before network resources are allocated for data transfer. Once authorization is granted, the user plane is responsible for the actual data transmission.

Automated resource allocation allows the system to respond efficiently to service requests by means of dynamically provisioning, such as bandwidth, processing power, or security measures. This capability obviously increases the system's attack surface. For example, malicious applications or compromised interfaces could flood the system with spurious resource requests, leading to denial-of-service (DoS) conditions or misallocation of critical resources. A potential solution is to implement anomaly detection systems that monitor resource requests in real-time, identifying patterns indicative of abuse or attacks. Machine learning algorithms can enhance these systems by distinguishing between legitimate high-demand scenarios and malicious activities. Moreover, rate-limiting mechanisms can throttle excessive requests from any single application, preventing resource exhaustion while maintaining deterministic guarantees for legitimate services.

Service adaptation mechanisms allow DETERMINISTIC6G systems to modify service parameters or resource allocations in response to RT conditions, such as changing traffic patterns or application needs. While this dynamic capability ensures performance optimization, it also introduces cybersecurity concerns. Adapting service parameters without stringent validation could inadvertently weaken security measures, for instance, by downgrading encryption standards to prioritize latency. To address this, service adaptation logic must incorporate security constraints as part of the decision-making process. For example, adaptation algorithms should ensure that minimum security baselines, such as encryption strength or authentication rigor, are always maintained, regardless of performance requirements. Policies defining these baselines should be embedded into the service description and enforced by the control plane.

The interface that allows applications to request resources or specify service parameters is one of the critical components of the system, but it is also a potential vulnerability. Without adequate safeguards, attackers could exploit the interface to inject malicious requests, overload the system, or extract sensitive information. Additionally, poorly secured interfaces may be vulnerable to Man-in-the-Middle (MitM) attacks for instance, where adversaries intercept and manipulate service requests. To mitigate these risks, the communication between applications and the system must use E2E encryption. Strong input validation mechanisms should be implemented to prevent injection attacks, ensuring that only well-formed, authorized requests are processed. Furthermore, audit logs should record all service requests and resource allocations, providing a trail for forensic analysis in case of incidents.

Overall, the integration of these service handling concepts necessitates a system architecture that balances flexibility and security risk. While no single solution can eliminate all risks, a multi-layered security approach can mitigate the most critical vulnerabilities. Specifically, key architectural implications include:

- Centralized Control with Distributed Enforcement: The system must centralize decision-making for service handling while delegating enforcement to distributed security agents at each interface, ensuring low latency and scalability.
- Policy-Driven Design: Security policies must be tightly integrated into the service description, defining tradeoffs between the risk situation, security resources needed and the performance requirements, guiding resource allocation and adaptation decisions.
- RT Monitoring and Threat Detection: Continuous telemetry from all interfaces is essential to detect and mitigate security threats dynamically.
- Resilience and Fail-Safe Mechanisms: The system architecture must include redundancy and fallback mechanisms to ensure uninterrupted service in the event of an attack or system failure.

9 Application of Service Description to Use Cases

This section demonstrates how the major findings from the previous parts of the document can be applied to the use cases outlined at the beginning. By mapping the use cases to the automation component model as described in Sections 3.2 and 4, readers will see how the findings can be effectively utilized. Additionally, relevant components of a service request for the scenarios in Section 2 will be discussed, detailing the value of the scenario, the necessary subservices, and the relevant parameters with respect to dependability (as discussed in Sections 5.3 and 7). This includes modes of operation, levels of operation, and the different values provided at these levels. The section will also explore which subservices might require adaptation and under what circumstances. By the end of this section, readers should have a basic understanding of how the theoretical concepts and findings can be practically applied to real-world use cases, enhancing the dependability and performance of their applications.

9.1 Extended Reality (XR)

This section discusses how the XR use case introduced in Section 2.1 can be represented using the OPC UA FX framework. We discuss what are the *AutomationComponents*, *Connections*, *Assets,* and *FunctionalEntities* identified in the use case description.

For the XR use case, we define the following *AutomationComponents*:

- Wearable XR Device
- Edge Server (for offloading of device functions)
- Digital Twin/Device

The C*onnections* between these *AutomationComponents* are:

- Wearable XR device – Edge server
  - Rendering function – rendering function (bi-directional)
  - Spatial compute function – spatial compute function (bi-directional)
  - Digital Twin/Device – spatial compute function (bi-directional)

- Digital Twin (or other device) – Edge server
  - Spatial compute function – spatial compute function (bi-directional)

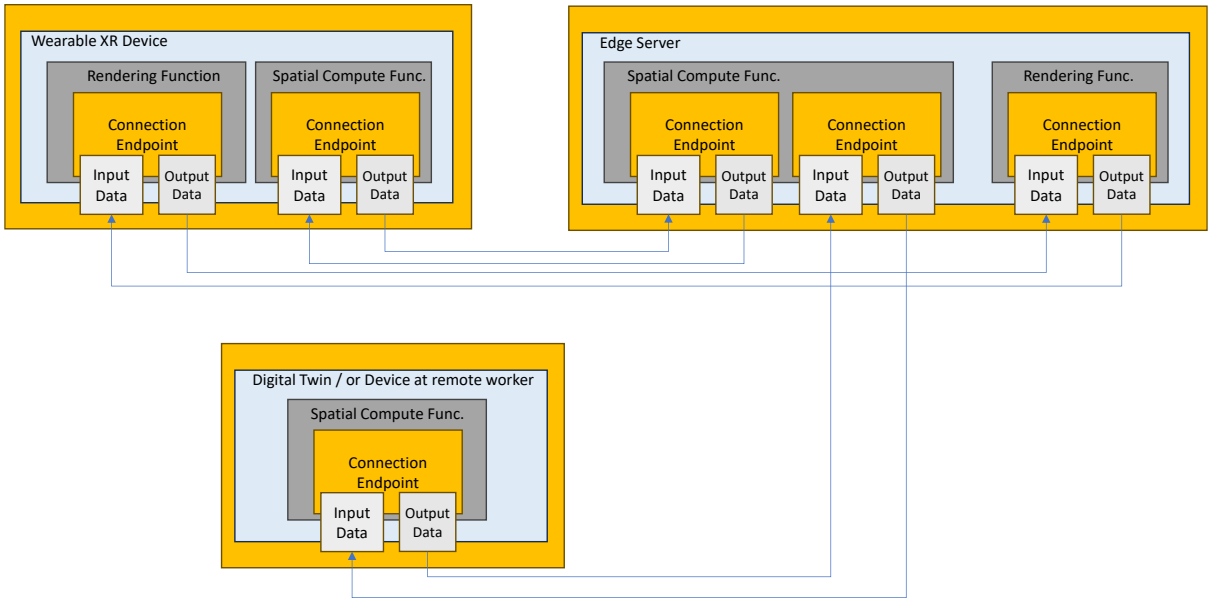

Figure 9.1: OPC UA FX AutomationComponent and Connections in the XR use case

Note that the Rendering Function and Spatial Compute Function are shown both in the Wearable XR Device and the Edge Server. The reason is that part of the functionality performed by these functions can be offloaded from the Wearable XR Device to the Edge Server.

As per the use case description, the Spatial Compute Function contains a set of sub-functions, e.g., related to localization. If these sub functions are executed both by the XR Device and the Edge Server the communication between those could be modelled using separate Connections, perhaps with different characteristics (QoS settings, etc.). However, for simplicity, all data exchange between two *FunctionalEntities* is modelled using a single *Connection*.

As per the use case description, the Wearable XR Device also contains a camera and sensors. However, as those do not directly exchange data with other *AutomationComponents*, but rather with the Spatial Compute Function within the same *AutomationComponent*, they are not modelled.

As mentioned in Section 2.1, the XR use case must exhibit on availability, regardless of the mode of operation. Therefore, all the subservices involved in the use case must exhibit availability too. From the communication subservice perspective, this implies using mechanisms to ensure that the communication among different *AutomationComponents* can exchange information successfully with a very high probability, which in networks with high bandwidth consumption may require mechanisms such as resource reservation, to ensure that the *Connections* between *FunctionalEntities* are available when needed.

From the computation point of view, achieving availability also requires proper resource reservation, in both the XR device but also the edge server. The computing resources to be reserved are CPU time,

memory, and storage. Furthermore, the execution of certain functions, such as the Spatial Compute Function, heavily rely on the synchronization subservice for operating correctly. This is because these functions can be executed in several automation components in parallel, so guaranteeing consistency is crucial. To that end, a global view of time is crucial to organize events in the correct order and ensure a correct operation. Thus, the synchronization subservice has reliability requirements, which can be satisfied with fault tolerance techniques, as well as accuracy and precision requirements, again, regardless of the mode of operation. The XR use case also requires the security subservice to protect sensitive data from being accessed by attackers. This subservice must also be present regardless of the mode of operation, as a breach in security could be catastrophic for the safety of the workers and could also have an important economic impact.

Nonetheless, when moving from the normal production mode of operation to the real-time fault search mode of operation (refer to Section 2.1 for details), the communication and computation subservices must also support RT operation. This is because the exchange of information between the operator and the digital twin in this mode must be made within time bounds to provide adequate support to the diagnosis of faults. In this mode of operation, the communication and computation subservices need to change their level of operation and move from a reservation-based approach for accessing resources, to approaches that can increase the guarantees and bound execution time and jitter, e.g., scheduling traffic and CPU access.

9.2 Exoskeleton in Industrial Context

In this section, it is shown how the aforementioned service description is applied to the exoskeleton use case. The use case example reported in Section 2.2 can be modelled using the OPC UA FX framework. Some *AutomationComponents*, *Connections* are identified, and their description is useful to highlighting the service and subservice requirements. This model can be updated and extended to any different exoskeleton use case scenarios by following the described guidelines.

The names of the *AutomationComponents* reflect the names of the corresponding elements in the use case description defined as:

- Exoskeleton
- Environmental Sensors
- Cameras
- Edge Server

For the exoskeleton use case the following communication *Connections* are defined where each communication endpoint contains a *FunctionalEntity* with a *ConnectionEndpoint*:

- Exoskleleton-Edge Server (bi-directional)
- Environmental Sensors-Edge Server (uni-directional)
- Camera-Edge Server (uni-directional)

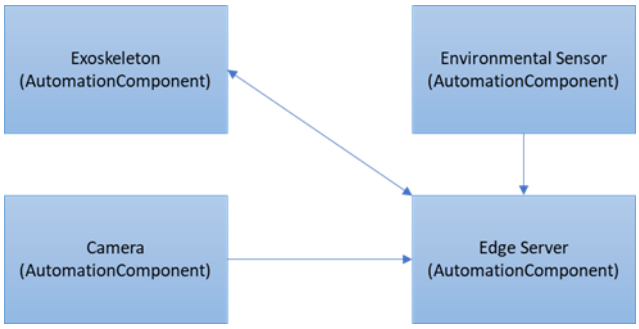

Figure 9.2: OPC UA FX AutomationComponent and Connections in the exoskeleton use case

The exoskeleton contains some on-board sensors and actuators and an on-board RT controller which are modelled as *Assets*. These *Assets* are responsible to provide assistance to the user while performing movement tasks and they are included in an embedded RT architecture modelled as Exo Actuators Control *FunctionalEntity*.

The Edge Cloud Server hosts the user task-movement detection and the implementation of a task-oriented assistive strategy, both of which are modeled as a compute *FunctionalEntity*. These entities are required by the exoskeleton to operate within a closed loop delocalized control architecture and must comply with RT execution and time synchronization. Additionally, the edge server's *FunctionalEntities* rely on synchronous data provided by the environmental sensors and the camera *FunctionalEntities* to enhance the overall exoskeleton control architecture.

The *FunctionalEntities* interact in order to impact the service adaptivity and to achieve the use case mode of operations and level of operations.

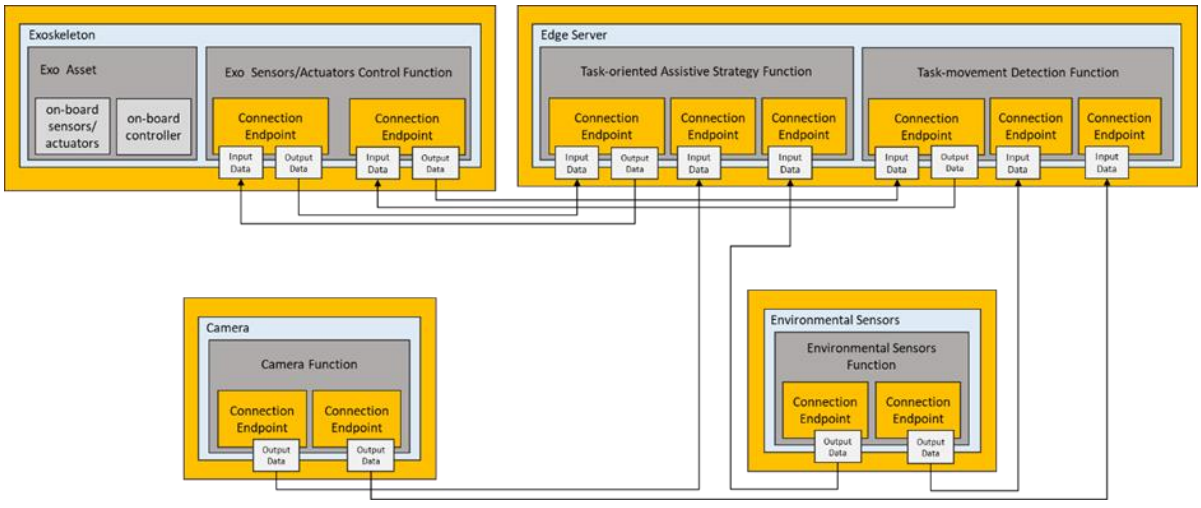

Figure 9.3: The figure shows Connections that are established between the different FunctionalEntities within each AutomationComponent. For bi-directional communications, the ConnectionEndpoints contain both Input Data and Output Data. For uni-directional communication only the input or the output data are reported.

The various modes of operation indicated for the exoskeleton use case example require a dependable communication service that must meet performance parameters, availability, and reliability. Regarding performance parameters such as latency and range of data rate, more detailed numerical considerations are reported in D1.1 [DET23-D11] and D1.2 [DET23-D12]. As explained in Section 2.2, to accommodate the unpredictability of human movement, the most stringent parameters must be maintained across all the modes of operation to avoid unsafe human-in-the-loop conditions. Despite the uniform communication parameters across different mode of operation, the level of operation must switch between Standard and Safe when the communication subservice fails to meet the requirements. Additionally, it is crucial to consider an adequate level of maintainability for the communication subservice.

The listed functional entities for the exoskeleton use case example require a computation service that must meet RT parameters and provide both volatile and persistent storage. Based on the specific implementation of the functional entity's algorithm, a computation resource equivalent to the required CPU capacity should be properly dimensioned, and the appropriate RT scheduler type selected. Refer to Deliverable D3.3 [DET23-D33] for additional numeric considerations.

The exoskeleton use case example involves various *AutomationComponents* that must be properly synchronized by a dedicated time synchronization subservice. Furthermore, the functional entities are part of a dependable hierarchical control architecture and require synchronized computation processes to meet the overall RT requirements for the modes of operation. The time synchronization subservice must ensure accuracy, precision, and an adequate fault-tolerance mechanism. Refer to D1.2 [DET23-D12] for further numerical considerations. Finally, an on-line diagnostic for the time synchronization subservice is also important to accurately handle the switch between Safe and Standard levels of operation.

The exoskeleton use case example also requires a Security Subservice to prevent breaches of sensitive use case data and ensure compliance with data privacy regulations in the relevant region. Considering the rapid growth of the exoskeleton market during the last ten years, along with evolving regulations related to data, data privacy and cyber security, it is essential to consider service adaptivity for security subservices. Refer to Sections 7.4.4 and 8 for detailed insights that directly benefit the use case.

9.3 Factory Automation: Adaptive Manufacturing

This section discusses how the adaptive manufacturing use case introduced in Section 2.3 can be represented using the OPC UA FX framework. We discuss what are the *AutomationComponents*, *Connections*, *Assets,* and *FunctionalEntities* identified in the use case description.

The *AutomationComponents* identified in the example described in this document are:

- Mobile processing module.
- Static processing module (SPM).
- Production line management entity.

The C*onnections* between these *AutomationComponents* are:

- MPM – Production line management entity (bi-directional).
- SPM – Production line management entity (bi-directional).

Note that these connections are only the ones identified in the use case example presented in Section 2.3, but it is not limited to just these two connections. Also, note that each *Connection* end point contains a *FunctionalEntity* with a *ConnectionEndpoint*.

The figure below shows the Connections that are established between the different *FunctionalEntities* within each *AutomationComponent*. For bi-directional Connections, the *ConnectionEndpoints* contain both Input Data and Output Data parts.

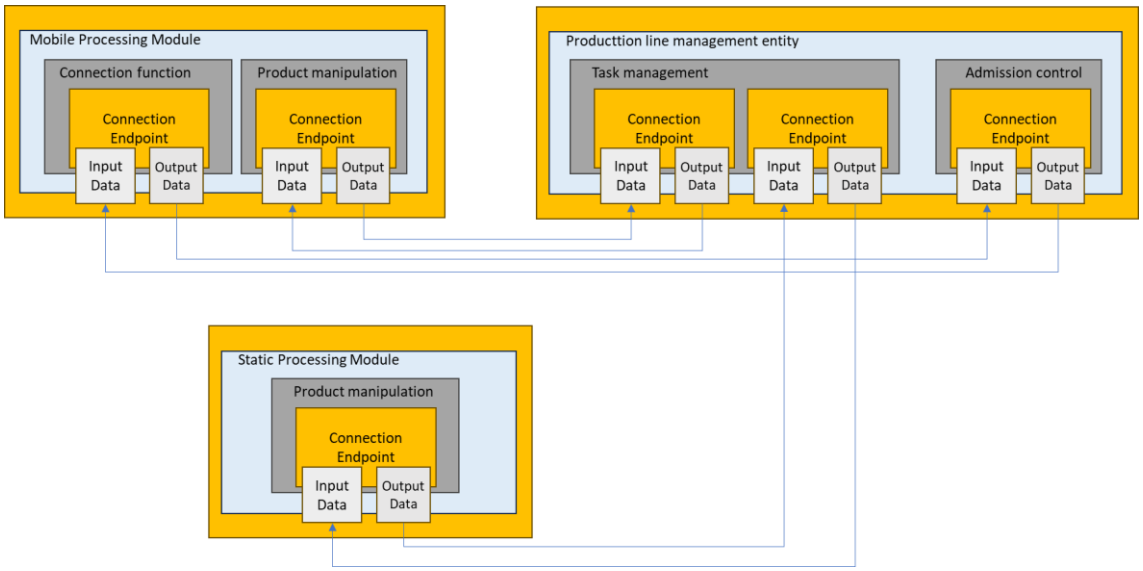

Figure 9.4: OPC UA FX *AutomationComponent* and *Connections* in the Adaptive Manufacturing use case

Both, MPMs and SPMs have sensors and actuators that they use to carry out their regular operations. Examples of sensor could be a camera that allows to capture images of the objects to be manipulated or speed sensors; while examples of actuators could be motors, or the tools used to manipulate the products. All sensors and actuators are *Assets* of the *AutomationComponent* they are embedded in. Furthermore, as discussed in Section 4, the applications can be represented using *FunctionalEntities*. Some examples of these are the tasks executed in the MPMs and SPMs, e.g., move towards the production line, grab a product, screw a new component, etc.; but another example of *FunctionalEntity* can also be the safety application executed in a distributed manner in the system.

All the modes of operation described in Section 2.3 have dependability requirements, therefore, the subservices used must also exhibit such dependability. This is because for an overall service to exhibit any dependability attribute all its subservices must also exhibit such attribute.

From the communication perspective, we can highlight the need for reliability, safety, and security throughout the whole operation of the system, regardless of the mode of operation being used at each specific moment. This is so as an interruption (or undesired behavior) of the communication subservice could lead to catastrophic consequences. Furthermore, the communication also has timing requirements, and thus the communication subservice must count with mechanisms to guarantee bounded transmission times and low jitter. These attributes can be achieved using already existing

techniques for providing the *Connections* with redundancy, e.g., FRER, and timing guarantees, e.g., traffic prioritization.

The computation subservice must also exhibit the mentioned dependability attributes, namely reliability, safety, and security, regardless of the selected mode of operation. This can be achieved replicating *FunctionalEntities* or enabling mechanisms to trigger the creation of a new *FunctionalEntity* in case of failure. This requires allocating CPU, memory, and storage, but also proper scheduling techniques to guarantee the timely execution of all the *FunctionalEntities* that compete for resources.

Section 2.3 also describes the importance of adequate time synchronization for the correct operation of the production line and the coordination of MPMs and SPMs. This is because the tasks carried out by the MPMs and SPMs must happen in the correct order to build the product correctly. Thus, the synchronization subservice must provide adequate accuracy and precision. Furthermore, since the communication and computation subservices rely on the synchronization subservice to provide the required value, the synchronization service should also exhibit reliability, safety, and security, by means of specific mechanisms, such as fault tolerance for the clock master, or authentication.

Even though the dependability exhibited by the different subservices must be the same in both modes of operation described in Section 2.3, the level of operation must be adapted. When compared to MO1, MO2 requires extra MPMs, which means extra *AutomationComponents,* with its *Assets, Connections,* and *FunctionalEntities.* Thus, the subservices must allocate the required resources, e.g., bandwidth, CPU, memory. Furthermore, as described in Section 2.3, the synchronization subservice must enable the change in synchronization domain from all devices, so they can synchronize their operation to the new MPM which counts with a different time synchronization protocol.

Finally, the security subservice is key to protect sensitive data related to the use case from being accessed by unwanted parties. This can imply implementing authentication and encryption algorithms to avoid unknown devices from accessing sensitive information, that if filtered could have serious economic and legal consequences.

9.4 Mobile Automation: Smart Farming

In this section it is shown how the Smart Farming use case can be modelled using the OPC UA FX framework, using *AutomationComponents and Connections*. The names of the *AutomationComponents* reflect the names of the corresponding elements in the use case description in Section 2.4.

For the Smart Farming use case, we define the following *AutomationComponents*:

- Harvester
- Trolley
- Drone
- Edge Server

For the Smart Farming use case, we define the following communication *Connections,* where each *Connection* endpoint contains a *FunctionalEntity* with a *ConnectionEndpoint*:

- Harvester-Trolley (bi-directional)
- Harvester-Edge Server (bi-directional)
- Drone-Edge Server (bi-directional)

The *FunctionalEntities* identified in this use case are:

- Reconfiguration function (drone)
- Camera function (drone)
- Safety function (edge server)
- Farming function (harvester and trolley)

The figure below shows the Connections that are established between the different *FunctionalEntities* within each *AutomationComponent*. For bi-directional Connections, the *ConnectionEndpoints* contain both Input Data and Output Data parts. For uni-directional *Connections*, one ConnectionEndpoint associated with the Connection contain an Output Data part, and the other ConnectionEndpoint contain an Input Data part.

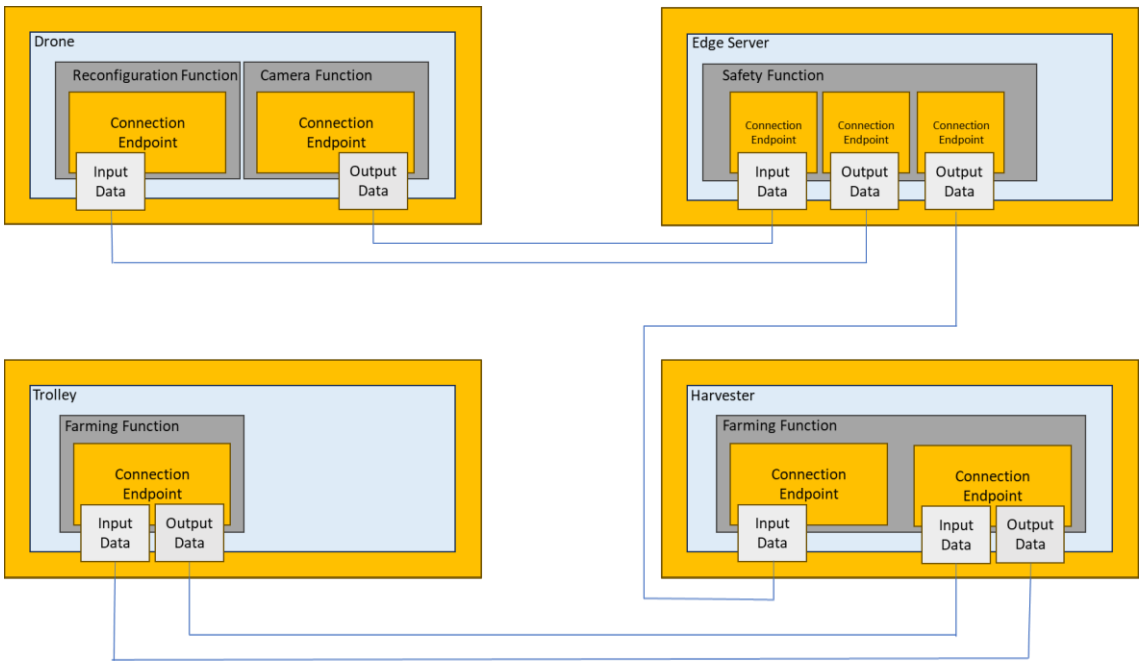

Figure 9.5: OPC UA FX *AutomationComponent* and *Connections* in the Smart Farming use case

As described in Section 2.4, the drone is equipped with a camera that captures images of the surroundings to enable obstacle detection by the ground vehicles (harvester and trolley). Nonetheless, since image processing is a computationally and resource-intensive task, the drone offloads this task to the edge server, which carries out obstacle detection as part of the safety function. Then the server makes safety decisions and relays them to the ground vehicles. All the communications are wireless, and must exhibit reliability, as the safe operation of the system depends on it. The communication must also be safe, secure, and must provide timing guarantees, such as bounded latency. These attributes can be achieved by means of fault tolerance, e.g., retransmissions, encryption of sensitive data, and resource reservation.

From the computing subservice point of view, the different *FunctionalEntities* have diverse needs. For example, the camera and the farming functions requires CPU time and memory, while the safety function also requires storage, to safe historical data basic to make safety-related decisions. Also, the safety function requires reliability, as its malfunction could lead to catastrophic consequences. On top of that, all functions rely on a global common view of time, so events can be ordered to ensure correct operation of the system.

The scenario described in Section 2.4 shows the need for adaptive subservices, as when an obstacle is detected, the drone increases the frequency with which it captures images. This generates a higher bandwidth consumption from the communication perspective, but also more computation resources, in terms of CPU time and memory, to ensure that the obstacle is properly monitored and, in that way, avoid accidents. Finally, the security subservice is also vital when there are changes in the mode of operation. For example, the subservice needs to provide means for authentication of *FunctionalEntities*, as new trolleys and drones may incorporate to the harvesting operation in runtime to allow for a fully loaded trolley to unload, or for a low-battery drone to charge.

10 Conclusion

This document has explored the essential components of dependable service design in 6G networks, focusing on the core services of communication, computation, time synchronization, and security. These services are indispensable in supporting a wide range of dynamic and time-sensitive applications, particularly in industrial contexts where adaptivity and reliability are paramount. By presenting models and methodologies for service descriptions, this work underscores the importance of structured approaches to manage the complex interplay between applications and the underlying 6G infrastructure.

Dependable services, as discussed, are dynamic constructs that must adapt to evolving conditions. Service descriptions provide the foundation for defining, managing, and transitioning between operational modes and levels of service. For instance, transitioning seamlessly between high-performance and safe modes in exoskeletons or adapting synchronization precision in adaptive manufacturing systems illustrates how these descriptions enable operational flexibility. This adaptivity ensures that critical applications continue to function reliably, even under challenging or changing conditions.

Furthermore, service descriptions play a pivotal role in fostering interoperability across the 6G ecosystem. By providing a standardized language for specifying performance requirements and system capabilities, they facilitate the integration of new applications, hardware, and network components. This capability is particularly valuable in industrial environments, where legacy systems must coexist and interact with state-of-the-art technologies.

Beyond their technical utility, structured service descriptions and value focused service requests also contribute to broader societal and environmental goals. They enable applications to optimize resource usage, prioritize sustainability metrics, and maintain efficiency, aligning the operation of 6G networks with the pressing need for greener and more sustainable practices. In parallel, robust security services reinforce trust in critical applications, safeguarding infrastructure and data against evolving threats.

A key concept explored in this document is the ability of applications to articulate their value in service requests. By prioritizing applications based on their value, resource assignment and scheduling can be optimized, ensuring that critical functions receive the resources they need to operate effectively. Moreover, the inclusion of multiple levels of service within requests allows the DETERMINISTIC6G system to dynamically adjust to RT conditions, optimizing system performance by seamlessly balancing trade-offs between competing applications and available resources.

In conclusion, the structured approach to service descriptions and service requests presented in this document serves as a cornerstone for achieving dependable and adaptable 6G networks. It empowers applications to clearly define their requirements and enables the DETERMINISTIC6G system to dynamically respond, fostering a synergy between application needs and infrastructure capabilities. This approach not only ensures reliability and flexibility but also drives innovation, operational efficiency, and societal benefits. By adopting these methodologies, stakeholders can unlock the full transformative potential of 6G technologies, paving the way for a new era of connectivity, dependability, and sustainability.

Appendix

Reference

[3GPP23-22104]	Third Generation Partnership Project (3GPP), TS22.104, "Service requirements for cyber-physical control applications in vertical domains"
[3GPP23-22261]	Third Generation Partnership Project (3GPP), TS22.261, "Service requirements for the 5G system"
[5GA21a]	5G-ACIA, *Service-Level Specifications (SLSs) for 5G Technology Enabled Connected Industries,* white paper, September 2021 https://5g-acia.org/whitepapers/service-level-specifications-slss-for-5g-technology-enabled-connected-industries/
[5GA21b]	5G Alliance for Connected Industries and Automation, "Integration of 5G with Time-Sensitive Networking for Industrial Communications," white paper, February 2021, https://5g-acia.org/whitepapers/integration-of-5g-with-time-sensitive-networking-for-industrial-communications/
[5GA23]	5G Alliance for Connected Industries and Automation, 5G-ACIA, www.5g-acia.org
[ABB+22]	Cecilia Andersson, Jonas Bengtsson, Greger Byström, Pål Frenger, Ylva Jading, My Nordenström, "Improving energy performance in 5G networks and beyond," in Ericsson Technology Review, vol. 2022, no. 8, pp. 2-11, August 2022, doi: 10.23919/ETR.2022.9911220, https://ieeexplore.ieee.org/document/9911220
[DET23-D11]	DETERMINISTIC6G, Deliverable 1.1, "DETERMINISTIC6G use cases and architecture principles," Jun. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D12]	DETERMINISTIC6G, Deliverable 1.2, "First report on DETERMINISTIC6G architecture" Apr. 2024, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D21]	DETERMINISTIC6G, Deliverable 2.1, "First report on 6G centric enabler", Dec. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D22]	DETERMINISTIC6G, Deliverable 2.2, "First Report on the time synchronization for E2E time awareness", Dec. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D33]	DETERMINISTIC6G, Deliverable 3.3, "Report on Deterministic edge computing and situational awareness via digital twining security solution" Jun. 2024, https://deterministic6g.eu/index.php/library-m/deliverables
[DET24-D34]	DETERMINISTIC6G, Deliverable 3.4, "Optimized deterministic end-to-end schedules for dynamic systems", Jun. 2024, https://deterministic6g.eu/index.php/library-m/deliverables
[Eri22]	Ericsson, "What's next for 5G spectrum utilization?", technical report, 2022, https://www.ericsson.com/en/reports-and-papers/further-insights/traffic-steering-for-5g-spectrum-utilization
[FJN24]	Pål Frenger, Ylva Jading, Ali Nader, "Energy performance of 6G Radio Access Networks: A once in a decade opportunity," Ericsson white paper, November 2024, https://www.ericsson.com/en/reports-and-papers/white-papers/energy-performance-of-6g-ran

[GMH+23]	Leefke Grosjean, Alan Minney, Chris Halton, Sapideh Matinfar, Valentin Tudor, Anders Erlandsson, Patrik Hedlund, Senthamiz Selvi A, Erik Simonsson, Joachim Sachs, ” Human-centric manufacturing - Industry 5.0 and the factory of the future,”, article, Ericsson Imagine Possible Perspectives, 2023, https://www.ericsson.com/en/about-us/new-world-of-possibilities/imagine-possible-perspectives/industry5-0-human-centric-manufacturing/
[GR03]	Jana van Greunen and Jan Rabaey. 2003. Lightweight time synchronization for sensor networks. In Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (‘SNA '03).
[Gra23]	T. Graves, (2023). *Basics – Service-Oriented Architecture. In: The Service-Oriented Enterprise*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-9189-4_2
[IEC09-61907]	International Electrotechnical Commission IEC, Communication network dependability engineering, IEC 61907:2009.
[IEC23-1920122]	International Electrotechnical Commission IEC, Electropedia: The World's Online Electrotechnical Vocabulary, IEV 192-01-22 (dependability), https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-01-22
[IEC23-1920123]	International Electrotechnical Commission IEC, Electropedia: The World's Online Electrotechnical Vocabulary, IEV 192-01-23 (availability), https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-01-23
[IEC23-1920124]	International Electrotechnical Commission IEC, Electropedia: The World's Online Electrotechnical Vocabulary, IEV 192-01-24 (reliability), https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-01-24
[IEC23-1920127]	International Electrotechnical Commission IEC, Electropedia: The World's Online Electrotechnical Vocabulary, IEV 192-01-27 (maintainability), https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=192-01-27
[IEC24]	IEC/IEEE 60802, “TSN Profile for Industrial Automation,” Draft 3.0, August 2024, https://1.ieee802.org/tsn/iec-ieee-60802/
[IEEE1588]	IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008) , vol., no., pp.1-499, 16 June 2020, doi: 10.1109/IEEESTD.2020.9120376.
[IEEE20-802.1AS]	IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications, in IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011) , vol., no., pp.1-421, 19 June 2020, doi: 10.1109/IEEESTD.2020.9121845.
[IEEE24-802.1ASdm]	IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications Amendment 3: Hot Standby and Clock Drift Error Reduction, in IEEE Std 802.1ASdm-2024 (Amendment to IEEE Std 802.1AS-2020 as amended by IEEE Std 802.1AS-2020/Cor 1-2021, IEEE Std 802.1ASdr-2024, and IEEE Std 802.1ASdn-2024) , vol., no., pp.1-164, 4 Oct. 2024, doi: 10.1109/IEEESTD.2024.10707145.
[IIC19]	Industrial Internet Consortium (IIC), “Time Sensitive Networks for Flexible Manufacturing Testbed Characterization and Mapping of Converged Traffic Types,” technical report, March 2019.
[IR17]	Industrial Radio, Aspects for Dependability – Assessment in ZDKI, 2017, www.industrial-radio.de

[KK22]	Amareet Kaur, Björn Karlberg, “The power of 5G Carrier Aggregation,” blog, February 2022, https://www.ericsson.com/en/blog/2022/2/the-power-of-5g-carrier-aggregation
[KO87]	H. Kopetz and W. Ochsenreiter, “Clock synchronization in distributed real-time systems”, IEEE Trans. on Comput., vol. 100, no. 8, pp. 933– 940, 1987.
[Kow06]	Kowalkowski, C. (2006*). Enhancing the Industrial Service Offering: New Requirements on Content and Processes* (Licentiate dissertation, Ekonomiska institutionen). https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-7215
[LSW+22]	J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, T. Wuest, D. Mourtzis, L. Wang, „Industry 5.0: Prospect and retrospect”, Journal of Manufacturing Systems, vol. 65, pp. 279-295, October 2022, DOI: https://doi.org/10.1016/j.jmsy.2022.09.017
[LZZ+13]	Li, Z., Zhong, Z. L., Zhu, W. C., & Qin, B. Y. (2013). A hardware time stamping method for PTP messages based on Linux system. TELKOMNIKA Indonesian Journal of Electrical Engineering, 11(9), 5105-5111.
[OASIS13]	OASIS, “Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0”, Jan. 2013, https://docs.oasis-open.org/tosca/tosca-primer/v1.0/tosca-primer-v1.0.pdf
[OFH+23]	Carin Omurcali, Linnea Faxén, Anders Helsing, Tao Cui, Tomas Köhler, “Advanced Traffic Steering in 5G: Why yo’ can't live without it,” blog, May 2023, https://www.ericsson.com/en/blog/2023/5/advanced-traffic-steering-in-5g-standalone
[OG09]	The Open Group, “SOA Source Book”, Van Haren Publishing, 2009, https://collaboration.opengroup.org/projects/soa-book/pages.php?action=show&ggid=1314
[OPC-10000-1]	*UA Part 1: Overview and Concepts*, November 2022, https://reference.opcfoundation.org/Core/Part1/v105/docs/
[OPC-10000-81]	*UAFX Part 81: Connecting Devices and Information Model*, June 2024, https://reference.opcfoundation.org/UAFX/Part81/v100/docs/
[PGK+24]	M. Paiola, R. Grandinetti, C. Kowalkowski, M. Rapaccini, *Digital servitization strategies and business model innovation: The role of knowledge-intensive business services*, Journal of Engineering and Technology Management, Volume 74, 2024, 101846, ISSN 0923-4748, https://doi.org/10.1016/j.jengtecman.2024.101846.
[RFC5905]	RFC: 5905 – “Network Time Protocol Version 4: Protocol and Algorithms Specification”, 2010, https://datatracker.ietf.org/doc/html/rfc5905
[SDL+21]	Schüngel, M., Dietrich, S., Leurs, L., Ginthör, D., Chen, S. P., & Kuhn, M. (2021, March). Advanced grandmaster selection method for converged wired and wireless networks. In 2021 22nd IEEE International Conference on Industrial Technology (ICIT) (Vol. 1, pp. 1007-1014). IEEE.
[SW04]	Dolev, Shlomi, and Jennifer L. Welch. "Self-stabilizing clock synchronization in the presence of byzantine faults." Journal of the ACM (JACM) 51.5 (2004): 780-799.
[XYS+11]	Xueqiao, L., Yuan, C., Shuang, L., & Yaxing, D. (2011, May). Implementation and research of hardware time stamping techniques based on IEEE1588. In 2011 I^{EE}E 3rd International Conference on Communication Software and Networks (pp. 6-9). IEEE.

List of abbreviations

Table 5: List of abbreviations

3Cs	Communication-Compute-Control
BC	Boundary Clock
BFT	Byzantine Fault Tolerant
BTCA	Best TimeTransmitter Clock Algorithm
CNC	Centralized Network Controller
CPS	Cyber-Physical System
CUC	Centralized User Controller
DetNet	Deterministic Networking
DoS	Denial-of-Service
DPU	Data Processing Unit
E2E	End-to-End
FPGA	Field Programmable Gate Array
FRER	Frame Replication and Elimination for Reliability
GMC	Grand Master Clock
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Units
IA	Industrial Automation
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
IOPS	Input and Output operations Per Second
IPS	Intrusion Prevention System
IPU	Infrastructure Processing Unit
KPI	Key Performance Indicator
KVI	Key societal Value Indicators
LLDP	Link Layer Discovery Protocol
MitM	Man-in-the-Middle
MO	Mode of Operation
MPM	Mobile Processing Module
NETCONF	Network Configuration Protocol
NIC	Network Interface Card
NO-RT	Non-Real-Time
NTP	Network Time Protocol
OC	Ordinary Clock
OE	Occupational Exoskeleton
OPCF	OPC Foundation
OPC UA	OPC Unified Architecture
PLC	Programmable Logic Controller
PL	Production Line
PTP	Precision Time Protocol

QoS	Quality of Service
RAM	Random Access Memory
RT	Real-Time
RTC	Real-Time Computing
SLA	Service-Level Agreement
SLS	Service-Level Specification
SOA	Service-Oriented Architectures
SPM	Static Processing Module
SSH	Secure SHell
TAI	Temps Atomique International or International Atomic Time
TLS	Transport Layer Security
TOSCA	Topology and Orchestration for Cloud Applications
TSN	Time-Sensitive Networking
TSN-AF	TSN Application Function
TTD	Time-to-Deploy
UE	User Equipment
UGV	Unmanned Ground Vehicle
UTC	Coordinated Universal Time
vCPU	Virtual CPU
VM	Virtual Machine
WP	Work Package
XR	eXtended Reality