DETERMINISTIC6G

Second report on deterministic edge computing and situational awareness via digital twinning

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

Second report on deterministic edge computing and situational awareness via digital twinning

Grant agreement number: Project title: Project acronym: Project website: Programme:	101096504 Deterministic E2E communication with 6G DETERMINISTIC6G Deterministic6g.eu EU JU SNS Phase 1
Deliverable type:	Report
Deliverable reference number:	D3.6
Contributing workpackages:	WP3
Dissemination level:	PUBLIC
Due date:	M28
Actual submission date:	29.Apr.2025
Responsible organization:	ETH
Editor:	Dávid Jocha
Version number:	V1.0
Status:	Final
Short abstract:	This deliverable expands the architecture for edge computing and digital twinning, emphasizing the advantages of Multi-access Edge Computing (MEC) over traditional cloud setups, especially in reducing delay variations, and integrating edge domains with Time-Sensitive Networking (TSN). It addresses challenges in time-sensitive application orchestration with a new packet scheduling enabler using extended Berkeley Packet Filter (eBPF) in Kubernetes environments and proposes a cloud (host) abstraction solution to ensure seamless integration of a cloud deployment into an end-to-end TSN system to enhance control plane integration and traffic handling. It explores both standalone and operator-enabled edge scenarios for hosting applications, considering 3GPP Edge Computing. In digital twinning, the report analyzes Digital Twin (DT) architectures in Operational Technology (OT) and network domains, focusing on 6G DT, and proposes a 6G DT Application function, exploring data provisioning and interaction architectures. It recommends using the 3GPP data analytics framework for 6G DT data provisioning and presents a call flow cycle for a use case of interaction between OT and DT systems via DTs for joint optimization.
Keywords:	Edge computing, MEC, TSN, 6G, Digital twin, Situational awareness

Contributors:	Armin Hadziaganovic (SAL) Damir Hamidovic (SAL) Dávid Jocha (ETH) Drissa Houatra (OR) Huu Nghia Nguyen (MI) János Harmatos (ETH) Joachim Sachs (EDD) Jose Costa Requena (CMC)

Reviewers:	Edgardo Montes de Oca (MI) Francesco Giovacchini (IUVO) James Gross (KTH)

Revision History

14.3.2025	Draft version for work package internal review
20.3.2025	Draft version for project internal review
8.4.2025	PMT review draft
29.4.2025	Final submission version

Disclaimer

This work has been performed in the framework of the Horizon Europe project DETERMINISTIC6G co-funded by the EU. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein. This deliverable has been submitted to the EU commission, but it has not been reviewed and it has not been accepted by the EU commission yet.

Executive summary

This document is the second report on Edge computing and situational awareness via digital twinning. It extends the architecture described in the first report [DET24-D33] and provides additional details.

Key contributions related to Edge computing:

- The integration of the edge domain with a TSN system is detailed, including necessary extensions to the 3GPP MEC architecture, utilizing a TSN Application Function (TSN AF) as a virtual bridge for TSN controllers.
- Challenges in orchestrating time-sensitive applications in Kubernetes environments are addressed with a new time-aware packet scheduling enabler using eBPF for seamless TSN support. The document emphasizes the importance of integrating user and control planes between legacy TSN domains and compute domains, noting the current lack of standardized methods for exposing cloud ecosystem details to TSN control planes.
- An abstraction of the cloud ecosystem is proposed to model cloud host deployments to ensure compatibility with IEEE specified TSN control plane. This TSN-aware cloud management entity aids in integrating control planes and configuring cloud-specific traffic handling solutions based on 802.1Qbv scheduling plans.
- The document explores both standalone and operator-enabled edge scenarios for hosting cloudified applications, outlining processes for application deployment, service initiation, edge capability exposure, and service configuration.
- The design of a dependable deterministic Software-Defined Network (SDN) controller approach in the edge is considered, focusing on reliability and real-time constraints.

Key contributions related to situational awareness via digital twinning:

- A comprehensive state-of-the-art analysis has been done on architecture aspects of OT domain DT and network domain DT with the focus on 6G DT. The analysis covered Asset Administrative Shell (AAS) standardization, architecture and data provisioning as well as network DT architectural recommendations and advancements.
- A maturity gap between the usage and advancements of DT technology in OT and network domain has been discussed.
- 6G DT Application function has been proposed as a placement of DT in 6G system. Additionally, two different architectures of 6G DT AF were proposed and analyzed with aspect to data provisioning and interaction with DT applications.
- Using 3GPP data analytics framework as data provisioning solution for 6G DT was proposed together with exploration of alternative solutions.
- A full call flow cycle has been done for a use case from the previous deliverable [DET24-D33] based on proposed DT AF architectures.

List of Figures

List of Tables

1 Introduction

The progress in digital transformation across industries and society, along with the rise of high-demand applications, imposes unique requirements on the network infrastructure to support end-to-end dependable time-critical communication services. An end-to-end communication service encompasses the integration of wired, wireless, and computing domains along with their interaction with the digital representation of physical entities such as robots, machines, sensors, and devices. Consequently, it is essential to shift control, monitoring, and maintenance functions towards an edge computing paradigm in a distributed manner to enable flexibility while offering computation and storage capabilities. Although existing edge computing systems can host applications requiring substantial computation, improvements are necessary to guarantee dependable, time-critical, and deterministic communication performance. The challenge of enabling end-to-end deterministic communication services can be addressed by incorporating features of Time Sensitive Networking (TSN) and Deterministic Networking (DetNet) into the edge computing domain. Furthermore, predicting the behavior of physical systems and processes while generating various alternative scenarios can further enhance the end-to-end time-aware communication system. This second report details data plane and control plane aspects of the architectural framework and mechanisms for achieving dependable and deterministic communication within the edge computing domain and offers aspects of obtaining Situational Awareness (SA) from 6G and operational digital twins, which can be utilized to manage and configure 6G deterministic communication more effectively.

1.1 DETERMINISTIC6G Approach

Digital transformation of industries and society is resulting in the emergence of a larger family of time-critical services with needs for high availability and which present unique requirements distinct from traditional Internet applications like video streaming or web browsing. Time-critical services are already known in industrial automation; for example, an industrial control application that might require an end-to-end “over the loop” (i.e., from the sensor to the controller back to the actuator) latency of 2 ms and with a communication service requirement of 99.9999% [3GPP23-22261]. But with the increasing digitalization similar requirements are appearing in a growing number of new application domains, such as extended reality, autonomous vehicles, and adaptive manufacturing. The general long-term trend of digitalization leads towards a Cyber-Physical Continuum where the monitoring, control, and maintenance functionality is moved from physical objects (like a robot, a machine, or a tablet device) to a compute platform at some other location, where a digital representation – or digital twin – of the object is operated. Such Cyber Physical System (CPS) applications need a frequent and consistent information exchange between the digital and physical twins. Several technological developments in the ICT-sector drive this transition. The proliferation of (edge-) cloud compute solutions provides new cost-efficient and scalable computing capabilities that are often more efficient to maintain and evolve compared to embedded compute solutions integrated into the physical objects. It also enables the creation of digital twins as a tool for advanced monitoring, prediction, and automation of system components and improved coordination of systems of systems. New techniques based on Machine Learning can be applied in application design, which can operate over large data sets and profit from scalable compute infrastructure. Offloading compute functionality can also reduce spatial footprint, weight, cost, and energy consumption of physical objects, which is in particular important for mobile components, like vehicles, mobile robots, or wearable devices. This approach leads to an increasing need for communication between physical and digital objects, and this communication can span over multiple communication and computational domains.

Communication in this cyber-physical world often includes closed-loop control interactions which can have stringent end-to-end Key Performance Indicators (KPIs) (e.g., minimum and maximum packet delay) requirements over the entire loop. In addition, many operations may have high criticality, such as business-critical tasks or even safety relevant operations. Therefore, it is required to provide dependable time-critical communication which provides communication service-assurance to achieve the agreed service requirements.

Time-critical communication has mainly been prevalent in industrial automation scenarios with special compute hardware like Programmable Logic Controllers (PLC), and based on proprietary, mutually incompatible wired communication technologies, such as Powerlink and EtherCat, which is limited to local and isolated network domains and configured for a specific purpose of the local applications. With the standardization of TSN and DetNet, similar capabilities are being introduced into the Ethernet and IP networking technologies, which thereby provide a converged multi-service network enabling time-critical applications in a managed network infrastructure allowing for consistent performance with zero packet loss and guaranteed low and bounded latency. The underlying principles are that the network elements (i.e., bridges or routers) and the PLCs can provide a consistent and known performance with negligible stochastic variation, which allows to manage the network configuration to the needs of time-critical applications with known traffic characteristics and requirements. Furthermore, using interchangeable TSN hardware components has economic benefits, avoids vendor lock-in, and enables third-party support for configuration and troubleshooting.

It turns out that several elements in the digitalization journey introduce characteristics that deviate from the assumptions that are considered as baseline in the planning of deterministic networks. There is often an assumption that for compute and communication elements, and also for applications, any stochastic behavior can be minimized such that the time characteristics of the element can be clearly associated with tight minimum/maximum bounds. Cloud computing provides efficient scalable compute service, but introduces uncertainty in execution times; wireless communication provides flexibility and simplicity, but with inherently stochastic components that lead to packet delay variations exceeding significantly those found in wired counterparts; and applications embrace novel technologies (e.g., Machine Learning (ML)-based or machine-vision-based control) where the traffic characteristics deviate from the strictly deterministic behavior of traditional control. In addition, there will be an increase in dynamic behavior where characteristics of applications and network or compute elements may change over time in contrast to static behavior, which does not change during runtime. It turns out that these deviations of stochastic characteristics make traditional approaches to planning and configuration of end-to-end time-critical communication networks such as TSN or DetNet fall short in their performance regarding service performance, scalability, and efficiency. Instead, a revolutionary approach to the design, planning, and operation of time-critical networks is needed that fully embraces the variability but also dynamic changes that come at the side of introducing wireless connectivity, cloud compute, and application innovation. DETERMINISTIC6G has as objective to address these challenges, including the planning of end-to-end resource allocation for diverse time-critical services over multiple domains, providing efficient resource usage and a scalable solution [SPS+23].

DETERMINISTIC6G takes a novel approach towards converged future infrastructures for scalable cyber-physical systems deployment. With respect to networked infrastructures, DETERMINISTIC6G advocates (I) the acceptance and integration of stochastic elements (like wireless links and

computational elements) with respect to their stochastic behavior captured through either short-term or longer-term envelopes. Monitoring and prediction of KPIs, for instance latency or reliability, can be leveraged to make individual elements plannable despite a remaining stochastic variance. Nevertheless, system enhancements to mitigate stochastic variances in communication and compute elements are also developed. (II) Next, DETERMINISTIC6G attempts the management of the entire end-to-end interaction loop (e.g., the control loop) with the underlying stochastic characteristics, especially embracing the integration of compute elements. (III) Finally, due to unavoidable stochastic degradations of individual elements, DETERMINISTIC6G advocates allowing for adaptation between applications running over such converged and managed network infrastructures. The idea is to introduce flexibility in the application operation such that its requirements can be adjusted at runtime based on prevailing system conditions. This encompasses a larger set of application requirements that (a) can also accept stochastic end-to-end KPIs, and (b) that possibly can adapt end-to-end KPI requirements at run-time in harmonization with the networked infrastructure. DETERMINISTIC6G builds on a notion of time-awareness, by ensuring accurate and reliable time synchronicity while also ensuring security-by-design for such dependable time-critical communications. Generally, a notion of deterministic communication (where all behavior of network and compute nodes and applications is pre-determined) is extended towards dependable time-critical communication, where the focus is on ensuring that the communication (and compute) characteristics are managed in order to provide the KPIs and reliability levels that are required by the application. DETERMINISTIC6G facilitates architectures and algorithms for scalable and converged future network infrastructures that enable dependable time-critical communication end-to-end, across domains and including 6G.

1.2 Relation to other work packages

This deliverable is part of Work Package 3 and has linkages with other technical work packages, as presented in Figure 1.1. The deliverable takes inputs in terms of edge computing and Situational Awareness (SA) using digital twinning from the use case requirements and the DETERMINISTIC6G architecture investigated in WP1, as formulated in deliverables [DET23-D11] and [DET24-D12]. The deliverable [DET23-D21] on 6G centric enablers from WP2 discusses architectural aspects of latency prediction, which are used as basis on identifying gaps to produce accurate delay predictions for SA. The end-to-end scheduling aspects discussed in deliverable [DET23-D31] provides inputs to the traffic scheduling mechanisms in integrating the edge compute domain. The results of deliverable [DET24-D33], specifically the integration aspects of edge computing with TSN for scheduling, serves as an input to WP4 in developing the validation framework. [DET25-D35] presents architecture and scheduling for multi-domain scenarios. This deliverable document is a direct continuation and extension of the deliverable [DET24-D33], where deterministic edge computing and situational awareness aspects were introduced.

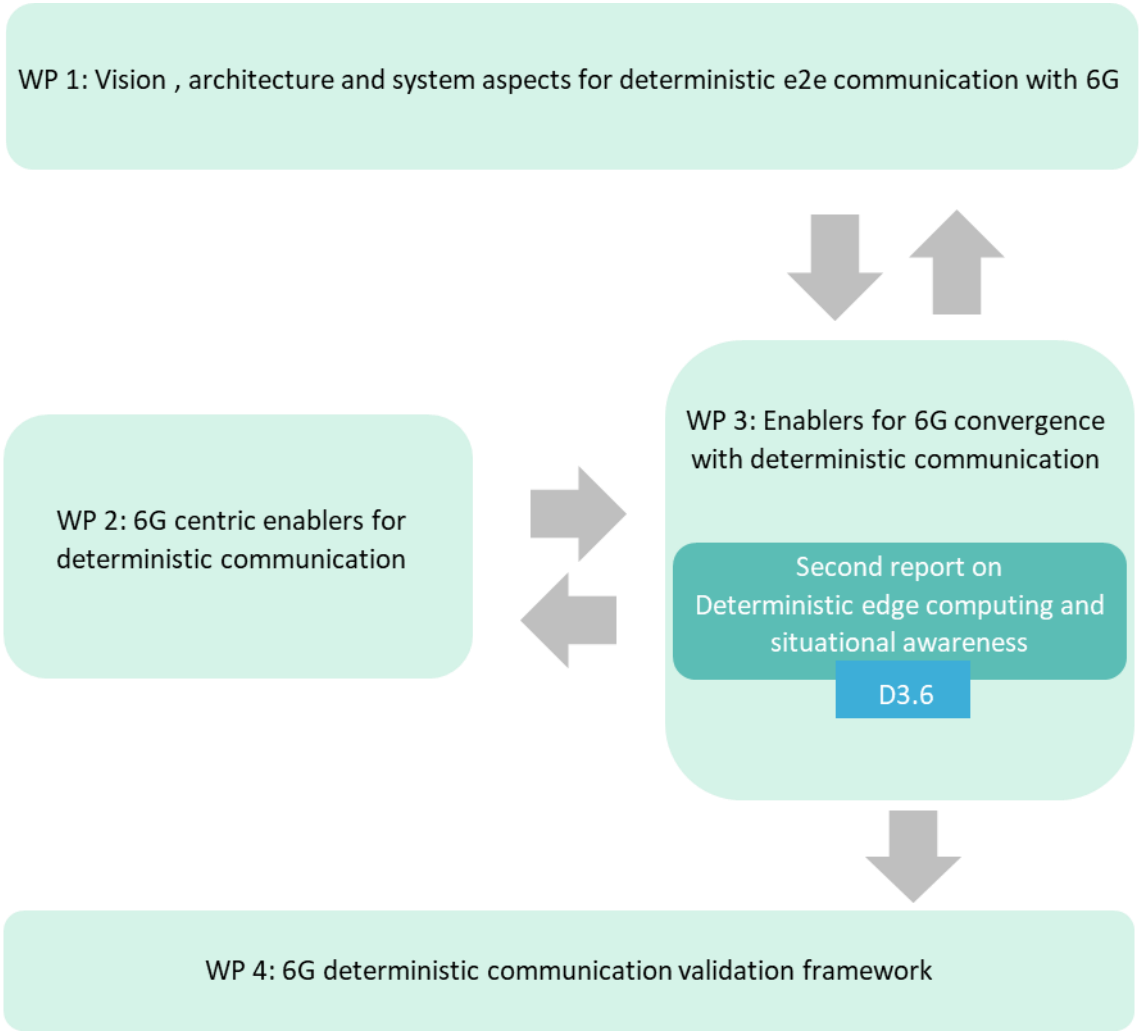


Figure 1.1: Relationship to other work packages

1.3 Objective of the document

This deliverable is the second report on Edge computing and situational awareness via digital twinning.

Work Package 3 focuses on the development of new concepts and investigates the features in the direction of 6G convergence with edge computing, deterministic communication standard and situational awareness enabled by interaction with cyber-physical digital twins. In line with the WP3 main objective, this deliverable provides details on the final edge computing and situational awareness solutions.

Edge domain data plane, control plane and controller topics discussed in the document are results from Task 3.2 (Edge cloud solutions for deterministic communication service). Main architectural options for integration of cloud and 6G deterministic communication networks, reliability and timeliness traffic handling for the cloud domain, control plane interworking is addressed.

Digital twinning topics of the deliverable are outcomes of Task 3.4 (Situational awareness via digital twinning). The aim of this task was to explore the state-of-the-art solutions and approaches for the 6G digital twin and possible interactions with other systems, via their digital twins. One of the main goals was to develop and discuss the architectural options of the 6G DT with the focus on the interaction with the physical 6G network detailing data provisioning aspects on one side, and external systems, such as OT system, on the other side. The developed architectural solutions will be presented on a use case with detailed description of end-to-end data flow from the request to DT to feedback or reaction in a real system.

1.4 Structure and scope of the document

The document is structured as follows. After the introduction in section 1, the two main parts are: Edge computing (in section 2) and Digital Twinning (in section 3).

Section 2 focuses on edge computing aspects. Section 2.1 summarizes the topics already discussed in [DET24-D33], the antecedent of this deliverable. Section 2.2 discusses data plane realization, introducing required extensions to current 3GPP MEC architecture for supporting TSN applications, and showing an Extended Berkeley Packet Filtering (eBPF)[1]-based solution for Kubernetes. Section 2.3 focuses on the control plane for deterministic edge cloud, explaining the abstraction of the Edge domain towards the TSN control plane, exposure and configuration processes for different architecture options. Section 2.4 discusses the reliable distributed control system itself, the approach of a dependable SDN controller.

Section 3 focuses on digital twinning via situational awareness in 6G, starting with the recall on the previous related deliverable D3.3 regarding the DT topic. Section 3.1 introduces the OT DT, state-of-the-art with the focus on the Asset Administrative Shell (AAS) approach as the OT DT. Section 3.2 introduces the 6G DT, state-of-the-art with the focus on the requirements of the Digital Twin Network (DTN), applicable to 6G DT. Further the architectural options of 6G DT are discussed with the focus on details of data provisioning from the physical 6G network and the interaction with external systems, such as OT domain. Moreover, a maturity gap between the OT DT and 6G DT is explained. Section 3.3 focuses on a detailed use case description with the focus on the data flow in case of interaction of OT and 6G systems via their DTs.

Finally, section 4 concludes the deliverable.

[1] Extended Berkeley Packet Filtering - https://ebpf.io/

2 Edge computing

2.1 Background

The “First report on Edge computing and situational awareness via digital twinning” [DET24-D33] examined the pressing need to establish a robust network infrastructure capable of supporting end-to-end time-critical communication services. This necessity is driven by ongoing digital transformation and the increasing demand for applications requiring high reliability and low latency. To meet these demands, [DET24-D33] advocates for a significant shift towards an edge computing paradigm, specifically through the deployment of the Multi-access Edge Computing (MEC) platform. MEC is proposed as an innovative solution to enhance low-latency applications by positioning time-sensitive tasks closer together, thereby improving data transit efficiency and overall network performance.

The [DET24-D33] deliverable outlined various deployment scenarios for MEC, focusing on its integration with Mobile Network Operator (MNO) infrastructure. It was highlighted how this integration can significantly reduce latency and delay variation compared to traditional, centralized cloud-based solutions, which is crucial for applications where even minor delays can lead to suboptimal performance or failure. Furthermore, [DET24-D33] explored integrating Time-Sensitive Networking (TSN) applications within the MEC framework, presenting initial solutions and a novel framework to support the IEEE 802.1Qbv [IEEE15-802.1Qbv] scheduled traffic standard. This framework proposed seamlessly combining application scheduling and traffic handling within the compute domain to ensure that time-critical cloudified applications receive essential support.

To substantiate these proposals, [DET24-D33] utilized the DETERMINISTIC6G simulation framework to demonstrate how two alternative realization options of the proposed framework can manage end-to-end 802.1Qbv-aware traffic effectively within the compute domain. Initial performance results underscored the advantages of MEC over cloud-based deployments, particularly in reducing delay variation, essential for maintaining the integrity and efficiency of end-to-end applications.

Extending what was described in [DET24-D33], this document provides details of the data plane and control plane integration aspects of edge domain to a TSN system.

2.2 User plane integration aspects of Edge compute domain with TSN

2.2.1 3GPP MEC architecture integration with TSN

This section describes the required extensions or additional service description required in current 3GPP MEC architecture for supporting TSN applications.

3GPP has defined a generic MEC architecture for supporting applications to benefit from low latency by allocating computing resources close to the radio access. Moreover, 3GPP includes the specifications to support TSN applications where the 5G system can be incorporated as a virtual bridge, controlled by means of a TSN Application Function (TSN AF). Therefore, the TSN AF presents the 5G network as a bridge to the TSN controller. The TSN AF acts as a Network Configuration (NETCONF) protocol with Yet ANother lanGuage (YANG) server which a Centralized Network Controller (CNC) would externally perceive as a bridge with remote management and TSN capabilities. To facilitate TSN bridge functionality, the 5G network must fulfill TSN requirements and the 3GPP MEC

architecture should support those requirements. The TSN AF would communicate device parameters to these devices based on the configuration sent by the CNC.

3GPP specifications do not define how to expose TSN capabilities through TSN-AF and how to manage the computing resources to fulfill the CNC requests. The primary traffic control mechanism in Time-Sensitive Networking (TSN) is the Time-Aware Shaper (TAS), defined in the IEEE 802.1Q-2022 standard [IEE8021-1Q, Sec. 8.6.8.4]. TAS operates based on a time-triggered schedule, enforcing deterministic transmission windows through mechanisms such as the Earliest Start Time (EST). By the standard, TAS is based on separating the packets of a port into queues with transmission of queues scheduled relative to a time reference. Queuing is performed by mapping packets to traffic classes based on their priority. Priority is assigned through the 3-bit Priority Code Point (PCP) of the IEEE 802.1Q VLAN tag in an Ethernet frame, requiring the network to be VLAN-aware. Each priority may have its own traffic class and associated queue, though a lower amount of traffic classes is also supported with a recommended priority to traffic class mapping for a varying number of traffic classes [IEE8021-1Q, Sec. 8.6.6]. This enables interoperability between hardware of varying capability. The enhancements to scheduled EST traffic define a two-state transmission gate for each queue [IEE8021-1Q, Sec. 8.6.8.4]. The open or closed state of the gate determines whether transmission is active for the associated queue. Furthermore, a Gate Control List (GCL) is defined as an ordered list of the gate configurations. Each Gate Control Entry (GCE) in the GCL can be represented as an 8-bit value with each bit representing the state of each queue alongside the time interval that state should be active for, before moving to the next entry in the list. The sum of these intervals makes up the total cycle time of the GCL, after which the list repeats.

The MEC computing platform can be deployed using queuing solutions such as the Linux kernel that handles packet scheduling in traffic control by queueing disciplines, or qdiscs. The Time-Aware Shaper (TAS) can be implemented using the Time-Aware Priority Shaper (TAPRIO) qdisc in Linux. In such deployments, the TSN Application Framework (TSN-AF) may translate centralized network configuration (CNC) commands into TAPRIO-compatible scheduling rules. User control of TAPRIO is available in the traffic control (TC) command-line utility packaged with iproute2 [IPROUTE2]. In TAPRIO, Linux packet priority from 0 to 15 is mapped to traffic classes with each traffic class assigned a contiguous range of queues. The qdisc parameters further include a set of schedule entries and a starting time for the schedule with base-time. A clock must also be specified to be used as a time reference. When using hardware offload capabilities of the qdisc on a supported Network Interface Controller (NIC), the execution of the gate control list is handled by the NIC, including use of a Precision Time Protocol (PTP) clock associated with the NIC as the time reference. In the aforementioned scenario, the packets are mapped from a Linux packet priority to traffic classes and traffic classes to hardware transmission queues. Von Arnim et al. [VGJ+22] explore deterministic use of TAPRIO. They explain that their scheduler is implemented via the netlink protocol, supporting update and replace commands to configure the qdisc. These commands correspond to identically named options in TC-TAPRIO. The authors find that the update command is orders of magnitude faster than the replace command, though only supporting changes in the schedule entries and base-time. By modifying the libnl[2] application programming interface (API) to support TAPRIO, the authors conclude it is possible to perform netlink change operations for TAPRIO in deterministic time without disrupting real-time communication. Von Arnim et al. note that the userspace TC control for TAPRIO is nondeterministic

[2] https://www.infradead.org/~tgr/libnl/

due to invoking new processes and performing string operations [VGJ+22]. However, as NETCONF is not a real-time protocol, it is reasonable to assume environments utilizing NETCONF would not substantially benefit from more advanced methods.

The 3GPP MEC architecture could expose the computing capabilities in terms of processing features provided by local TAPRIO and TSN-AF could interact with CNC to allocate the required queuing and priorities into the TAPRIO interface. The design could be as shown in Figure 2.1.

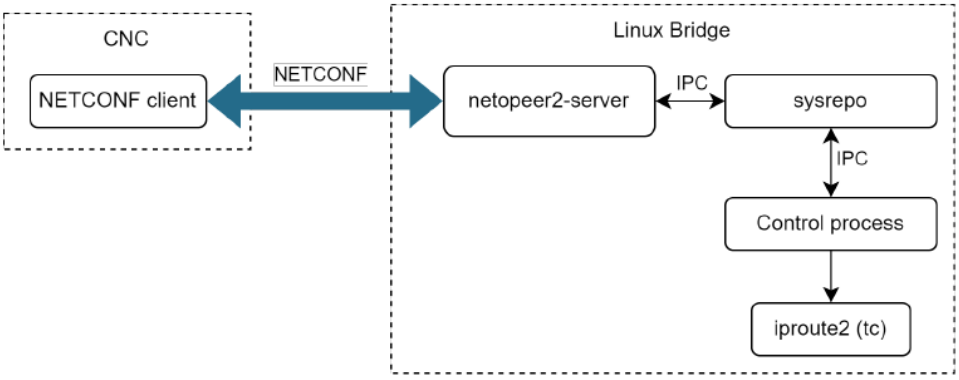


Figure 2.1: High level architecture CNC and Linux Bridge integration

The Netopeer2-suite[3] with libnetconf[4], sysrepo[5] and libyang[6] can be used to implement a NETCONF server with relevant YANG modules for a bridge with TSN features. To facilitate control of TAPRIO in iproute2, an intermediate control process is required to interface with sysrepo and invoke relevant user-space TC commands. To simplify implementation, the control process uses TC replace operations for all changes pertaining to the TAPRIO qdisc. Figure 2.1 shows a high-level view of the implementation. Netopeer2-server is used to communicate with an external NETCONF client. The server interfaces with sysrepo, for which relevant YANG modules have been installed.

Figure 2.2 represents the architecture of a virtual programmable TSN switch we are developing and experimenting during the DETERMINISTIC6G project. The switch combines the deterministic features of TSN with the programmability of P4-enabled data planes. It is built on the existing Linux taprio qdisc, which is crucial for traffic shaping in TSN environments. By integrating programmability through the P4 programming language, the switch leverages state-of-the-art technologies to enable dynamic traffic management, such as dynamic packet classification and real-time monitoring of network traffic.

[3] https://github.com/CESNET/netopeer2
[4] https://github.com/CESNET/libnetconf
[5] https://github.com/sysrepo/sysrepo
[6] https://github.com/CESNET/libyang

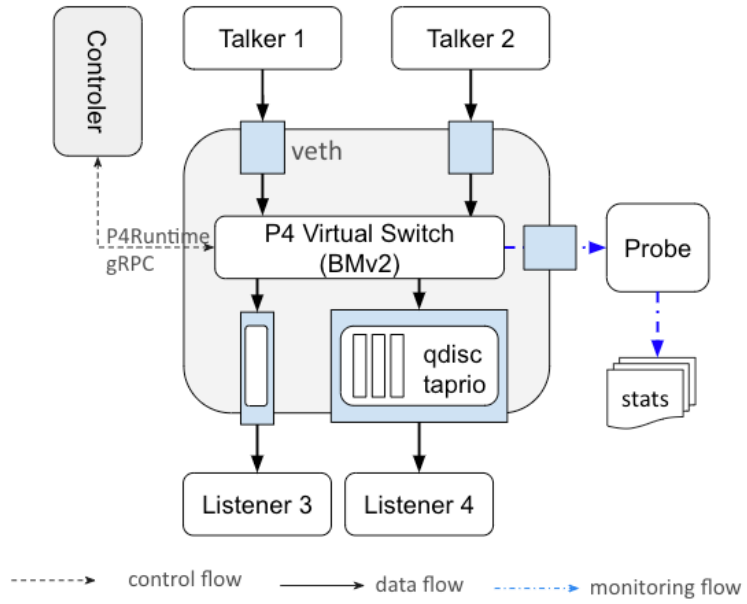


Figure 2.2: A P4-based virtual programmable TSN switch

The P4 Virtual Switch (BMv2) is the key component of the switch, serving as the data plane responsible for packet processing. By utilizing the P4 programming language, the switch allows users to define at runtime how packets are processed, classified, and forwarded to the appropriate qdisc ports. This programmability removes the dependency on manual traffic control commands, e.g., by using the TC command, enabling flexibility and runtime adjustments.

Specifically, the BMv2 virtual switch integrates seamlessly with the TAPRIO qdisc by leveraging Linux packet priority. When a packet enters the BMv2, it undergoes a parsing process defined by the P4 program. This parsing step allows users to extract relevant fields from the packet header and perform logical processing as required. For instance, operations like In-band Network Telemetry (INT) can be implemented to monitor the packet's journey through the network, collecting data on latency, jitter, or path utilization. After completing the logical processing, the PCP field in the packet header is updated to reflect its traffic class or priority. Simultaneously, BMv2 updates the `skb->priority` value of the packet in the Linux kernel. The TAPRIO qdisc uses this priority value to determine the appropriate output queue for the packet, aligning it with the preconfigured time slots defined in the time-aware scheduler. By dynamically setting priorities at runtime, this approach eliminates the need for static configurations or manual intervention, offering greater flexibility in handling diverse traffic patterns.

2.2.2 eBPF based time-aware traffic handling

Kubernetes has become the de-facto orchestration platform for deploying, operating, monitoring, and managing microservices. However, orchestration of time-sensitive application (TSA) microservices in Kubernetes is challenging for various reasons. The main difficulties can be summarized in the following points:

- lack of abstractions needed to request real-time scheduling for TSA microservices;

- cluster-wide time synchronization with PTP up to the TSA;
- centralized policing and packet scheduling of traffic classes based on their cycle times and priorities;
- unprivileged access to TSN NIC hardware capabilities from TSA microservices;
- providing network security and isolation for the TSA while passing per-packet scheduling metadata to the TSN NIC;
- relying on established send/receive primitives (e.g., Linux/BSD socket API) to avoid modification of the TSA;
- integrating various Kubernetes network implementations (CNI plugins) and container runtime.

This part of the deliverable is dedicated to the challenges described in the last four bullet points. Here, the TSA microservice is distributed as a container image and treated as a "black-box". Modification of it is not allowed and, in most cases, not even possible, since the operator who deploys it does not have the source code or packaging dependencies. As a result, simply using some internal proprietary method for TSN scheduling signaling is ruled out.

As continuation of the work described in [DET24-D33], in the following we present advances focusing on seamless TSN integration in virtualized host networks. A new time-aware packet scheduling mechanism has been developed, leveraging tight integration with cluster networking rather than bypassing it. The scheduling information (`skb->priority` and `skb->tstamp` packet metadata) is propagated with eBPF, so no direct access to the TSN NIC or application modification is required. Based on this behavior the developed solution is called as *TSN metadata proxy* or *TSN proxy* in short.

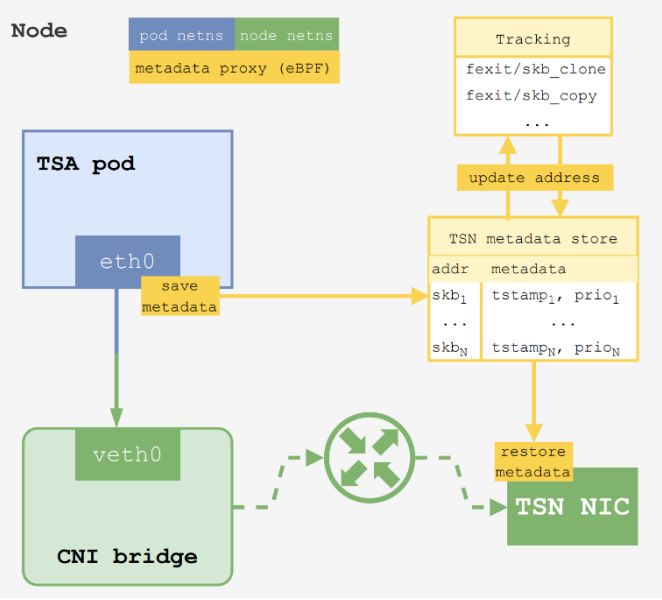


Figure 2.3: Simplified architecture of the TSN metadata proxy

The TSN proxy is implemented as a secondary Kubernetes Cluster Networking Interface (CNI) plugin to be used in conjunction with a primary CNI plugin. The primary CNI plugin can be anything that conforms to the CNI specification and is built on top of the Linux networking stack. Examples of

industry standard CNI plugins are Antrea[7], Calico[8], Cilium[9], and there are simpler ones for small clusters such as Flannel[10], Kindnet[11]. The TSN proxy can work with such plugins and extend their functionality with TSN scheduling information propagation. Advanced features such as egress and L7 policies, load balancing, and network virtualization/tunneling are all implemented by the primary plugin and require no modification or additional configuration. The simplified architecture of the TSN proxy is shown in Figure 2.3.

When a TSA microservice sends packets, `skb->priority` and/or `skb->tstamp` are set by the user space application using the socket API. Before the packet leaves the pod's network namespace (blue in the figure), the TSN proxy stores this metadata in an eBPF hashmap that acts as a key-value store (*TSN metadata store* in the figure). The key is the `skb`'s memory address and the value is the metadata itself. The `skb` representing the packet then continues its journey in the Linux network stack, and network namespace isolation takes place, deleting all the metadata. At some point, after some (virtual) bridging and routing decisions are applied, the packet reaches the TSN NIC. Before this happens, the TSN proxy restores the metadata required for TSN scheduling of the packet and removes it from the eBPF hashmap (yellow *restore metadata* box in the figure).

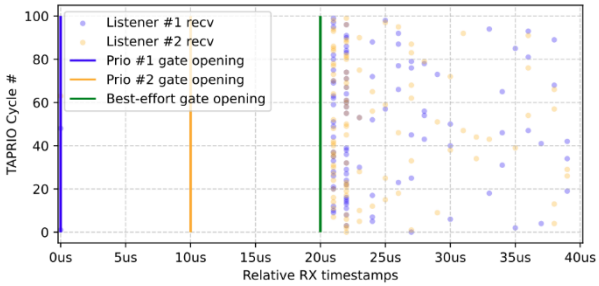
During normal operation, the packet may be dropped by the Linux network stack before it reaches the TSN NIC, or the `skb` may be reallocated with a different memory address. Dropping of packets can be caused by the lack of buffer space or an egress policy. This can happen at many places in the network stack. Reallocation of the `skb` is also common when the packet is bridged or encapsulated in a tunnel (e.g., VxLAN, IPIP). Thus, the TSN proxy needs to address the following challenges: if a packet is dropped, its entry remains in the eBPF hashmap forever, similarly if it is reallocated, the address used as the key changes, therefore finding the recoverable metadata for the packet will not be possible.

To address these challenges, the TSN proxy implements simplified packet tracking and garbage collection. The packets sent by the microservice are tracked through the Linux kernel's network stack, its routing, switching and tunneling functions. With that tracking, the TSN proxy can trace events if the packet is re-allocated because of tunneling. If that is the case, the TSN proxy can keep track of the metadata-to-packet mapping. Without this function, the TSN proxy cannot restore the scheduling metadata for the encapsulated (tunneled) packets. Packet tracking is implemented within eBPF probes that monitor the `skb_copy` and `skb_clone` functions. These functions are the Linux network APIs for `skb` reallocation and are called from many places in the kernel. When they are called, the TSN proxy checks if the source `skb` is in the eBPF hashmap, and if so, replaces its address with the newly allocated one.

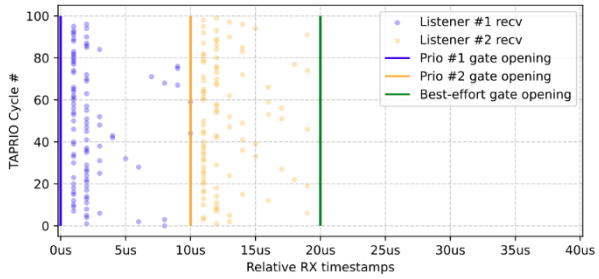
In case of packet drops, a different approach is used to keep the metadata store hashmap up-to-date. Instead of tracking the individual packet drop events per-packet basis inside the kernel, old key-value entries are periodically checked by a garbage collector. If an entry is older than a threshold (5 seconds by default), it is automatically removed. This prevents the TSN proxy from storing metadata for dropped packets forever in its hashmap.

[7] https://antrea.io/
[8] https://www.tigera.io/project-calico/
[9] https://cilium.io/
[10] https://github.com/flannel-io/flannel
[11] https://kindnet.es/

The deployment of the TSN proxy is like any other microservice. Given the manifest file and container image, a single `kubectl apply` command will install the TSN proxy's CNI plugin and do the configuration of the eBPF programs and hashmap. The name of the TSN NIC is required and can be specified in the manifest file. A limitation of the deployment is that the already running microservices need to be redeployed, as the TSN proxy is not aware of any pods started before.



(a) Without TSN metadata proxy



(b) TSN metadata proxy enabled

Figure 2.4: Packet receives timestamps mapped into 802.1Qbv cycles

The open-source implementation of the TSN proxy is available on GitHub[12]. The following simple 802.1Qbv schedule on the TSN NIC with 40 microsec cycle time can be used for evaluation:

- Priority 1: from 0 to 10 microsec
- Priority 2: from 10 to 20 microsec
- Best-effort: from 20 to 40 microsec

This cycle is repeated indefinitely, and the priorities are set by the TSAs.

Without the TSN proxy, the talker's priorities are deleted before they reach the NIC. As a result, they fall into the best-effort timeslot shown in Figure 2.4. (a). Note that packets received outside of their timeslot are transmitted immediately after their gate opens. Therefore, we have more timestamps right after the gate opens than in the rest of the slot. With TSN proxy (Figure 2.4. (b)), the priorities of the TSA packets are preserved. They are placed in their proper time slots and respect the correct gate openings. As one can see in the figure, *listener #1* receives packets in timeslot priority 1 and *listener #2* receives packets in timeslot priority 2.

[12] https://github.com/EricssonResearch/tsn-proxy

2.3 Control plane integration of Edge compute domain with TSN

As it was discussed in [DET24-D33], TSN is a key technology to ensure dependable and reliable communication solution for cloudified time-critical applications hosted, e.g., in an Edge computing environment. Section 2.4.2 of [DET24-D33] also overviews the advantages of using 802.1Qbv [IEEE15-802.1Qbv] Time-Aware Shaper (TAS) for various use cases, where the control application is deployed on the edge. The key essence of 802.1Qbv is the scheduling, which defines time-slices for forwarding frames belonging to certain Ethernet priorities across all devices in the communication network. The scheduling plan is calculated by the TSN Centralized Network Controller (CNC), based on the service requirements and the network characteristics (such as topology, TSN bridge delay, TSN end host's capability). The detailed process of configuring a new TSN stream based on a service request is showcased in Figure 2.5.

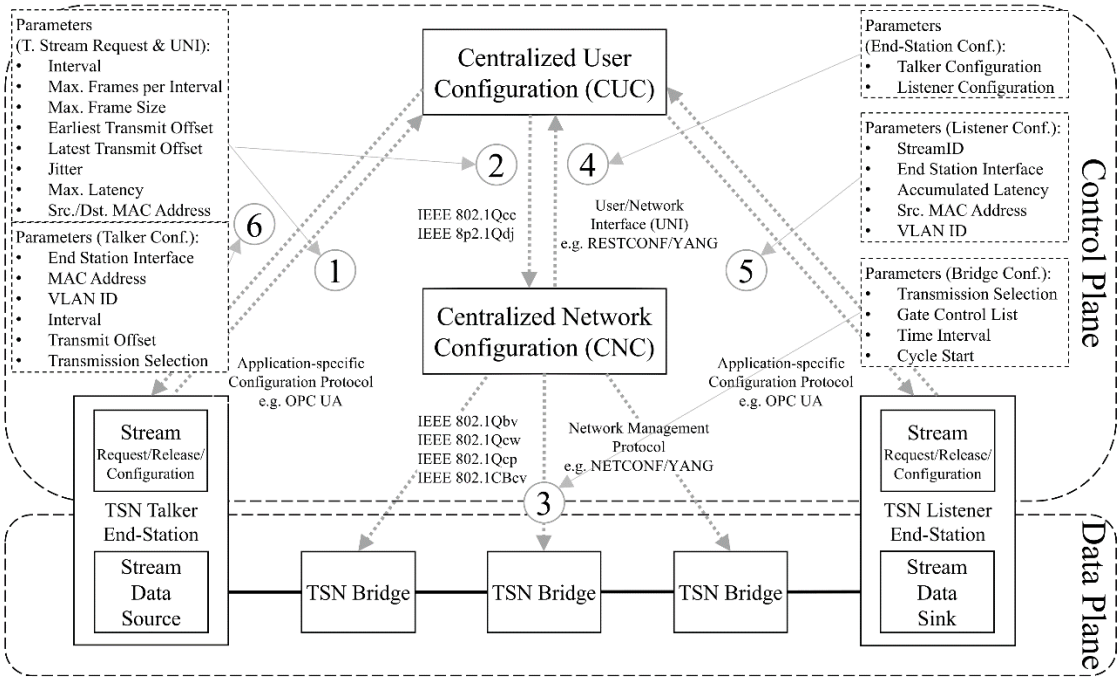


Figure 2.5: TSN stream request procedure according to 802.1Qcc

First, the TSN domain is explored by the CNC, according to IEEE 802.1Qcc [IEEE18-802.1Qcc], obtaining the per QoS class port-to-port *min* and *max bridge delay* of every TSN bridge. Additionally, the latency between each link in the network is reported. Using this information, the CNC can have a global view of the network with corresponding latency parameters. If a new service request (TSN flows) should be established, the Centralized User Controller (CUC) collects the required service parameters – *step 1* – and forwards the request description to the CNC together with the end station capabilities (e.g., Earliest/Latest Transmit Offset) – *step 2*. Then the CNC calculates the 802.1Qbv end-to-end schedule and it distributes the schedule configuration to the TSN bridges – *step 3*. The CNC also informs the CUC about the schedule – *step 4* – and the CUC configures the Talker and Listener (TSN endpoints) according to the standard IEEE 802.1Qdj [IEEE24-802.1Qdj] – *step 5*.

To support IEEE 802.1Qbv-aware traffic handling in the compute domain, in Section 2.4.3 of [DET24-D33] two traffic handling solutions in the virtualized networking of a host were proposed, which can

mimic an 802.1Qbv compatible operation. In Section 2.2.2, an eBPF-based time-aware packet scheduling enabler is showcased that integrates cluster networking, instead of bypassing it.

However, seamless user plane integration between a legacy TSN communication domain and compute domain (e.g., cloudified TSN endpoint) is not enough in itself. Control plane integration is also required. However, regarding control-plane integration two major issues arise:

1. Currently, there is no way to explore the details of a cloud ecosystem towards the TSN control plane in a standardized way. This is indispensable for constructing the proper 802.1Qbv scheduling plan by the CNC. To calculate the schedule, the following characteristics of the cloud ecosystem need to be exposed to the CNC:

    - Deployment setup: The type of virtualization has direct impact on the timing of application execution (how the application can get the required CPU resources and what is the latency to forward the packet from an application instance to the host's physical NIC).

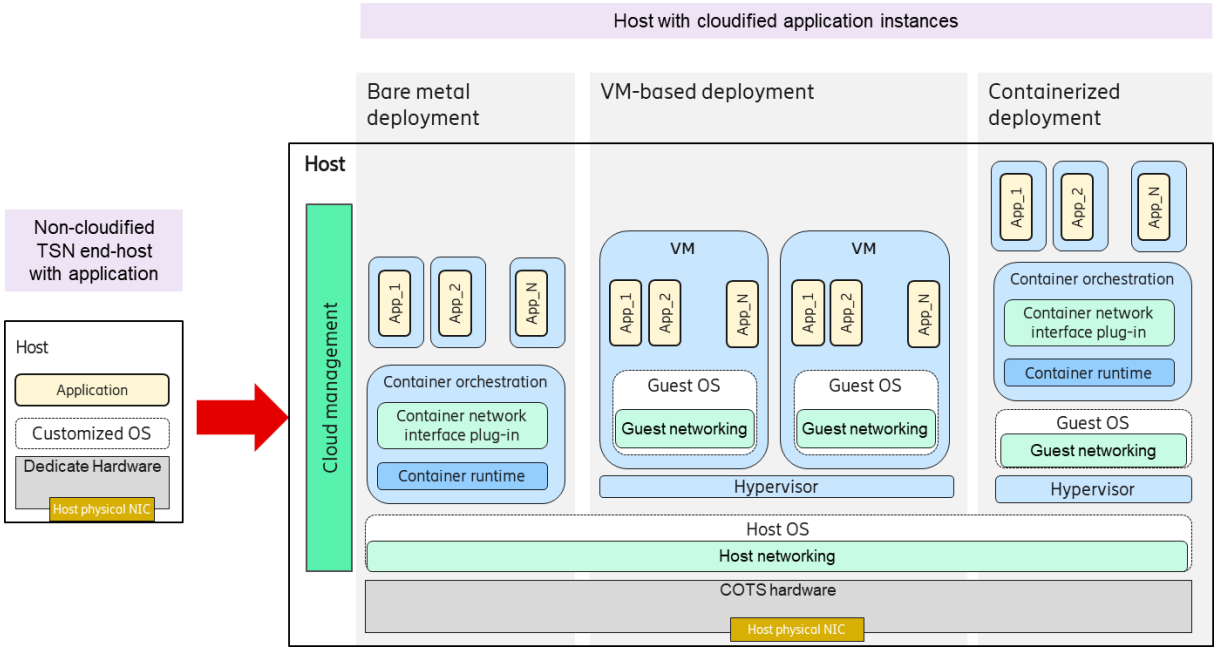


Figure 2.6: Various cloud deployment options

As shown in Figure 2.6 various deployment setups are possible, including bare metal, virtual machine (VM)-based, containerized. Even in the case of a single host, the combination of the above options is also possible.

- Furthermore, it is not enough to provide the type of deployment, but the timing characteristics of the deployment should also be known by the TSN control plane. The crucial parameters are:

    - Type of Operating System (OS), to explore if there is any real-time feature provided by the OS.

    - Type of CPU scheduling, to explore if any time guarantees can be ensured for the application execution.

- Type of networking used in the virtualized domain: one on hand, it is needed for describing the timing uncertainties of the networking. On the other hand, information about the realization options for the time-gating scheme described in Section 2.4.2 of [DET24-D33] should also be provided. This is required since, in different cloud deployments, different toolsets can be used to realize the 802.1Qbv-aware traffic handling. For example, the TAPRIO or ETF[13] qdisc could be possible options, but other traffic manipulation schemes can also be applied, leveraging the eBPF toolset.

As shown above, there are numerous parameters and characteristics of the cloud deployment, which need to be exposed to the TSN control plane in order to prepare the proper 802.1Qbv scheduling, but currently there is no standardized solution for this. However, without this information, 802.1Qbv cannot be used properly for cloudified applications.

2. The second problem arises when the TSN control plane needs to configure the setting required for the 802.1Qbv support in the cloud. The critical problem here is that the deployment of specific application scheduling and traffic handling schemes (that need TAPRIO, ETF or eBPF, as explained before) should be configured based on the scheduling plan provided by the CNC. However, since the 802.1Qbv scheduling is TSN specific, it does not contain any cloud specific details. Hence, it cannot be directly used for configuring the cloud specific traffic handling features.

To sum up, the major problem is twofold:

- There is no standardized way to provide information to the TSN control plane (CUC, CNC) about the capabilities and characteristics of a cloud host. This means that the cloud deployment specific details cannot be exposed directly to the TSN control plane and remain hidden from the TSN control plane. Without this information, the CNC cannot prepare the 802.1Qbv traffic scheduling plan.
- There is no way to properly configure the deployment specific traffic handling schemes in the cloud deployment based on the 802.1Qbv traffic scheduling plan provided by the CNC. There is no management/control plane entity in the cloud which can perform the cloud-specific configuration based on the information provided by the TSN control plane.

2.3.1 Abstraction of a cloud host towards the TSN control plane

To resolve the above-mentioned issues, an abstraction of the cloud ecosystem for the TSN control plane is proposed, which can model and describe any kind of deployment of a cloud host according to the 802.1Qcc and 802.1Qdj standards. Figure 2.7 shows the illustration of the abstraction in the case of various virtualization deployment options of a cloud host.

[13] ETF - Earliest TxTime First (ETF) Qdisc - https://man7.org/linux/man-pages/man8/tc-etf.8.html

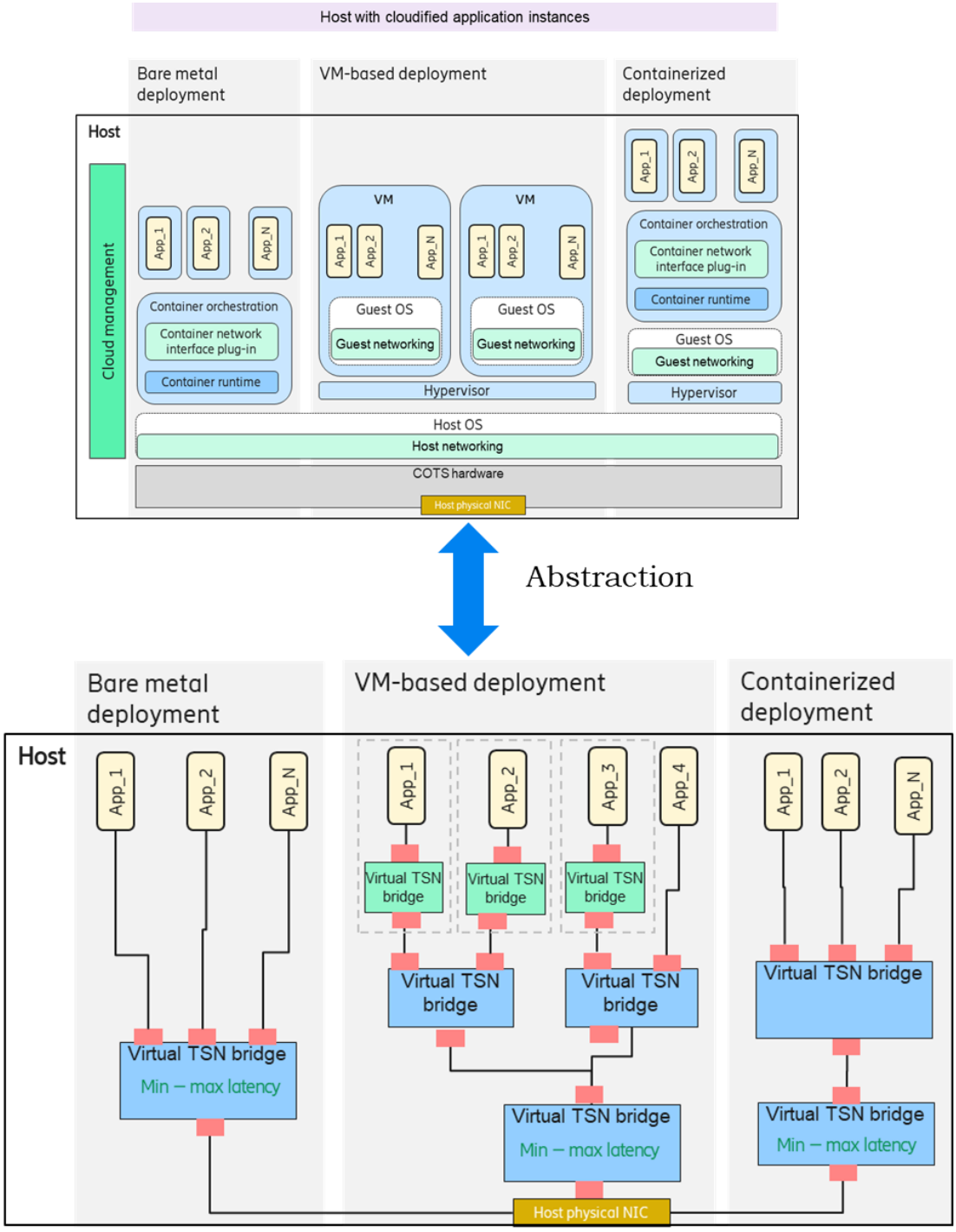


Figure 2.7: Abstraction modeling of a cloud host virtualization deployment

The upper side of the figure shows a cloud host with several types of deployments - left to right: (1) bare metal, (2) VM deployment, (3) Cloud-native deployment, when a container execution environment is deployed in a VM. The lower part of the figure shows the abstracted view of the host.

The most important components of the abstraction model are:

1. **Virtual TSN endpoints** (App in the figure)**:** these elements represent the cloudified application instances in a one-to-one mapping manner. The main intention here is to provide application timing/scheduling information[14], which represents the application scheduling capabilities in the cloud deployment. If the precise timing of the scheduling of an application instance can be guaranteed by the cloud manager and the underlying CPU scheduling, then the latency on the link between the TSN endpoint and the next virtual TSN bridge can be considered as *Transmit offset* for a certain application instance. If there is some uncertainty or variance in the scheduling of an application, then the *Earliest Transmit Offset* and the *Latest Transmit Offset* (see Figure 2.5) can be reported towards the CUC. In some cases, a more detailed description of application scheduling would be needed. In this case an alternative application modeling can also be applied, where an application is represented by a TSN endpoint and a virtual bridge (green virtual bridges in Figure 2.7). In this case, the reported *min* and *max bridge delay* can represent further characteristics/uncertainties of the application scheduling.

2. **Virtual TSN bridges**: These elements represent the networking characteristics, depending on the cloud deployment option. The default way is to represent the networking in each domain of the host (e.g., host networking, networking in the guest OS, container network interface (CNI)). The virtual bridge, which represents the networking in the host OS, is directly connected to the host's NIC. If there are multiple NICs then two alternatives are possible:

    1. Only a single virtual bridge is used with multiple northbound interface ports, which are connected to the NICs, respectively.
    2. Multiple virtual bridges are used on a per NIC basis.

    The networking on the different levels of the virtualization is also represented by separate virtual bridges as it can be seen in the VM-based deployment part of the figure (separate bridges are used for the different VMs due to the isolation between VMs). According to the levels of virtualization the virtual bridges form a tree structure, where the root is the virtual bridge that represents the host networking and is connected directly to the host's physical NIC.

2.3.2 Detailed deployment exposure and configuration process based on the abstracted model

[14] In this context, the application scheduling means how the CPU resources are reserved for an application instance for task execution. By other words, it means the timing, when a certain application instance can start to forward a packet toward the NIC.

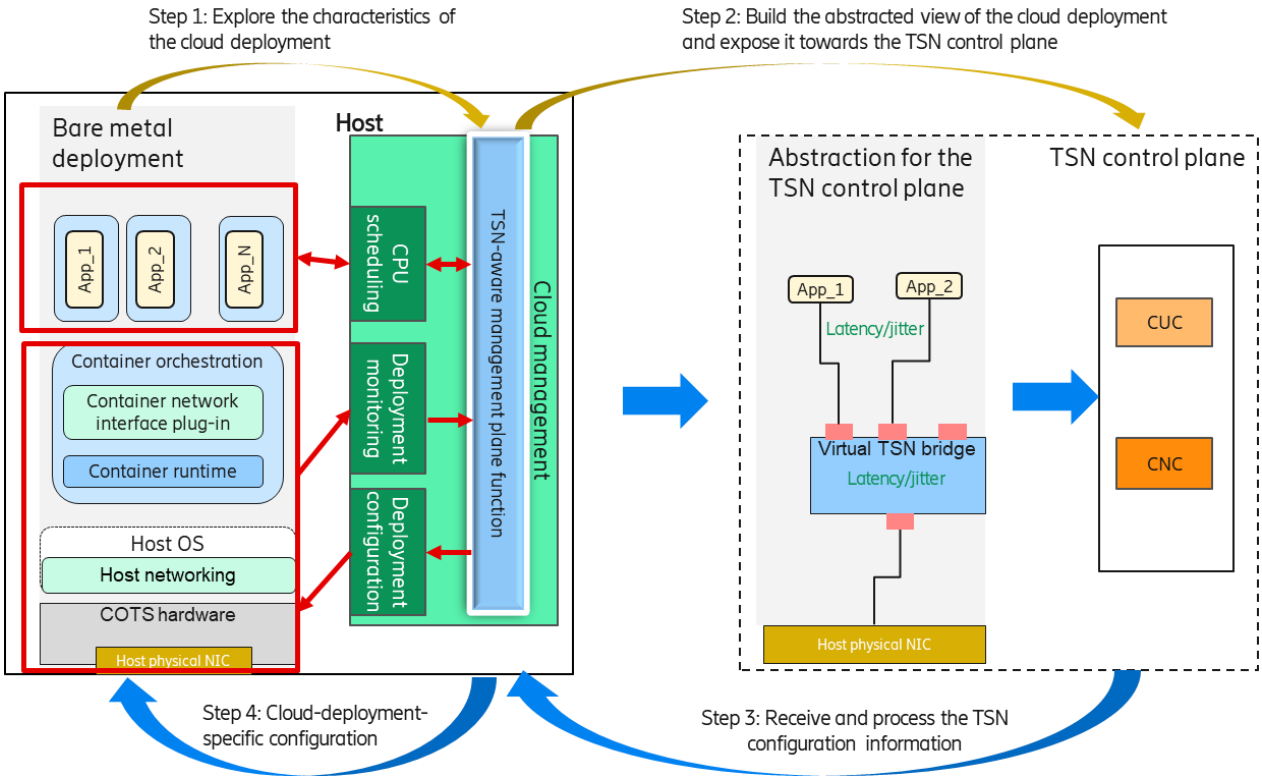


Figure 2.8: Cloud deployment abstraction, exposure and configuration process

As illustrated in Figure 2.8, a *TSN-aware cloud management entity* (blue entity in the cloud management box in the figure) is also proposed with the following functionalities:

- It is fully aware of cloud deployment and able to explore and collect all the details of the deployment and its characteristics, like application scheduling and networking. To this end, the *TSN-aware cloud management entity* has interfaces towards the CPU scheduling entity of cloud management. Furthermore, it can obtain information about the time characteristics/uncertainties of the networking on various virtualization levels.
- Based on this information, the *TSN-aware cloud management entity* can construct the abstracted view of the cloud deployment including the TSN endpoints and virtual bridges and can parameterize the abstracted view according to the above description of the components (virtual TSN endpoint and bridges) of the abstraction model.
- Towards the TSN control plane (CUC/CNC), the *TSN-aware cloud management entity* acts as a proxy with the capability to expose the abstract view according to the IEEE 802.1Qcc and IEEE802.1Qdj.
- The *TSN-aware cloud management entity* is also responsible for receiving the 802.1Qbv scheduled traffic configuration and application configuration provided by the CNC and CUC.
- Additionally, leveraging its cloud-awareness, the *TSN-aware cloud management entity* is able to translate the 802.1Qbv scheduling plan to cloud deployment specific configuration and it is able to enforce the required configurations in the (1) application scheduling and (2) traffic handling in the different levels of virtualized networking, to ensure the 802.1Qbv-aware traffic handling in the cloud deployment.

Step 1: Explore the cloud deployment by the TSN-aware cloud management function

Initially, the TSN-aware cloud management function starts to explore the characteristics of the cloud deployment, including:

- Exploring the type of virtualization, which is used to determine the structure (topology) of the virtual TSN bridges in the abstraction model;
- Exploring the networking configuration capabilities in the virtualization domain of the host. This information is required for the traffic handling configuration step, based on 802.1Qbv schedule provided by the CNC (details will be provided in Step 4). This exploration is necessary, since different (cloud-specific) configuration options are possible in the case of container- and VM-based deployment, Furthermore, in some container network interface options, hardware offloading of traffic handling is possible, so such related information should also be collected;
- Using a monitoring solution, the TSN-aware cloud management function will measure the network capabilities, like latency and jitter in the virtualized host networking. This information will be used to calculate the *min* and *max bridge delay* parameters for every virtual TSN bridge that represents the networking in the virtualized domains of the host. The monitoring solutions must be deployed by the owner of the cloud deployment, but there are no further requirements on the monitoring. Any kind of monitoring option that can determine the latency and jitter components in a virtualized environment is suitable for this purpose. Due to the stochastic characteristics of the compute components, in some cases, the *min* and *max bridge delay* are not enough to precisely describe the system capabilities, which may cause undesirable inaccuracies in the calculation of the TSN schedule. One possible approach to address this issue is to apply the approach proposed by DETERMINISTIC6G for the detailed latency description of wireless systems, which is documented in Deliverable D3.4 [DET24-D34]. In a similar way, instead of the static *min* and *max* values, the delay of the virtual bridge components of the compute system can be extended to a histogram. This enables a fine-grained description of the system capabilities in the demanded granularity, ensuring the accurate calculation of the TSN schedule.
- To provide the *Earliest Transmit Offset* and the *Latest Transmit Offset* parameters for each application instance for the CUC, the TSN-aware cloud management function obtains the details about the application scheduling plan and possible uncertainties. First, the used scheduling (e.g., FIFO, Earliest Deadline First (EDF)) and the scheduler configuration are obtained and then the possible application scheduling timing is determined, considering the application execution time. Based on this information the *Earliest Transmit Offset* and the *Latest Transmit Offset* parameters are calculated in the abstraction model. If the combination of a TSN endpoint and a virtual TSN bridge is used for representing an application, the *min* and *max bridge delays* are also determined in this step, based on the scheduling capabilities/limitations.

Step 2: Build the abstracted view of the cloud deployment and expose it towards the TSN control plane

Using the collected information, the abstracted view of the cloud host deployment can be constructed and parameterized as a set of virtual TSN endpoints and bridges. Then, the TSN-aware cloud management function exposes the details of the abstracted view (topology and characteristics of the virtual TSN entities) to the CNC and the CUC using the standard NETCONF protocol. Based on this information, the CNC can calculate the 802.1Qbv schedule.

Step 3: Receive and process the configuration information from TSN control plane (CNC and CUC)

When the CNC determines the 802.1Qbv scheduling plan, it initiates the configuration of the bridges in the network. In an end-to-end deployment, it includes the configuration of physical bridges, and in addition, includes the configuration of the virtual bridges of the abstracted view of cloud deployment. This means that, for the egress port of each virtual TSN bridges (that represents the networking in the virtualization domain), the corresponding 802.1Qbv scheduling configuration is provided by the CNC.

It should be noted that, albeit the current description focuses on the description of the 802.1Qbv scheduled traffic, the generic manner of the abstraction model enables the usage of other traffic handling solutions of the TSN. For example, the IEEE 802.1Qci - Per-Stream Filtering and Policing [IEEE17-802.1Qci] provides mechanisms for filtering and policing individual data streams to ensure robust performance by preventing congestion and ensuring compliance with predefined traffic policies. In the case of 802.1Qci, per-stream filtering is configured to the ingress port of a TSN bridge. Using the abstraction model, the 802.1Qci configuration can also be determined for the ingress ports of the virtual bridges.

Since the CUC – based on the received information from the CNC – is responsible for configuring the TSN endpoint, the CUC sends this configuration information to the TSN-aware cloud management entity and contains the *Interval* and *Transmit offset* parameters.

Step 4: Cloud-deployment aware configuration

The final step is to perform the TSN specific 802.1Qbv configuration information to cloud-deployment specific configuration:

- The *Interval* and *Transmit offset* parameters provided by the CUC are translated to application scheduling configuration. In the case of Earliest Deadline First (EDF) scheduling, considered as an illustrative example, the *Interval* is used to specify the `period` parameter of EDF, while the *Transmit offset* is used to calculate the `deadline` parameter.
- The per (virtual TSN bridge) port 802.1Qbv configuration must be translated to the cloud-specific time-aware traffic handling solutions. For example, if a Linux TAPRIO-based per-container gating scheme is applied (see Section 2.4.3 of [DET24-D33]) then the opening times for each per-container gates are configured in a coordinated manner, based on the 802.1Qbv scheduling information and the application scheduling. If a centralized 802.Qbv-aware traffic scheme is used (as also described in Section 2.4.3 of [DET24-D33] for an OVS-based cloud management deployment), the required traffic shaping realized by the usage of Linux qdisc discipline is also configured based on the 802.1Qbv scheduling information.

The TSN-aware cloud management entity is responsible for calculating and configuring the parameters of the applied cloud deployment specific traffic handling solution.

2.3.3 Standalone Edge scenario

In a standalone edge deployment, a standalone datacenter infrastructure is deployed to host the cloudified applications (also referred as TSN Talkers/Listeners). This type of edge deployment scenario typically supports local deployment, and enables significant flexibility, such as distributed edge

infrastructure for ensuring high reliability, customized traffic handling, and networking solutions in the virtualized domain tailored for tight interworking with the TSN domain.

From a deployment perspective, two major options can be described. In the first option – which is depicted in Figure 2.9 – all the devices in the edge deployment infrastructure (including the border gateway switch(es), cloud transport switches and host NICs) are TSN capable devices.

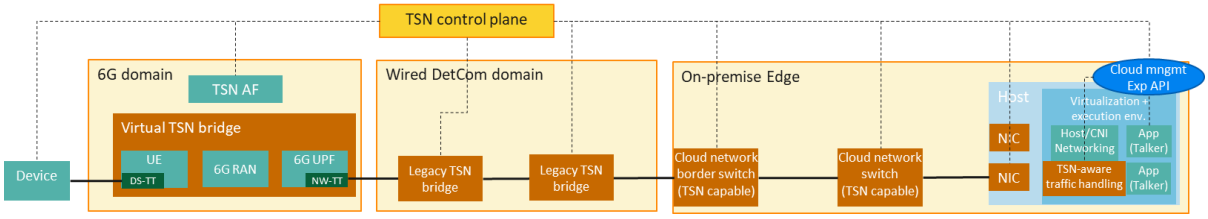


Figure 2.9: End-to-end TSN deployment

The TSN control plane can obtain all the required information on the topology and the capabilities of the switches, and using the solution described in Section 2.3.2, the host capabilities can also be obtained. Based on these capabilities, the traffic schedule parameters can be configured in all devices, in an end-to-end manner. However, this option requires some investments to upgrade all the cloud infrastructure devices and host NICs. The obvious advantage of this option is that all the TSN features (e.g., TSN 802.1Qbv – scheduled traffic, or TSN 802.1CB – Frame Replication and Elimination for Reliability) can be fully utilized in the cloud domain.

In the second option, which is depicted in Figure 2.10, only the host NIC and the border gateway switch(es) of the cloud infrastructure domain are TSN capable, while the cloud transport switches are not TSN-capable.

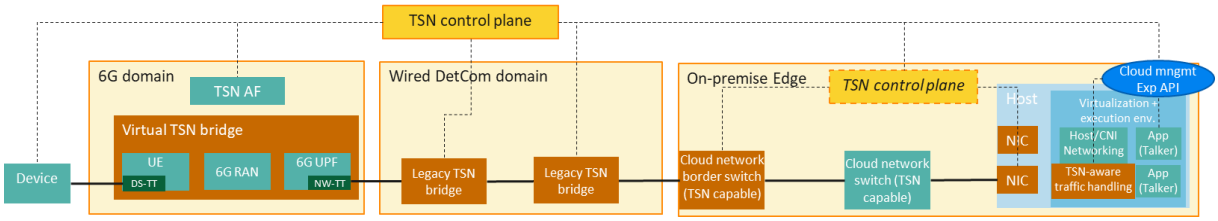


Figure 2.10: TSN capabilities only in cloud end-host and border gateway switches

This means that this option avoids upgrading the (legacy) cloud infrastructure devices and requires the installation of new border gateways and host NICs. The result is a more cost-efficient solution than the previous option. On the other hand, due to the presence of non-TSN devices in the cloud transport network, the TSN streams remain transparent to the cloud transport network. Consequently, detailed capability exposure of the cloud transport network is not possible; only an aggregated view can be exposed to the TSN control plane outside the cloud domain. One way to achieve this abstraction is to use a local CNC (*TSN control plane* in italic, in Figure 2.10), which can obtain all the necessary information about the cloud infrastructure domain, aggregate this information, and forward it towards the higher-level TSN control entities.

2.3.4 Operator enabled scenario

In this setup as illustrated in Figure 2.11, the 6G network operator owns or operates the edge cloud domain as well, enabling a tighter integration of this domain. As outlined in Section 2.3 of [DET24-D33], the components of the 6G network can be deployed within the edge cloud domain. However, this report specifically concentrates on facilitating the deployment of TSN applications, such as TSN talker/listener, at the edge.

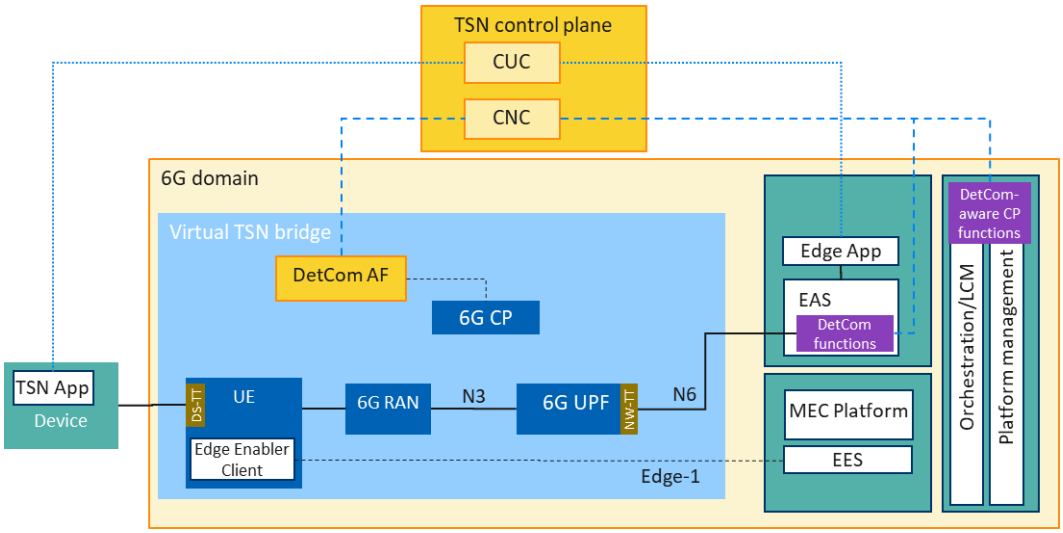


Figure 2.11: Operator enabled Edge

For a TSN App on the device to utilize functionality provided by another TSN App deployed in the cloud, the application deployment, service initiation and service configuration steps are needed.

2.3.4.1 Application deployment

Application deployment is the step where a new edge application instance gets deployed, see Figure 2.12. This edge application is then connected to the device TSN applications.

The participating entities in this process are the Application Provider (AP), which supplies the application to be deployed; the Application Owner (AO), like a factory owner, which wants to deploy the application; the Mobile Operator (MO); and the Edge Provider (EP), which in this case is the same as the MO.

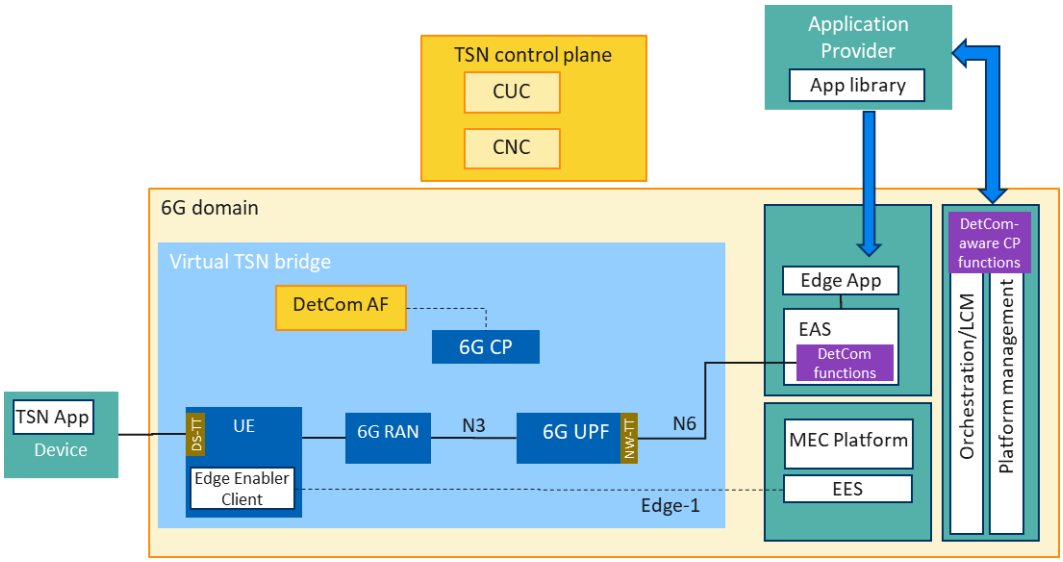


Figure 2.12: Application deployment

The process of application deployment is:

1. AP/AO intends to deploy an application (e.g., exoskeleton control app) on one/multiple Edge premises.
2. The placement of the application is an orchestration task: finding a suitable edge host for the app (placement), with TSN 'supported' user plane NW-TT connection. The Host selection is based on the capabilities of the host to support deterministic/real-time computing. The details of these capabilities are described in Section 2.4.1 of [DET24-D33]. These include Hardware and Software (including Operating System) capabilities like RTOS, isolation, CPU pinning, etc. It is worth noting that EAS (Edge Application Server) service KPIs defined in Table 8.2.5-1 of [3GPP23-23558] could also be considered, but for the proper placement of apps with real-time requirements, additional parameters should be considered.
3. The host is configured to support the real-time application (via MEC Platform Manager [ETSI19-MEC003], including TSN-aware CP functionalities).
4. Optionally, the Control Plane (CP) of the application deployment is exposed towards the TSN control plane. This may include the establishment of CP connection between the host cloud management and the TSN control plane (CUC/CNC). Note: alternatively, this CP level connection could be established when the TSN communication services is initiated, see the next step.

2.3.4.2 Service initiation

Service initiation (or application discovery) is the step where the deployed edge application is made available and discoverable, see Figure 2.13.

The entities in this process are the Application Owner, like a factory owner, that wants to initiate the application; the Mobile Operator; and the Edge Provider, which in this case is the same as the MO.

As a result, the AO wants to establish an edge-enabled TSN service.

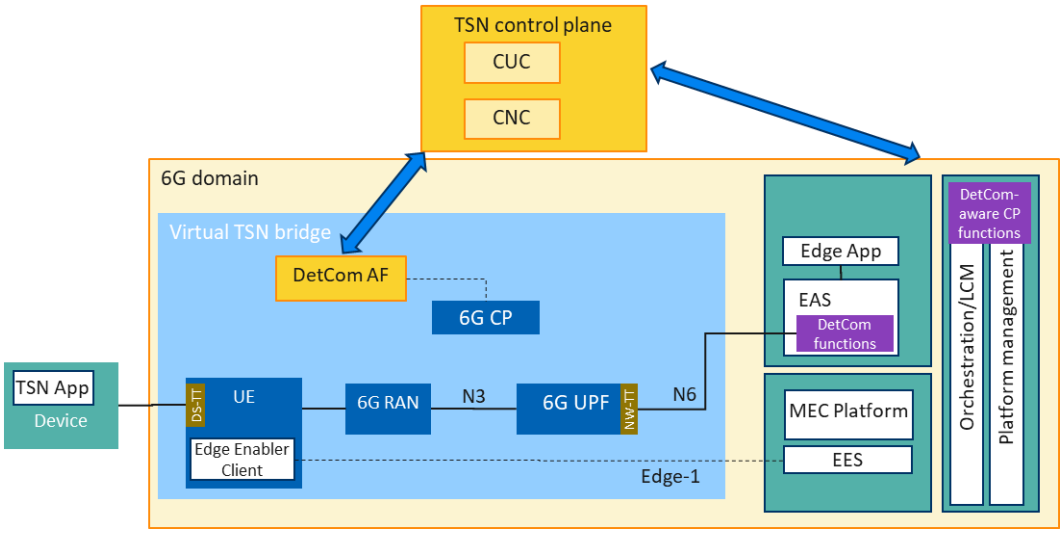


Figure 2.13: Service initiation

The process of service initiation is:

1. EAS discovery initiated by the EEC (Edge Enabler Client) (see clause 8.5 of [3GPP23-23558]. This is based on the EAS profile and AC (Application Client) profile match. In some cases (depending on the service), an extension of the parameters would be needed, such as 'TSN/real-time support', as hinted in [DET24-D33]. Pre-defined EASIDs (Edge Application Server Identifiers) could also be applied.
2. The ECC can directly talk to the selected EAS where the Application instance is deployed.

2.3.4.3 Edge Capability exposure

Capability exposure is the step where the Mobile domain and the Edge domain report their capabilities and configuration to the TSN control plane (see Figure 2.14).

The entities in this process are the Application Owner (control plane owner), Mobile Operator and the Edge Provider.

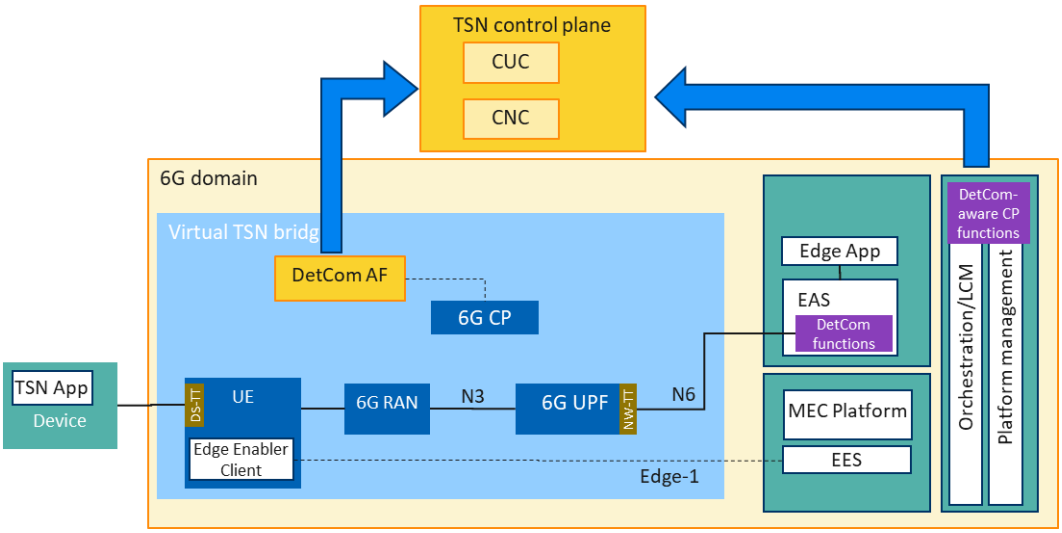


Figure 2.14: Capability exposure

The steps for this process can be summarized as:

1. The DetCom-aware CP function exposes the Edge host/deployment details and capabilities to the CUC/CNC; this includes the virtual end points and the virtual bridges model of the host deployment. Host specific information is translated to 801.Qcc description by the DetCom-aware CP function.
2. Edge deployment exposure, according to the 3GPP-supported exposure options (described in the next sub-sections).
3. The exposed topology (between the end device and edge application/deployment) is then taken into consideration by the CNC. (Here, 6G virtual bridges need to be considered.)

The way the information is sent to the CNC/CUC depends on how the edge control plane is connected to the TSN control plane. We highlight two options in the next subsections.

Option 1: Direct interface between the Edge platform and the CUC/CNC

In this setup the Edge domain is connected to the TSN control plane like in the case of the standalone Edge (described in Sec 2.3 of [DET24-D33]), see Figure 2.15.

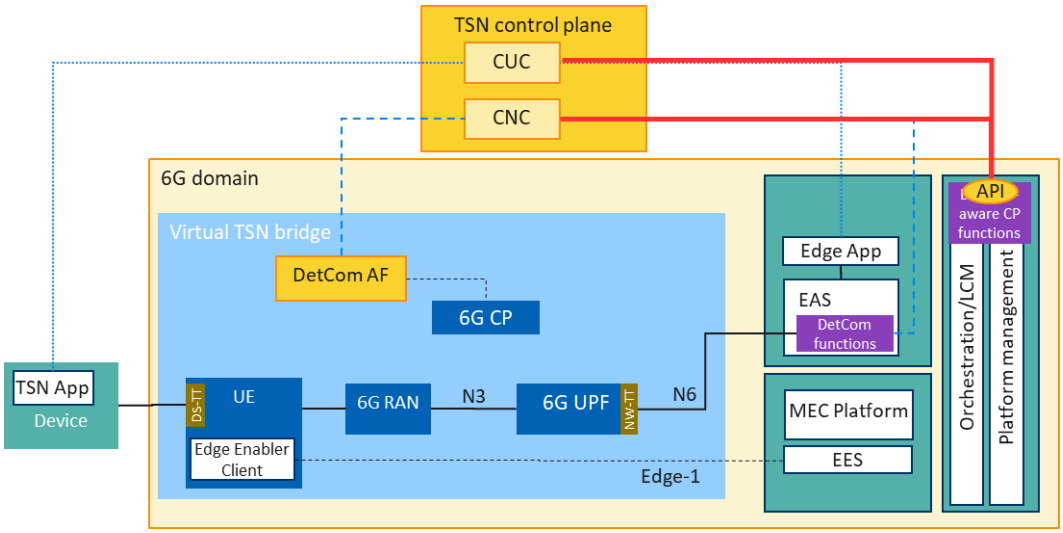


Figure 2.15: Direct Edge CP interface

The interface between the TSN control plane and the Edge domain is similar to the interface between the TSN control plane and the 6G domain. The domain is represented as a single or as a set of TSN Bridges and endpoints. Furthermore, the interface could be extended with (proprietary) components to better expose Edge capabilities.

The main advantage of this option is that no 3GPP standard is required, as the CNC/CUC communicates directly with the Edge domain without involving Mobile domain control plane entities. The advantage of having the same operator for the Mobile and the Edge domain comes from easier configuration between the domains and shared cloud deployment.

For the edge capability exposure, an MNO (Mobile Network Operator) specific/proprietary solution located in the TSN aware CP functions block will obtain the edge capabilities and translate them to TSN conform parameters, then sending them to the CNC/CUC.

Option 2: Using SEAL architecture

In the Service Enabler Architecture Layer (SEAL) [3GPP18-23434] based architecture the Edge domain control plane is connected to a SEAL Network Resource Manager (NRM) component. This component also acts as the TSN-AF for the 6G domain, see Figure 2.16.

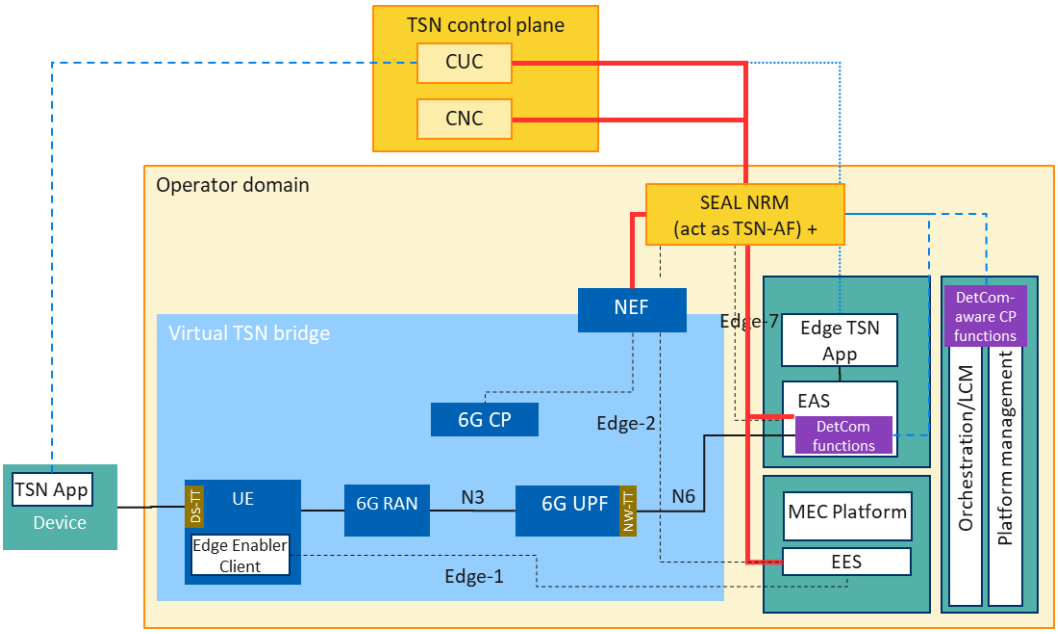


Figure 2.16: Seal-based Edge CP interface

On one hand, the SEAL NRM is the TSN-AF, representing the 6G domain to the CNC/CUC, like in a non-Edge enabled scenario described in [DET24-D12]. In this case, the SEAL NRM configures the 6G domain via the N5 (to the PCF - Policy Control Function) or N33 (to the NEF - Network Exposure Function) interfaces as described in Section 14.2.2.2 of [3GPP18-23434].

On the other hand, the SEAL NRM is responsible for the Edge capability exposure and configuration, interacting with the TSN CNC/CUC also on behalf of the Edge domain. To support this operation, the Edge-2 and Edge-7 interfaces could be used to exchange information between Edge deployment and the SEAL NRM. According to [3GPP23-23558], currently these interfaces are used for 'for retrieval of network capability information'. Optionally, the Edge-2/Edge-7 interfaces could be extended to provide information about Edge capabilities towards the SEAL NRM, or a proprietary interface could be used between the SEAL NRM and the Edge domain.

The main advantage of this solution is that both the 6G and the Edge domain are represented by a single entity towards the TSN control plane. This enables various abstraction and optimization options. While the SEAL NRM increases in complexity, it does not affect the existing TSN-AF functionality.

For the edge capability exposure, the SEAL NRM will obtain the edge capabilities and translate them to TSN conform parameters and then send them to the CNC/CUC. Similarly to the previous architecture option, standard information could be extended with Edge specific parameters.

Option 3: Extension of TSN-AF

To ensure thoroughness, we include this option. The concept involves extending the TSN-AF component in the 6G domain to communicate with the Edge domain and represent both domains to the CNC/CUC.

However, in addition to the functionality required in the previous option, this option has major impact on the TSN-AF, with significant standardization efforts needed and requiring the extension of Edge-2/Edge-7 interfaces. There is no intention within 3GPP to pursue this direction, and we do not advocate for this approach.

2.3.4.4 Service configuration

Service configuration is the step after the capability exposure. TSN specific configuration of both the 6G and the Edge domain is performed here, see Figure 2.17.

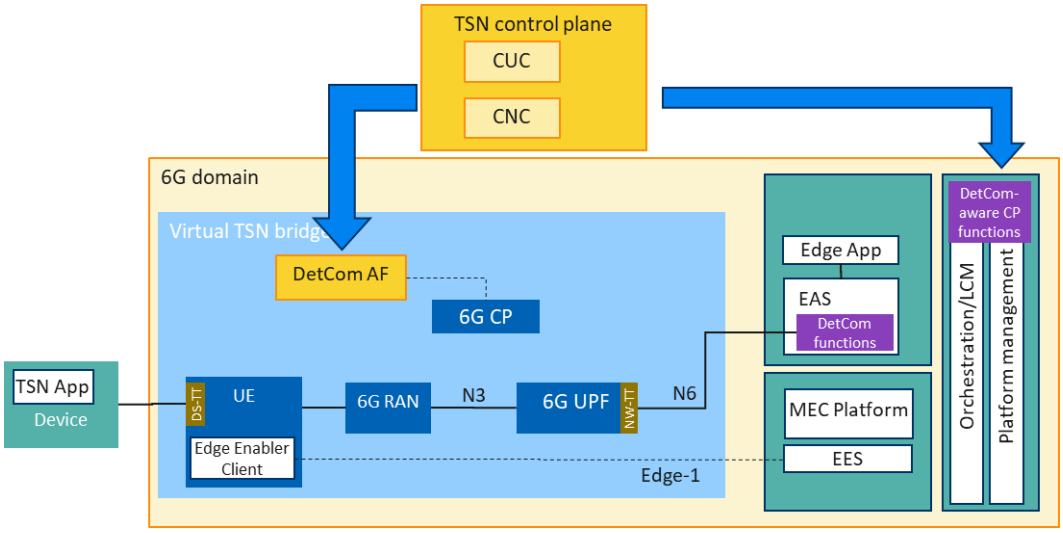


Figure 2.17: Service configuration

The entities in this process are the Application owner (the factory owner that wants to deploy an application), Mobile Operator as Edge Provider (also known as Compute as a service) and the Edge provider (edge service offered via the mobile operator).

The steps for this process can be summarized as:

1. The CNC configures the 3GPP system according to the current 3GPP standards. No new functionality is needed here.
2. The Edge cloud domain is configured, according to the architecture option (described in the previous section), either via direct interface to the Edge (see Section. 2.3.4.3 Option 1), or via the SEAL NRM (see Section 2.3.4.3 Option 2).

The configuration of the data plane in the Edge domain is done according to the chosen implementation (see Section 2.2).

2.4 SDN controller

In the deliverable [DET24-D33] related to edge computing, a high-level introduction to the view of the DETERMINISTIC6G consortium about a 6G network controller as an edge computing service is given, which can be realized as a software-defined network (SDN) service running on edge computing resources. The network controller could be connected to the TSN control plane via the 6G domain's TSN-AF. Then the network controller is intended to control the elements of the packet forwarding network (or the data plane) of a 6G network under dependable, time-critical communications constraints. Since the controller is intended to be run on edge computing systems, it realizes the deployment of a load (in this case, a 6G network controller) to the edge computing domain. The focus of this section is on the dependability of this 6G network control plane itself. More specifically, the focus is on the study of the reliability of 6G network controllers provided as edge computing services. Then the possible extension of the principles of such a reliable SDN control network to include time-critical constraints is studied, and (more generally) the possibility to include other dependability constraints is studied. Therefore, a basic principle of a framework for the design of this kind of reliable control network as an example of dependable network is provided, and then the possibilities of extending the properties and capabilities of this example dependable control network design to cover time-critical constraints and other dependability constraints are studied.

2.4.1 Dependable deterministic SDN controller design

2.4.1.1 Main objective and control principles

One of the basic principles of network architectures, a principle that we consider still valid for 6G networks (at-least the transport network part), is the interconnection of network elements (NE)s into a packet forwarding network (forming the data plane, also referred to as "user plane" in 3GPP standards) which is controlled by a separate control network (forming the control plane). The data plane transports user and application data end-to-end, while the control plane controls the behavior of the data plane. In terms of this, the control plane can be used, for example, to manage elements of the data plane and provide network management applications.

The control plane is the focus of section 2.4. The recent evolutions of control network design principles to include more complex software techniques and the use of general-purpose processors (GPP)s at scale, allow to think of a control plane as consisting of an external control system that is combined with (and complements) internal control systems that are integrated into network elements.

Figure 2.18 shows an example of a control network system consisting of an external control system, which is connected to (and exchanges control information with, via a southbound interface) a set of NE(s), each NE consisting of a forwarding element and a control element. In each NE, the forwarding element implements a forwarding logic, and the control element implements a control logic. The control logic and the forwarding logic within each NE can communicate typically with each other in a way chosen by a given implementation, and in some cases with a proprietary bus inside the NE. The forwarding elements of the NE(s) are interconnected to form the packet forwarding network, while the control elements are interconnected to form the internal control system. The internal control system is complemented by an external control system, which implements an external control logic, and constitutes an extension of the internal control system to provide more computing resources.

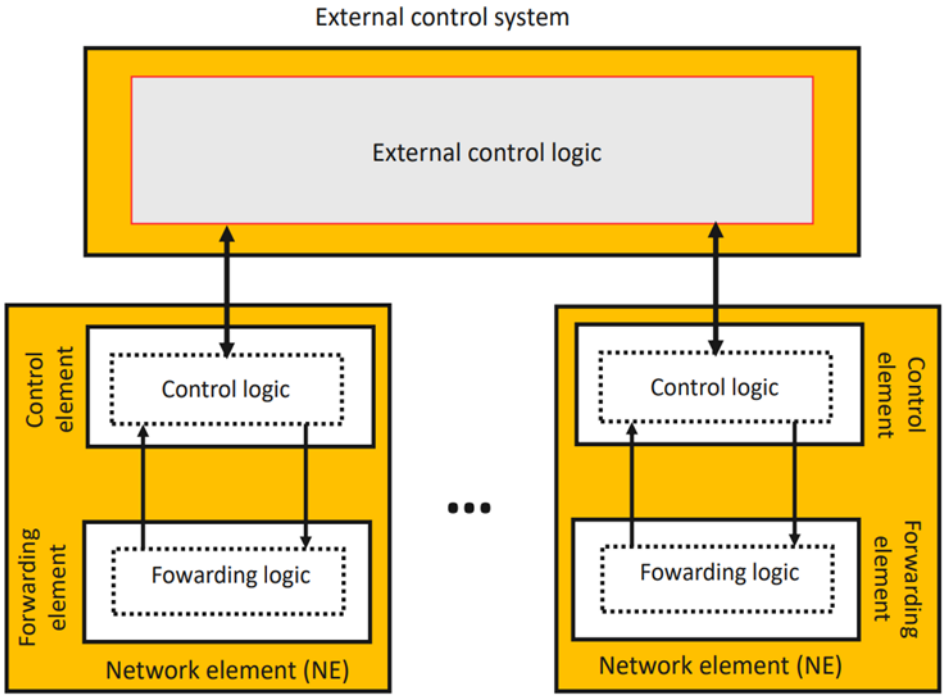


Figure 2.18: External control system, control and forwarding elements.

The control system, which consists of the internal control plane and its extension as an external control plane, is used to control the packet forwarding network, which can be basically the 6G data network on a public network operator. The central problem is the dependability of this control system itself:

- We want to ensure the dependability of the control system, by focusing basically on the reliability and the time-critical properties of the control system, as found in classical dependable deterministic networks standards such as TSN or DetNet.
- Then, later we could study the possibility to generalize the initial solutions to include other dependability properties, especially those related to availability, safety, confidentiality, integrity, maintainability, and any other attribute and/or metric relevant to dependable computing and networking.

We consider the possibility that 6G data packet forwarding networks are controlled and managed according to modern control and management systems based on increasingly complex operations (possibly based on AI/ML algorithms) that may be involved in deterministic dependable data communication service offerings over these 6G data networks. This possibility implies the need of extensible and a potentially important amounts of resources (compute, memory, storage, communications) from an external control system, thus the need to provide edge cloud computing resources to enable these potentially complex dependable control service computations. This possibility also implies the need of an open software programming environment to enable the sharing of edge cloud computing resources between the external network control system and other edge cloud applications, thus the use of the following two SDN design principles: the externalization and the separation of concerns.

The external control system is typically used to complement the internal control system, and to provide additional control resources to overcome the resource limitations of the internal control

system. The external control system can also be used, in some cases, to substitute (or override parts of) the internal control system to liberate resources used by control elements at NE(s). Finally, the external control system can be a centralized or a distributed system. We consider the case in which the control system (both the external and internal control system) belongs to a single administrative entity, and the controlled network belongs to a single administrative entity (e.g., a network operator). The implication and use of an external control system is typically enabled by SDN service architectures.

2.4.1.2 Centralized external control system

The internal control system, consisting of a network of control elements inside the NE(s), is always physically distributed by design. Each control element operates individually and communicates with its neighbors. However, the external control system may be centralized or distributed. In the case of a centralized external control, the control system runs on a single centralized machine, typically consisting of a GPP (possibly with multiple processing units) or several GPP that are integrated in a single physical control machine. In that case, we are faced with something equivalent to a central network controller. The main difference relies on the use of a SDN architecture, and a network programming environment, which provides southbound interfaces for the interactions between the external control system and the internal control system (e.g.: Forwarding and Control Element Separation (ForCES) [IETF10-RFC5810], [IEEE09-802.1Qay], OpenFlow [ONF15-TS025], etc.), interfaces for the implementation of external control logic, and northbound interfaces for the interactions with network applications. In particular, the interfaces for the external control logic can allow the implementation of complex AI/ML algorithms (e.g., intent interfaces) for dependable deterministic network control operations. Some of these control operations may consume a lot of computing resources, and the use of a centralized control system represents a single point of failure (SPoF), which could appear not very scalable. The use of distributed services here appears as a key for solving compute-intensive at scale and SPoF problems.

2.4.1.3 Distributed external control system

In the case of a distributed external control, the external control machine consists of a network of centralized machines, each node of the network (the centralized machine) typically consisting of one or several general purpose processors [GPP] (possibly with multiple processing units) that are integrated in a physical control node. The external control asset forms a distributed machine, which is a (physically) distributed system. However, this external control asset can be exploited in different ways, among others:

- Each of the nodes is operated individually (an autonomous machine), and it communicates with its neighbors to provide external control services.
- At the other extreme, all the nodes are integrated and operated as a single control infrastructure, which is used to run the control services.

In the second case, we are faced with a fully distributed system in which several nodes of the control machine can be involved in the same computations, which requires mechanisms, algorithms, and protocols for the distributed control of these computations. This second case appears particularly interesting for network control services based on AI/ML algorithms, especially those based on distributed ML algorithms running on multiple control nodes to provide deterministic dependability of the external control network. For example, the use of multiple nodes in a single control infrastructure may help in solving the problem of a single point of failure: if one node fails, it may be replaced by some of the other nodes of the control infrastructure. The SDN architecture and network

programming environment may be one of these services requiring the distributed control of computations, while still providing interfaces for its interactions with elements internal to the SDN software, with the internal control system, and with the network control applications.

2.4.2 Reliable distributed control system based on software controllers

Concerning reliability, the main goal of our design approach is basically to avoid that a controller constitutes a SPoF. This means that each NE needs to be controlled by a certain number of controllers (at-least two controllers), so that the failure of a one or many controllers does not threaten the overall reliability of the control system. If a NEs in the transport network (the data plane) is controlled by many controllers, in case one of these controllers fails another controller is always available to control this NE. More generally, if a NE is controlled by a number $P_{\{P>1\}}$ of controllers, then in the worse-case of the failure of P-1 controllers, there is still one controller ready to take over the control role. There is a network between all the controllers in the system, which is used for interactions between these controllers to perform the control operation in a distributed way, either for control operations on a single NE or for control operations on multiple NEs. Figure 2.19 shows an example of such a reliable distributed control system, with four (4) physical control servers intended to control a portion of nine (9) NEs of a transport network. In this example, the physical servers are linked with a simple ring network for the communications between these physical servers. Also, each physical server can typically support the execution of a certain number of virtual servers (or software servers) – a number varying from 0 to a given maximum number of virtual servers for each physical server – running in a software-defined network service environment based on virtual machines and/or containers. Each of these software servers will be dynamically assigned the role of controller of one or many of the NEs, and each NE will have the possibility to be controlled by a number $P_{\{P>1\}}$ of these software controllers (possibly) running on each of the physical controllers, the role of an actual controller being exclusive to one of the P software controllers and being assigned dynamically.

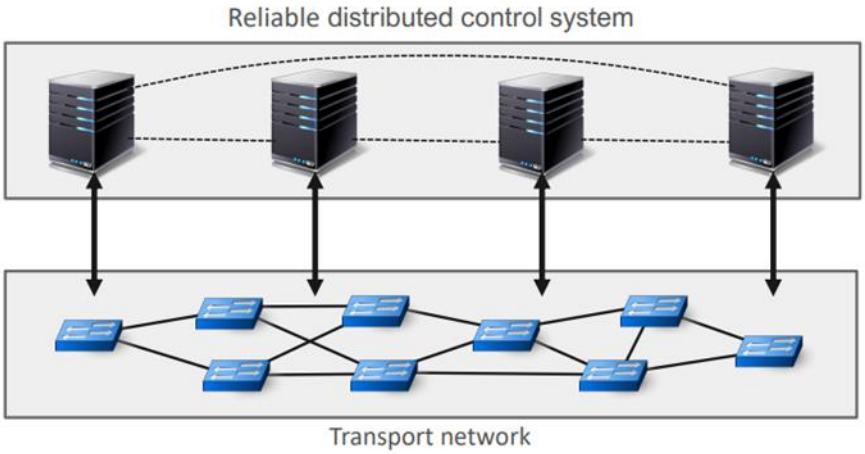


Figure 2.19: Reliable distributed control system for the transport network

The control servers may be used to provide network control and management operations, such as those that can be implemented in a cloud or cloud-native Operational Support Systems (OSS) for data networks for example, based on GPP edge computing resources. The Open Digital Architecture (ODA), which integrates cloud-native technologies as a modern basis for future OSS and Business Support

Systems (BSS), can be used for the control and management of 6G transport and data networks. This implementation and use of ODA-based OSS/BSS services can be seen as a potentially interesting first application domain of the control system design, with would be used to enhance the intrinsic/inherent reliability of 6G transport networks and provide new basis for network management systems, typically those based on the fault, configuration, accounting, performance, security (FCAPS) management model and its evolutions for 6G data networks.

A control server, whether physical or virtual, can control a certain number of NEs. The number of NEs controlled by a control server depends on time (the association between control servers and NEs can change dynamically), the load of that control server, the load of other control servers, and the policy behind the dynamic association of control servers with NEs. The policy behind this dynamic association can have as objective a well-balanced load between control servers. Each NE is controllable, potentially controlled (or covered) by at least two (2) control servers, for the purpose of permanent coverage of the controlled NEs and reliability of the control system. Each NE is also controllable by a maximum number of control servers. For a given network element NE_i, a couple (NE_i, MAX_i) is determined, with MAX_i being the maximum number of control servers that can potentially control NE_i, which depends on operational goals related to the level of reliability of the control system and the permanent coverage of NEs. At any time, each NE is assigned to (and is controlled by) exactly one control server. If the assigned control server fails for example, the assignment is transferred to another control server. The set of control servers that can control a NE are also placed on different control nodes, so that to face the situation in which a control node crashes. The association between the control servers and NEs is made to ensure complete coverage of the NEs, redundancy of control servers, and redundancy of control nodes.

2.4.2.1 Control network elements, communications, coverage index and footprint

Figure 2.20 shows a more comprehensive representation of the control systems, showing a set of control network elements in a distributed control plane, the communication link between these control elements, and a portion of the transport network with five (5) of its elements that are controlled by two of the control network elements.

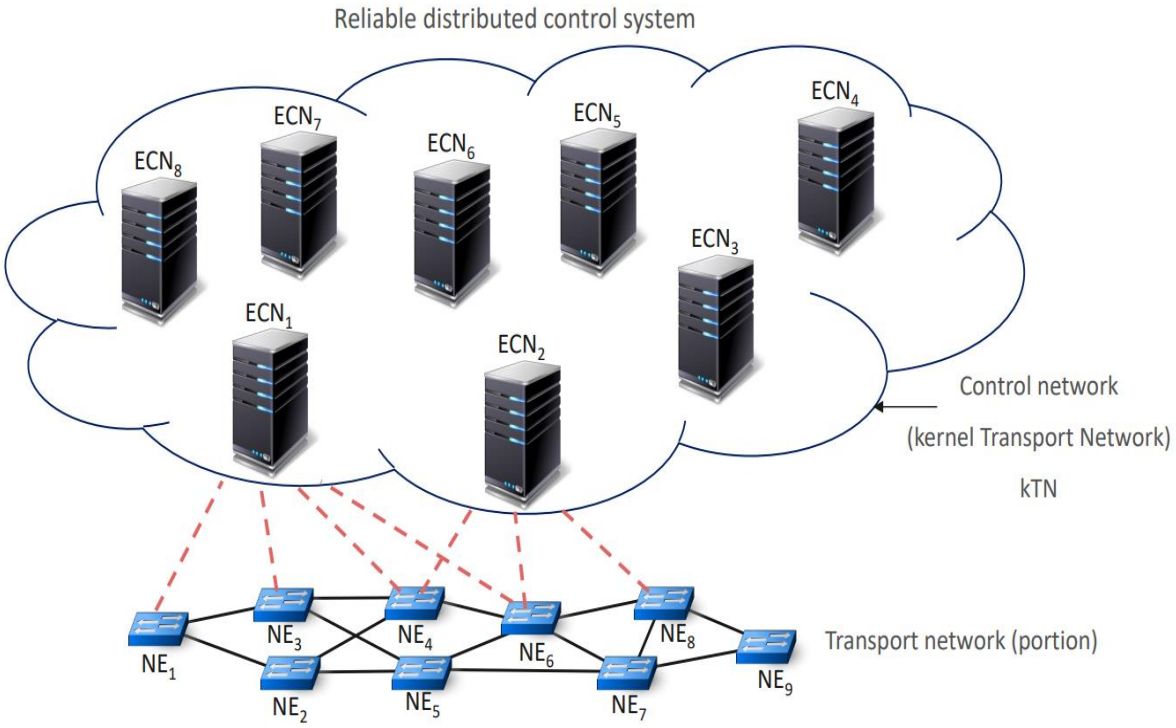


Figure 2.20: Control nodes, kTN, coverage index, footprint

Figure 2.20 is intended to transition from a simplified representation of Figure 2.19, to a more detailed illustration showing a generalized communication medium between the network elements, and the coverage of NEs by control servers, i.e., the possibility that a NE is controlled by a control server during time (or the evolution of the system). The main intention here is to ensure a better and a proper understanding of the basics (or basic design choices) and the basic assumptions behind the underlying control infrastructure. The nodes of the control infrastructure are intended to be edge computing nodes, establishing a direct link between edge computing and the control system. In the figure, it is possible to see eight (8) edge computing nodes (ECN)s, named ECN_1, ECN_2, ..., ECN_8.

These edge computing nodes are individually (each of them) exploited as a standalone edge computing node, and it can interact/communicate with the other ECNs via a control network, sometimes called the kernel Transport Network (kTN) – in particular when referring to the Open Distributed Processing (ODP) terminology. The kTN is a control network interconnecting ECNs, which is used for communications between control servers running on different ECNs, hereby establishing a physically distributed system of ECNs interconnected by the kTN. The kTN itself can be implemented in-band using communication resources of the transport networks, typically in the form of an overlay network of the transport network, or in the form of a network slice dedicated to communications between ECNs and control servers. Finally, the kTN can also be implemented out-of-band, based on dedicated physical network resources, that are not shared with the transport network.

The set of ECNs {ECN_1, …, ECN_8} represented in the figure are the nodes and computing resources of a distributed edge cloud infrastructure used to support the execution of control services. These edge computing nodes can also be shared with other edge cloud services, for example application hosting services in the case the edge cloud nodes are exploited according to the MEC standard. Each of the

ECNs can support the execution of one or several control services (both virtual control services provided by software – virtual machines, containers, and physical control services on physical servers), but for simplicity, the figure shows each ECN as a single control service. The set of ECNs {ECN_1, …, ECN_8} in the figure controls a set of NEs {NE_1, …, NE_9} representing a portion of the transport network in the figure. We define the coverage index of a NE as the number of control servers that can control this NE. For example, in the figure, the coverage index of NE_1 and NE_2 and NE_8 is 1, and the coverage index of NE_4 and NE_6 is 2. We also define the footprint of a control server as the number of NEs that can be controlled by this control server. Another example, in the figure, the footprint of the physical control server ECN_1 is 4, and the footprint of the physical control server ECN_2 is 3.

2.4.2.2 Reliability of the control system and implementation considerations

The notions of coverage index and footprint are important parameters and relevant for ensuring the reliability of the control system, and its proper operations. The coverage index of a NE is a key indicator of the number of control servers able to control that NE that can fail while the NE is still under control. The coverage index is a key indicator of the ability of the system to ensure that, at any time, there will always be a controller available to control a given NE, which is important for ensuring the reliability of the control system. Also, by design, at any time, only one control server is assigned the role of controlling a given NE, also called the elected control server. The elected control server can change following an election decision taken by the control system, typically after the failure of the previously elected control server. Also, the notion of footprint is important for evaluating the consequences of the failure of a control server on the coverage, and for performance considerations. A control server with a "large" footprint will be more subject to performance demands, performance degradations, and the failure of such a control server will have an impact on large numbers of NEs, requiring the triggering of election processes for all these NEs, which have the potential to place important computing loads on the distributed control system.

At any time, only one redundant control server of a NE is active, called the leader. When a leader crashes, a leader election algorithm is activated, and in some implementation cases, this election may require the execution of consensus algorithm between processes running on several ECNs, which can be resource consuming and time consuming. Leader election and consensus algorithms are basic illustrations of the need of distributed control of the computations of the control system to ensure reliability, which require the use of the kTN for the exchange of control messages between ECNs and control system processer, and for the exchange of information collected from the data network.

2.4.3 Towards an edge computing service support

Each ECN is exploited as an edge computing system, with a range of software stacks provided according to the "aaS" model (virtual resources, software development infrastructure and variable number of services platforms, virtual network function (VNF), cloud-native function (CNF) and end-user software hosting and execution). The ECNs conform to edge computing standards, typically but not necessarily/exclusively the MEC standard, which is particularly investigated in DETERMINISTIC6G project. The possibility to exploit ECNs according to edge computing standards that could emerge in the future is considered, especially those based on open-source cloud-native technologies, seeking to define a telecom cloud service infrastructure. The ECNs are also intended to provide computing resources to facilitate the integration of hosted applications with different network services (network control and management services in particular) within shared computing resources.

In order to enable the execution of reliable network controllers in ECN service environments, we need to revisit the architecture and composition of that environment. The impact for a MEC node, for example, will be to provide reliable controllers with a set of VNFs or CNFs, which are special purpose VNFs/CNFs running the software of the reliable controllers. Each reliable controller can be implemented with one or many VNF/CNFs. In the case of many VNFs/CNF, these service environments are typically grouped and tied together as parts of the same entity. In the particular case of CNFs, the containers may be created and manipulated as supports for the execution of reliable network services designed as microservices.

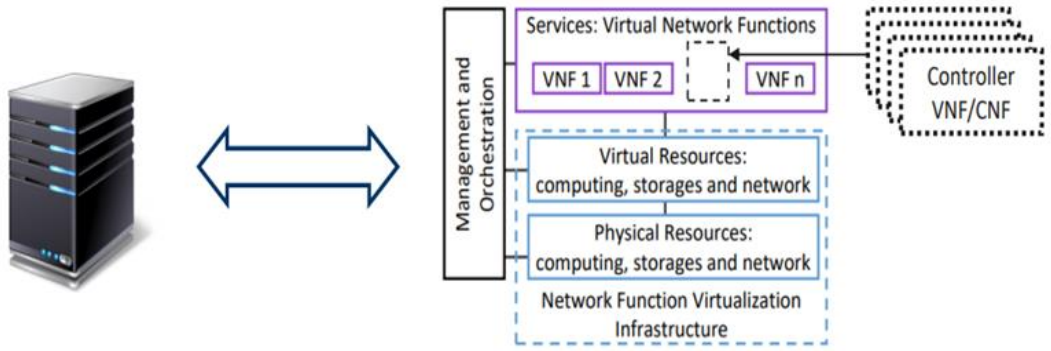


Figure 2.21: Controllers in an edge computing software stack

Figure 2.21 shows an example ECN (left) and the equivalent view of its software environment (right) that is exploited as an edge computing system more or less compliant to the MEC standard. On top of the software environment, the services (virtual network functions) box supports a set of VNFs, numbered VNF_1, ..., VNF_n. These are generic VNF, which could be replaced by a specific VNF in an actual implementation. Following this principle, a subset of VNFs can be replaced by a set of controller VNFs/CNFs as indicated in the figure. Based on this approach, additional software, a reliable controller service support, is required to enable the proper execution of the controller VNFs/CNFs as required. This additional software, the supporting services, can come in the form of system services and libraries used by VNF/CNFs for the distributed control of computations made by reliable controllers. The additional software can also be part of an SDN controller software stack, or software that needs to be added to the management/orchestration block. In a first step, the reliable controllers and their service support are intended to be run on a single ECN. They can be deployed on a single physical server or (at the other hand) on multiple physical servers within the ECN. In this case the kTN interconnects the controller VNF/CNFs.

2.4.4 Extensions of the reliable control system

The reliability of a controller refers to the probability that this controller delivers a correct service, i.e., it controls the NEs consistently, without significant (or perceptible) failure. The use of redundant controllers leads to a coverage index larger than 1, is presented here as a basic approach to enhancing (or improving) the reliability of controllers. However, this redundancy and coverage indexing lead to more complex control mechanisms, which may negatively impact performance and real-time.

Considering reliability and real-time at the same time is even more challenging, it introduces the need to always arbitrate between two different (and sometimes opposite) constraints at the same time. Time-sensitiveness here means that the controller system itself needs to react in real-time, to run its control operations and deliver its control service within real-time constraints. If an elected (or assigned) controller fails, for example, there is an urgent need to elect a new controller (possibly a pre-defined controller, as a special case of elected controller) and keep delivering the control service in time. In the case of important values of coverage index and footprint, the reliability mechanisms can require a lot of processing and also time-consuming communication, especially in case the algorithm behind these mechanisms does not scale very well. To enable reliable control service with real-time constraints, there is a need to develop novel mechanisms, and algorithms, especially those leveraging resource scaling techniques in cloud computing, compute-intensive and real-time techniques, and reliable computing techniques.

In addition to reliability and real-time, the following attributes and related metrics are usually mentioned as key element for ensuring dependable computing and networking:

- Availability, as it is defined in [DET25-D13]. Availability and reliability are very related, usually mentioned and dealt with together.
- Safety, as it is defined in this project, and the related deliverables. Safety is also sometimes seen as a measure of the time to catastrophic failure. In that case safety is therefore a reliability with respect to catastrophic failures. Safety can sometimes be seen as an extension of reliability and treated as such.
- Confidentiality: sometimes defined as the absence of unauthorized disclosure of information.
- Integrity: sometimes defined as the absence of improper system state alterations.
- Manageability: this attribute refers to the ability to repair and modify the system.

There are also several other attributes and metrics related to dependability, some of which are sometimes presented as secondary attributes, which can be derived from those presented as primary attributes: robustness, accountability, authenticity, or non-repudiation.

All these attributes and related measures are related to reliability and time-sensitiveness (real-time) and they have a potentially important impact on the ways to deal with deterministic dependable control services design and their implementation in the future. Their inclusion implies a further level of complexity, and the perspective usage of novel AI/ML-based techniques in many of the approaches to deal with some of these dependability attributes and their metrics.

3 Situational awareness via digital twinning

The concept of Digital Twins (DT) in the context of dependable networks has been initiated in deliverable [DET24-D33] elaborating the main ideas and benefits of the DTs. The main idea of DTs is to achieve the real-time, dynamic, intelligent and evolving representation of the physical world with predictive capabilities which allow systems to forecast failures, test solutions, and provide optimizations without compromising the physical systems. Moreover, the interaction between different DTs has been considered as an important enabler for the optimal integration of different systems. The platform that defines the interactions between the physical and digital worlds including the interaction between the DTs was defined as the Cyber-Physical System (CPS). Regarding the DT topic, in this report we focus on the interaction between the Operational Technology (OT) and 6G systems, and therefore with CPS we consider the interaction between those two systems, those systems with their DT representations and interaction between their DTs. Furthermore, the concept of Situational Awareness (SA), applied to the CPS is defined with three main parts: sensing of the environment and the system, analysis of the data to interpret the current state, and prediction of the future states of the CPS. With those definitions, and a high-level architecture of the CPS including the 6G and OT systems, their DTs and the SA via interaction between those DTs, is shown and discussed in [DET24-D33]. However, the architecture's details have not been discussed so far, and it will be one of the main discussion points of this section. A focus of the [DET24-D33] was also to explain the benefit and needs of the SA using the interaction of DTs on industrial automation use cases, e.g., Automated Guided Vehicle (AGV) path planning based on the joint 6G-OT information. However, the description of use cases in [DET24-D33] was on a conceptual level, while in this section we will focus on the details of the use-case definition, the data collection flow description, protocols and interfaces used for the interactions of DTs for the purpose of SA to optimize either the 6GS based on the information provided by OT DT or vice versa.

3.1 OT Domain DT

As already introduced in [DET24-D33], the concept of DT was proposed in 2003 by Grieves, the first scientific paper about the DT was published in 2011, while the phase of growing interest in the topic has started from 2014 when the first white paper from Grieves was published [Gri14]. From then, there has been high interest from many industrial sectors in the direction of having the digital replica of a system in real-time. Nevertheless, it should be noted that the general concept of a digital representation of a physical system has existed for a longer time in various fields yet not associated with the term digital twin [Mal21].

A major challenge in smart manufacturing is bridging the physical systems and their virtual replicas. Advances in simulation, data acquisition, and communication technologies have strengthened the interaction between these domains and the concept of DTs enables this cyber–physical integration, which has gained significant attention in both academia and industry in the last decade. DTs and big data analytics complement each other in smart manufacturing. DTs integrate data from both physical and virtual worlds across a product's lifecycle, generating large datasets that can be processed by advanced analytics. The insights derived from this analysis can then be used to enhance product and process performance in the physical space. In [TZL+19] the state-of-the-art of DTs until 2019 is provided, where it is stated that the theoretical foundations of DTs come from disciplines such as

information science, production engineering, data science and computer science. The main parts of DTs are: (i) DT modeling, simulation, verification, validation, and accreditation (VV&A); (ii) data fusion; (iii) interaction and collaboration; and (iv) service. According to this overview paper, the focus of DT area of application is as follows: 35% in production, 38% in Prognostics, and Health Management (PHM), 18% in the design and 9% in other areas.

3.1.1 Background on AAS

In the last five years, there has been considerable work in the direction of combining the Asset Administrative Shell (AAS) approach for the implementation of the DT to synchronize multiple DTs and ease the data exchange between different systems. Figure 3.1 shows the asset administration shell, as digital twin of a physical asset.

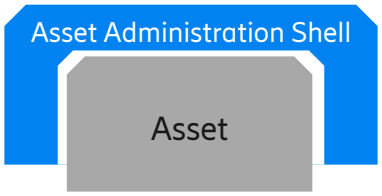


Figure 3.1: The asset administration shell as digital twin of an asset.

The AAS is being standardized in IEC, and it is [PI4022] *"... the digital representation of an asset. The AAS consists of a number of submodels in which all the information and functionalities of a given asset – including its features, characteristics, properties, statuses, parameters, measurement data and capabilities – can be described. It allows for the use of different communication channels and applications and serves as the link between objects and the connected, digital and distributed world."* As depicted in Figure 3.2, the AAS has standardized ways of communicating and interacting with other AAS (see [PI4022]).

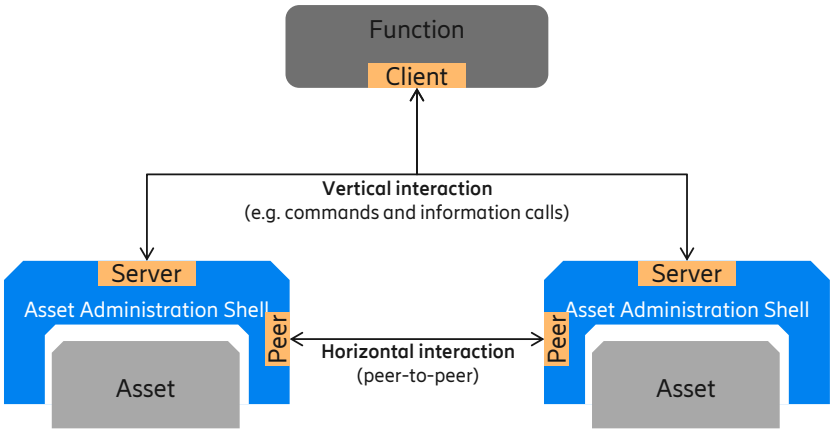


Figure 3.2: Standardized external interactions of the digital twin (AAS).

For complex systems, systems of digital twins or *composite* digital twins (and AAS) can be built [IIC20] [5GA21]. As an example, a component AAS (together with others) can be part of a machine AAS, which can be part of a production line AAS. A separate AAS could be defined for all independent components; conversely, a component that depends on another component should be modeled as a characteristic of the AAS of the component it depends on [5GA21]. For example, an industrial system has many different components of different vendors: machines, robots, conveyor belts, machine vision systems.

Each of them is provided with their corresponding AAS, typically provided by the vendor or the physical asset. To create a digital twin of a larger system, a *composite digital twin* can be created that embeds the digital twins of the subcomponents [IIC20], as shown in Figure 3.3. To this end, it is important that a common interoperable representation is used, as defined by the AAS.

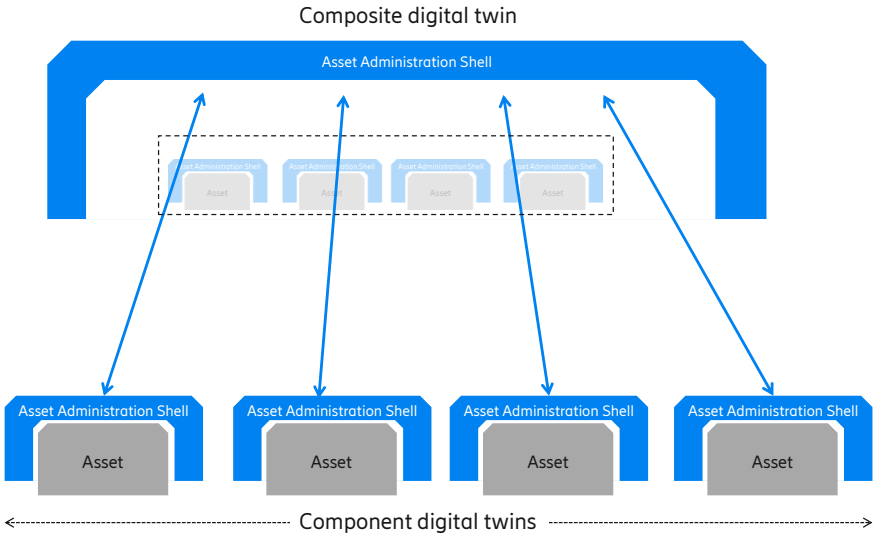


Figure 3.3: Composite digital twin of a larger set of assets, integrating their corresponding component digital twins.

According to the Industrial Digital Twin Association (IDTA): '*the DT is, in contrast to frequently chosen illustrations, not primarily a realistic virtual image of the object under consideration. In terms of effect, it is quite similar to a USB connector that works in multiple applications irrespective of manufacturer because its standard is defined'.* The AAS designed by IDTA implements the DT for Industry 4.0. The AAS creates greater digital value through interoperability simplifying data management for non-intelligent and intelligent devices alike. The AAS represents the complete lifecycle of products, devices, machines, and plants [IDTA]. The AAS is based on a set of submodels that define all relevant data and functions of a specific asset, including its characteristics, properties, states, parameters, measurement data, and capabilities. It enables seamless integration with various communication channels and applications, connecting objects to a networked, digital, and decentralized environment. This allows a multi-vendor exchange of information with industry-neutral standards. The IDTA defines several AAS specifications, such as Metamodel [IDTA-01001-3-0-1], API interfaces [IDTA-01002-3-0-3] or Data specification [IDTA-01003-a-3-0-2], where it is specified in detail which data formats, data types, value ranges, interfaces, protocols, request/response types. AAS needs to be supported in order to interoperate with another AAS. In [IDTA-01001-3-0-1] there is defined a model of an AAS, where a specific AAS defines the list of supported submodels, interfaces, services, user applications, etc. There are three types of data exchange between AASs defined: file exchange, API-based and peer-to-peer interaction using Industry 4.0 language (based on the Metamodel, specified in [IDTA-01001-3-0-1]).

As mentioned before, an AAS consists of submodules, which define the exact parameters and types of variables and data that an AAS complies with, when using a specific submodel. An example submodel, interesting for the scope of this section, is the 'wireless communication' submodel [IDTA-02022-1-0]. The main aim of such a submodel is to integrate industrial wireless communication in the

context of the Industry 4.0 framework. Here, the approach was to design the submodule which contains the parameters related to the wireless communication system. Other submodels targeting a specific communication standard, such as cellular from 3GPP, can be standardized later, based on this submodel. The idea of Industry 4.0 and smart manufacturing is the adapted production to the needs of customers, without the need of planning all the details in advance in a fixed manner. Similarly, wireless communication should not be planned based on the worst case, but such that the wireless communication medium and its use should be adapted according to a use-case. This is only possible with the information exchange between the industrial and wireless systems, and therefore the idea of developing the AAS for the wireless system is beneficial. According to [IDTA-02022-1-0], assets of wireless communication system are divided into two groups: material (routers, base stations, etc.) and immaterial (heatmap, system manager, etc.). Several use-cases relevant for the wireless communication submodel are listed; some of them are: integration of a new wireless device, providing digital version of a wireless sensor from the manufacturer to a customer, replacement of a wireless device, coexistence management, diagnostics and failure analysis, monitoring, managing the automation system based on the wireless system status, etc. The most relevant use-case for the scope of this project is the 'managing the automation system based on the wireless system status', which is described in [IDTA-02022-1-0] in more detail.

3.1.1.1 OT example use-case

In a production environment, there can be several AGVs, carrying goods and equipped with cameras, various sensors, robot arms, servo motors etc. AGVs are controlled from a central station where wireless communication is used for communication with the vehicles. An AAS as the virtual representation of an AGV consists of different submodels representing different aspects of an AGV. Similarly, a wireless communication submodel is part of such an AGV. As explained in previous subsections, virtual representation in the form of AAS with different submodels is beneficial because the AGV itself can make decisions based on the status of external or environmental states. Therefore, the wireless communication module can monitor and capture changes regarding the quality of the communication in specific time frames or in specific areas. A more specific example is dynamic change in the radio environment due to obstacles on the planned path of an AGV, or due to significantly different density of devices in certain areas of interest. In such cases, AGV can change its route planning, change the speed of approaching certain areas, or change mode of operation in specific time frames or areas of interest to avoid communication losses. This directly shows the benefit of the OT system interacting with external systems, such as the communication system, for joint optimization of both systems. In [IDTA-02022-1-0] it is stated that the communication system in Industry 4.0 should not be planned according to the theoretical worst-case scenario, but the interaction with the OT system should be considered, such that rare worst-case scenarios can be avoided. This can significantly increase the capacity of the communication system by decreasing the resource overprovisioning that would be required only to cover some corner scenarios.

It is important to note that the AAS of an AGV is specified such that it can communicate with other AASs, other virtual instances, or other assets/systems directly for parameter negotiation and for example communication system adaptation. An OT system with specification of digital versions of assets compliant with standardized AASs is ready to be part of a CPS with the ability for joint interaction with other systems, such as 6GS, via interaction of OT DT/AAS and the 6G DT. However, the support and readiness of the 6GS in this regard will be discussed in detail in following subsections.

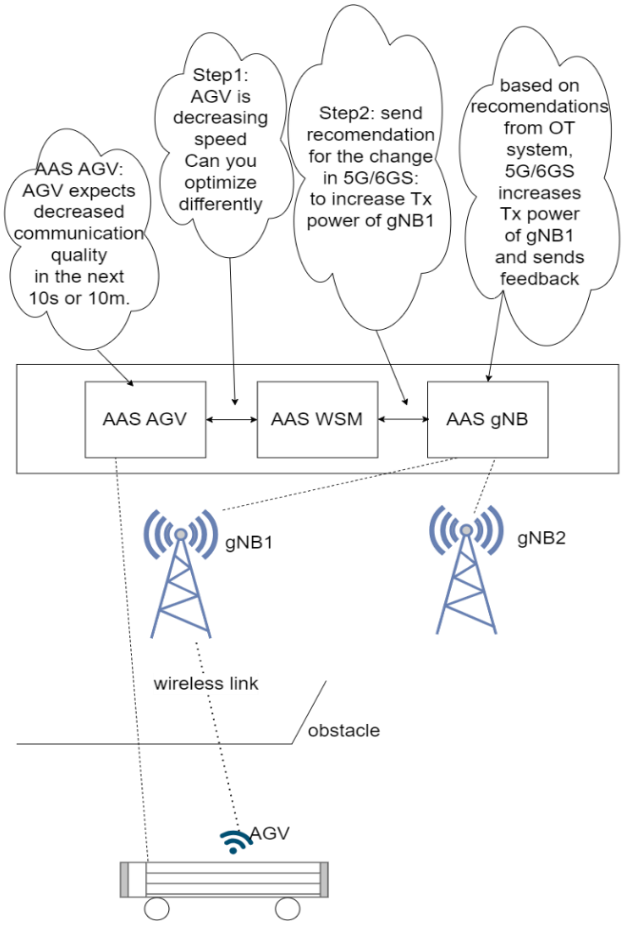


Figure 3.4.: Use-case example from [IDTA-02022-1-0] specification of OT and 5G interaction via AAS/DT interaction for joint performance optimization.

Figure 3.4 illustrates an example use case from the OT perspective, based on the use-case provided in [IDTA-02022-1-0] where the approach of AAS, used as standardized DT implementation, can be beneficial for joint performance improvement of the systems via DT-DT interaction. Here, physical industrial assets such as AGVs are connected wirelessly to the network via gNBs. The physical assets have their digital representations in the form of AASs. In this scenario, there are AASs of physical assets, AGVs and gNBs, and also a Wireless System Manager (WSM) AAS, together used to represent a digital version of the OT system. The WSM AAS here serves as a central hub for communication management, containing data related to communication quality from different assets.

The presented use-case in Figure 3.4 can be described in steps:

- An AGV AAS continuously monitors the quality of the communication channel via its wireless communication submodel. The parameter values of the channel quality are obtained via interaction of the AGV AAS with its physical asset. At a point in time, AGV AAS, based on the parameter values form the wireless communication submodel, detects signal quality degradation.
- AGV AAS preventively sends instructions to the physical AGV to decrease its speed.

- AAS initiates the interaction with the WSM AAS to mitigate the observed quality degradation in order to avoid potential service disruptions.
- Upon reception of the request from the AGV AAS, the WSM sends recommendation to 5G/6GS via gNB (5G/6G) AAS for increased Tx power of a gNB.
- 5G/6GS, based on recommendations from the OT system, increases the TX power of the gNB and sends feedback to the OT system via the gNB (5G/6G) AAS.
- WSM AAS forwards the feedback to the AAS AGV.
- AGV AAS instructs the physical AGV to increase the speed of the AGV (to the original value that was set before the preventive decrease).

These procedures are an example from [IDTA-02022-1-0] presenting the interaction between the AAS (DT) of the OT system and the AAS (DT) of the 5G/6GS for joint optimization, as seen from the OT point of view. However, as in practice the OT and network are often split into two domains with separate concerns, the WSM would send recommendations to the gNB via 6G DT or other exposed services, which will be described in detail in Section 3.2.

3.1.2 Data provisioning

Communication between the physical asset and its AAS is one of the most important aspects to ensure that the AAS truly reflects the physical asset in real-time and that the AAS can send commands to a real asset, if needed.

Referring to the specifications from IDTA, this communication is specified through submodels within an AAS, that are directly linked with the physical asset's data sources, such as sensors, actuators or controllers. There is no specific protocol defined for this communication, but different options can be used, depending on the support of an asset, such as Open Platform Communications Unified Architecture (OPC-UA), Representational State Transfer (REST) API or Message Queuing Telemetry Transport (MQTT) or other methods. The communication can be event-based for assets to push the changes of parameter values to the AAS (via related submodels), pulling-based or subscription-based, where AAS periodically gets measurement data from the physical assets [IDTA-01001-3-0-1]. The data flow from physical assets to AAS submodules can also go over a data gateway/Programmable Logic Controller (PLC) . In such cases, the data from sensors to a PLC can be transmitted over protocols such as Fieldbus, Modbus, PROFINET, EtherCAT, or Ethernet-TSN, while communication between PLC and a submodel in a AAS goes over OPC-UA, REST API or MQTT. If the data to be transmitted from the physical assets is not real-time data, but periodic status/configuration update, the communication can also be file-transfer-based. The data from submodels can be exposed to other submodels within an associated AAS or to other AAS instances via defined external interfaces of AAS, over protocols such as REST API or OPC-UA. The AAS related to a specific asset can be physically integrated into a physical asset, or it can be separated, and deployed anywhere in the network which is connected to the physical asset.

As an example, referring to the use-case illustrated in Figure 3.4 an AAS AGV submodel can subscribe to a specific MQTT topic, regarding specific parameters of the AGV, such as speed and get real-time updates of those submodel parameter values. Those values are published by the related AGV to the same MQTT topic, and other submodels of the AAS AGV or other AAS instances, such as AAS Wireless System Manager (WSM) can get those parameter values via a REST API protocol and perform an action, feedback or request to other AAS, such as AAS gNB for their submodels values or changes in their associated assets such as physical gNB configuration.

3.2 6G Domain DT

Although the concept of DT has been known for years in the industrial domain, the concept of DT Network (DTN) is still at an early stage. The main reference point to follow regarding the DTN is the ITU-T recommendation [ITU-TY3090]. The main scope of the recommendation is related to the functional requirements, service requirements, architecture and security considerations of DTN. The definition of the DTN according to [ITU-TY3090] is that *'A digital twin network (DTN) is a virtual representation of the physical network. DTN is useful for analyzing, diagnosing, emulating and controlling the physical network based on data, model and interface, to achieve real-time interactive mapping between physical networks and virtual twin networks*. According to the definition, four main characteristics need to be part of the DTN: data, mapping, model and interface:

- Data assumes a massive data collection for DTN to precisely replicate its physical network.
- Mapping defines real-time interaction between the physical network and its DT, which is the main differentiation compared to a typical network simulation.
- Model defines the ability of how a network part, or a network feature is represented in a digital world to be able to replicate the real network and to predict the network states in the future using for example Artificial Intelligence (AI)/Machine Learning (ML).
- Standardized interfaces are of utmost importance for the compatibility and scalability of the DTNs and the interactions to the physical network on the one side, and other applications and DTs on the other side.

3.2.1 DTN requirements

In the recommendation [ITU-TY3090], there are several functional and service requirements that a DTN needs to fulfill. Regarding the functional requirements, some of them are:

- Efficient data collection, which assumes the complete (but only required) data, such as operational status, logs, statistics, in a time-aware manner. Diverse tools of data collection, which assume various types of data with different collection frequency, using multiple tools and standardized protocols, such as Simple Network Management Protocol (SNMP) and NetConf. It is recommended to consider efficient data collection in a collaborative way, to avoid unnecessary data replication and potential time synchronization issues between the data samples.
- Unified data repository for storing and managing massive data of different types supporting real-time access in a reliable and secure way. This would allow various network applications to access the data without the need to retrieve it from the physical devices.
- Unified data models to be capable of modeling various network elements for a precise and real-time representation of the physical network with suitable interfaces to collect the data and to serve the services of the DTN to other applications.
- Open and standardized interfaces to avoid hardware or software vendor lock-in and allow for interoperability. The northbound interface is for the interaction with the network applications while the southbound interface is for the interaction with the physical network.
- Management of DTNs assumes the management of data, storage, modelling, deployment and instantiation of entities.

Recommended service requirements for the DTN are:

- Compatibility, to support network elements from different vendors, different types of network elements, collect and manage different types of data, backward compatibility, etc.
- Scalability, to support different scales of networks, to maintain stable performance regardless of the network size, to be able to add new functionalities.
- Reliability, to be able to check the external input and to recognize and mitigate abnormal data and operation and to ensure high availability with backup and data restore options.
- Security, to avoid all possible cyber-attacks and to guarantee data confidentiality and integrity.
- Privacy, to protect the network users' private data.
- Flexibility, in terms of on-demand data collection, storage and exposure to applications.
- Visualization, to have a user-friendly visual interface for all kinds of elements, models and data.
- Synchronization, in terms of time-alignment with the physical network within the acceptable time-range, which is dependent on the type of data and service that DTN provides.

3.2.2 State of the art

Although the concept of the DT was first proposed in 2003, and the first paper about DT published in 2011 [TZL+19], all the papers related to DT in telecommunications have been published after 2020 [KHP+22]. This means that the concept of DT in telecommunications is still at a very early stage, compared to other sectors, in which the research and development started a decade earlier. The main contributions regarding DTNs are summarized in [KHP+22]. According to [VSP+24], the main topics considered regarding the DTNs are network optimization, offloading, floor monitoring, resource allocation, AI/ML, architecture, scheduling, anomaly detection, connectivity and network slicing. Most of the works include computer simulations, some include laboratory experimentation, while just a few include field study or experiments. The majority of papers see the application domain of DTNs in the sector of smart industry, edge computing and vehicular technology, while some see the benefit in the service sector, communications in general, non-terrestrial networks and smart cities. In most of the papers, the AI/ML approaches were suggested for various purposes, such as predictions based on the collected data, AI-driven network management or generative-AI approaches for data augmentation. There are different views of the DT's deployment options, where in most papers the edge and combination of edge and cloud deployment is seen as the most suitable one, while some see cloud deployment as a better option. Similarly, architectural aspects of the DTN deployments are divided, where most contributors see the centralized approach as the optimal one, while others see the distributed or hybrid deployment more suitable [VSP+24]. Most of the contributions published from 2022 follow the architectural and requirements recommendation from [ITU-TY3090].

However, to the best of our knowledge there has not yet been focus on the details of interactions between the 6G DT and the physical 6GS on the level of explaining the exact data flow procedure, interfaces, protocols and the existing network components that can be utilized for the data collection from the physical 6GS to the 6G DT as well as the collection of the data from the outside world, directly or via other DTs, into the 6G DT for joint interaction. In the following subsection, we focus on the architectural options and details regarding the data flows from end-to-end, i.e., from the request for a service from DT, via data collection, to the response, or an action into the physical system. After the discussion on the architectural details, the detailed concepts of the data flow, protocols, interfaces and components included will be explained on a concrete use-case of the interaction of 6G DT with

the external DT (OT DT). Figure 3.5 shows the generic architecture of the 6G DT, based on the DTN architecture from [ITU-TY3090] and presented similarly in [LKD+23]. The three layer in the architecture refer to the physical 6G network layer, 6G digital twin layer and 6G network application layer of the 6G digital twin. Within the DT layer, there are three domains: data domain, model domain and management domain. On one side, DT layer is connected via southbound interface to the physical network for data collection in one direction and delivering control signaling and configuration changes to the physical network, in the other direction. On the other side, DT layer is connected to the application layer via northbound interface to deliver the requirements from applications to the DT, and to deliver the digital copy of the 6G DT, its parts or only specific parameter values to internal 6G and third-party applications.

In the following subsection, we focus on the architectural options and details regarding the data flows from end-to-end, i.e., from the request of a service from DT, via data collection, to the response, or an action into the physical system.

After the discussion on the architectural details, the detailed concepts of the data flow, protocols, interfaces, components included, etc. will be explained through a concrete example of the interaction of 6G DT with the external DT (OT DT).

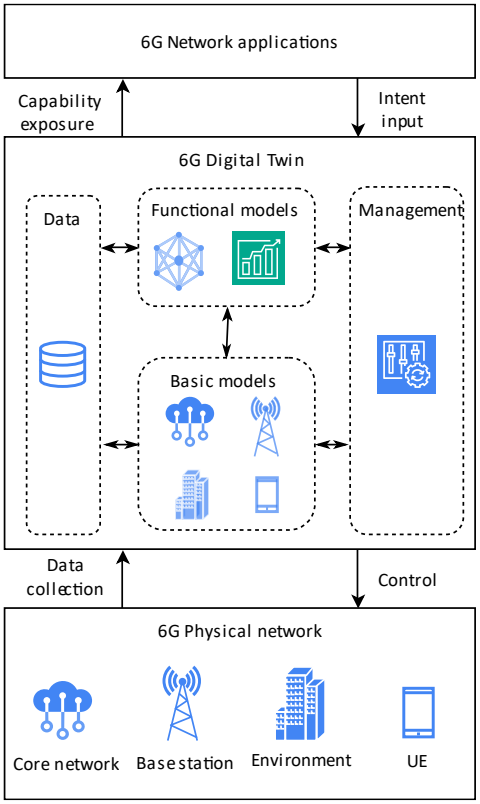


Figure 3.5: 6G DT architecture according to recommendation from [ITU-TY3090]

3.2.3 Architecture aspects

To research different possibilities of interaction between DT and physical twin we first have to place the 6G DT in relation to the 6G system. As already explained, to the best of our knowledge, the existing literature focuses mainly on the internal architecture of the future 6G DT but not on the architecture of future 6G systems with 6G DT. Considering the amount of data flowing in both direction between the two twin parts, it is justified to assume a very tight connection between the DT and the physical twin. At the same time, the 6G network should not explicitly depend on the existence of a DT and should be able to operate without it. One possible integration of DT into the 6G architecture is in the form of a trusted Application Function (AF). The AF is an important NF in 5G networks. It plays a vital role in interaction between application layer and Network Functions (NF) of 5G providing access to the network resources. As an AF, the 6G DT could interact with other NFs through the standardized interfaces to collect data or provide recommendations as feedback. Another important aspect is data privacy and security concerns. Being a trusted AF, the 6G DT is protected by 6G security mechanisms and the internal 6G data is kept secure. Other integration possibilities might exist, but in any case the DT is either an internal 6G entity, e.g., part of the trusted domain, or an external entity that requires controlled data exposure.

3.2.3.1 Data provisioning

The crucial part of operating a DT is having access to the data from the physical twin. The [ITU-TY3090] recommendation specifies the requirements related to data purity and suggests the usage of standardized and lightweight tools for obtaining the data. Our approach into finding the suitable ways to feed the 6G DT with needed data was to first explore the suitability of standardized tools and services already defined in latest 3GPP releases of cellular networks.

6G DT can utilize different interfaces and NFs to obtain relevant data from the 6GS. As defined in [3GPP25-23501], Network Function Service Framework, NF services may communicate directly between NF Service consumers and NF Service Producers. An NF service is one type of capability exposed by an NF (NF Service Producer) to other authorized NF (NF Service Consumer) through a service-based interface. The communication between the NF producer and NF consumer can be established in two possible ways:

- ”Request-response”: Producer NF is requested by the consumer NF to provide a service, which can be an action, information or both. The producer NF can utilize services from other NFs to provide the required service within a given timeframe. Additionally, the producer NF can add the Binding Indication to the response which can then be used by the consumer NF to select the suitable NF producer for subsequent requests.
- ”Subscribe-notify”: Consumer NF subscribes to NF service offered by the NF producer. The producer NF then notifies the subscribed NFs about service results. In this case the consumer NF can add the Binding Indication in the subscribe request which can be used by the producer NF to discover a suitable notification endpoint.

Both these communication options are at the disposal of the 6G DT and can be used for data collection or for controlling requests. As the 6G DT should be a real-time updated model of 6GS, the data is to be collected periodically and continuously from the physical twin. Therefore, the subscribe–notify communication between the DT AF and other NFs should be practiced for data collection. DT AF might also benefit from data preprocessing or data analytics where the Network Data Analytics Services

[3GPP25-29520] and Data Collection Coordination Services [3GPP24-29574] can be used. In Release 16, 3GPP introduced a reference architecture for gathering data from 5G network and making it available to downstream consumers. As specified in [3GPP25-23501] and [3GPP24-23288], SA2 has defined the Network Data Analytics Function (NWDAF), which collects and analyzes data from NFs and UEs within the 5G System. The processed insights are then shared with subscribing data analytics consumers. NFs participating in this data analytics framework implement a service-based API for event exposure. NWDAF subscribes to specific events published by NFs to acquire relevant data, which it subsequently analyzes and makes available to its subscribers.

DCCF

In Release 17, the 5GS architecture was enhanced to further improve and support the network data analytics services. A new Data Collection and Coordination Function (DCCF) was added to coordinate the collection and distribution of data requested by NF consumers. The main purpose of DCCF is to prevent duplicated data subscriptions on NFs for the same data and therefore remove redundant notifications. When the DCCF receives a data request, it assesses the data collection status from the data source. If the parameters in a data consumer's request match those of a previous request or a registered data collection profile, the DCCF may conclude that the requested data is already being collected from a data source. Alternatively, it may determine that an existing subscription to a data source can be adjusted to also fulfill the new data consumer's request. If the DT AF is to rely on data analytics services for data collection, then the usage of DCCF services is recommended while DT AF will most likely subscribe to the majority of NF events. There are multiple data sources from which DT might need sporadic or continuous data collection:

1. Core NFs such as Access and Mobility Function (AMF), Session Management Function (SMF), Policy Control Function (PCF), Unified Data Management (UDM), Network Exposure Function (NEF), AF, Network Repository Function (NRF), Network Slice Admission Control Function (NSACF) and User Plane Function (UPF),
2. Radio Access Network (RAN),
3. User Equipment (UE),
4. external systems

Data collection from NFs

[3GPP24-23288] lists NFs that are possible data providers for the NWDAF analytics. The list of events that can be captured is defined in [3GPP24-23502] in the tables shown in the Table 3.1 in the third column. The DT AF would have a choice to subscribe to each of these events individually via DCCF or to subscribe to the provided analytics provided by the NWDAF that are based on generated events.

Table 3.1: NF Services consumed by NWDAF for data collection

Service producer	Service	Reference in TS 23.502
AMF	Namf_EventExposure	5.2.2.3 5.2.3.5
SMF	Nsmf_EventExposure	5.2.8.3 5.2.3.5
PCF	Npcf_EventExposure (for a group of UEs identified by an Internal-Group-Id or any UE) NpcfPolicyAuthorizationSubscribe (for a specific UE)	5.2.5.7

UDM	Nudm_EventExposure	5.2.3.5
NEF	Nnef_EventExposure	5.2.6.2
AF	Naf_EventExposure	5.2.19.2
NRF	Nnrf_NFDiscovery Nnrf_NFManagement	5.2.7.3 5.2.7.2
NSACF	Nnsacf_SliceEventExposure	5.2.21.4
UPF	Nsmf_EventExposure or Nupf_EventExposure	5.2.8.3 5.2.26.2

Data collection from RAN

Operations, Administration and Maintenance (OAM) in 5G networks plays a crucial role in ensuring the efficient management, monitoring, and optimization of network functions. The 3GPP-defined OAM framework provides mechanisms for fault management, configuration management, performance monitoring, security management, and lifecycle orchestration across various 5G Core (5GC) and Radio Access Network (RAN) components. Unlike previous generations, 5G OAM is designed to support virtualized, cloud-native network architectures, making it more scalable and flexible. OAM functions are implemented within the Management Service (MnS), which interacts with different NFs and external Network Management Systems (NMS) to enable automation and intelligence-driven network operations. One of the key responsibilities of OAM is collecting and distributing network performance and operational data to various consumers that require analytics-based insights for real-time decision-making. OAM data is typically collected using protocols such as NETCONF, SNMP, HTTP/REST APIs, or gRPC. Data exposure can also be achieved through event-based mechanisms. To collect analytics for OAM, DCCF must first discover the data source and then proceed with data request as shown in Figure 3.6. The DCCF uses MnS Discovery service to discover the correct data producer. Then it sends a reporting request and, after receiving the data, it forwards it back to DT AF.

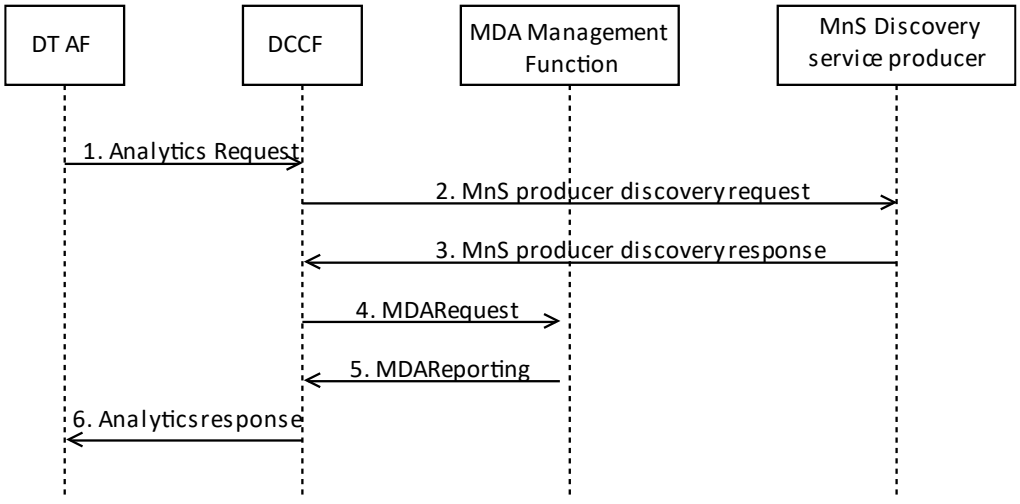


Figure 3.6: Subscribing to OAM data.

Data collection from UE

For the purpose of data collection from the UE Applications, new architecture enhancements are introduced in [3GPP24-26531] which explain how new Data Collection AF acts as centralized entity responsible for provisioning data collection configurations to UEs and processing the collected data. It interacts with various network functions to facilitate efficient data reporting and analytics. The UE application has to establish a connection to the data collection AF via an existing Protocol Data Unit (PDU) Session. This can be either a direct connection or an indirect one via an Application Server (AS) which can also reside inside or outside the trusted domain. Upon connection, the data collection AF stores the IP address of the UE and uses it to request data collection. The data collection AF registers its profile with NRF and specifies the events it can provide.

Data collection from external systems

As it is explained in the [DET24-D33], the 6G system can greatly benefit from external data such as planned activities, sensing data, mobility patterns, traffic requirements, etc. One option to access this data is during the request-response interaction with OT DT where OT DT can provide limited number of parameters in the API call. Another possibility could be continuous data collection via subscription-based communication. However, this would require DCCF subscription to external non-trusted AF. [3GPP24-23288] describes the procedure used by the NWDAF or DCCF to collect information from AFs via the NEF. However, the prerequisite for this is registration of external AF to the NEF. This is done via OAM configuration at NEF by providing the AF, the OT DT AF, collectable data information such as AF identification, AF service identification (e.g. endpoint information of Naf EventExposure), available data to be collected per application (e.g. identified by Event ID(s)). After this, NEF generates an event exposure with new EventID to be associated with available data to be collected from AF and uses Nnrf_NFRegister or Nnrf_NFUpdate service from NRF to create or update its NF profile that now contains new analytics from external AF. On success, 6GC NFs including DCCF can call Nnrf_NFDiscover service of NRF to determine the subscription endpoint at NEF for receiving external data.

Software probes

5G data analytics framework provides valuable insights but may not always offer the granularity or real-time adaptability required for modern, data-driven architectures. To address this, software probes have emerged as a complementary or alternative solution, enabling fine-grained, real-time network observability and analytics-driven optimization. Software probes are lightweight, distributed monitoring tools that could potentially be deployed within 5G NFs, including e.g. the RAN, UPF, AMF, and other core components. Such probes could capture real-time network data, including packet flow metadata, Quality of Service (QoS) metrics, latency, jitter, and congestion levels, enabling AI/ML-driven network optimization and automation. Compared to the centralized approach with DCCF and NWDAF, probing is a distributed system where each network component has its own probe for data collection. A drawback of this approach is that it would need to be integrated into the product realization of different network vendors, which would require standardized functionality to provide a common and interoperable analytics framework.

3.2.3.2 DT AF

Following the 6G DT architecture from [LKD+23] that is based on [ITU-TY3090] recommendation, our focus is to define southbound interfaces for data collection and control, as well as northbound communication toward applications that utilize the DT. In the southbound interfaces, the DT AF could

consume the services of DCCF. As explained, the DCCF can provide periodic data notifications, thus providing either already processed data in the form of analytics from NWDAF or direct singular event notification from other NFs. DCCF is a suitable data source for a DT as it also ensures that collected data is not duplicated and can also provide data filtering and manipulation. An additional important aspect is the interaction of the 6G DT with other DTs, such as the OT DT, to gain situational awareness benefits as presented in [DET24-D33]. Accordingly, we are considering two architectural options that mainly differ based on the interaction of 6G DT with its consumer applications.

In the first option, illustrated in Figure 3.7, the 6G DT as an AF has internal applications for intent validation, network management, visualization, optimization and others. These applications can communicate directly with the DT via its REST API interfaces resulting in responsive and secure usage of DT services. In this case the OT DT or parent DTN and other third-party applications are seen as external. As such, they can only interact with 6G DT services via the NEF. The provider of 6G DT decides which services or applications are exposed to external applications and registers DT AF profile at NRF accordingly. The DT AF collects data via DCCF using subscription-based communication. In Figure 3.7 this is represented with a dotted line.

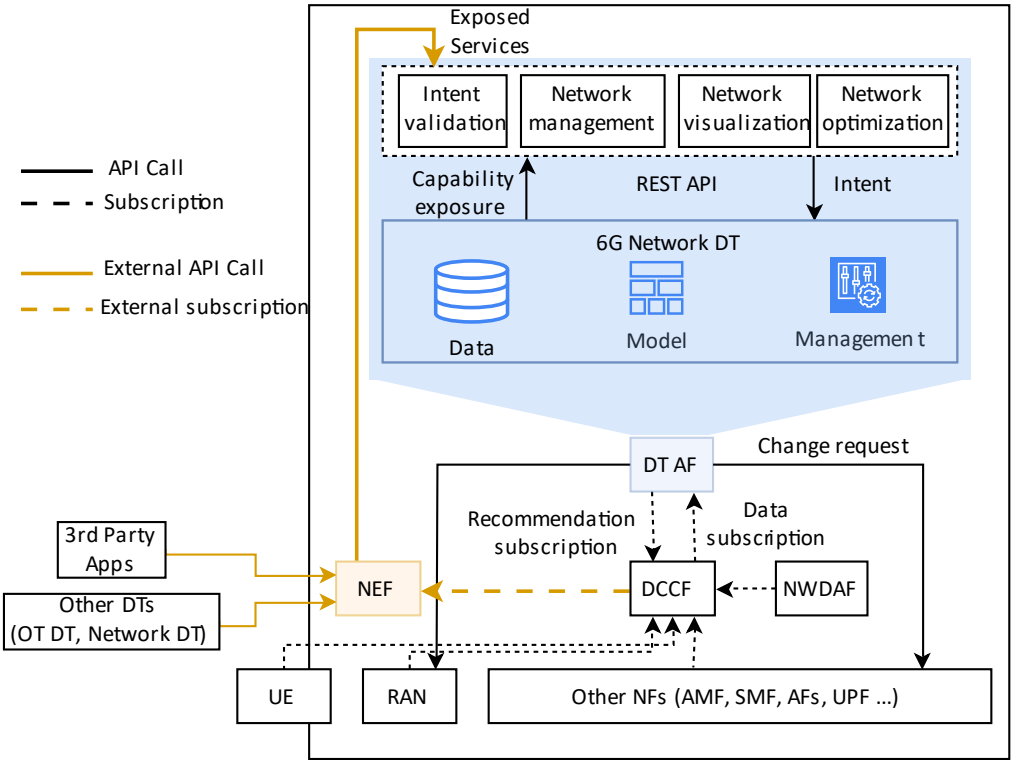


Figure 3.7: Architecture option with 6G DT applications inside 6GS.

The second option focuses more on the service exposure, as shown in Figure 3.8. In this case the REST API endpoints of 6G DT are directly exposed to the NEF and all applications that interact with the 6G DT are considered external. Although the 6G DT service consumers are non-trusted applications, the DT remains a trusted AF within the 6GS. Data privacy and ease of communication with physical twins

remain unchanged. The benefit of this approach is an easier interaction between OT DT and consumer applications of 6G DT as they are out of 3GPP scope. This requires, according to network exposure capabilities, to exchange relevant information between the 6G DT and third-party applications, like the OT DT or functions for network optimization. Certain functions such as Network management and Network optimization will always be in the trusted domain, however, additional functions can exist in the untrusted domain and provide assistance information or service intents to the trusted ones.

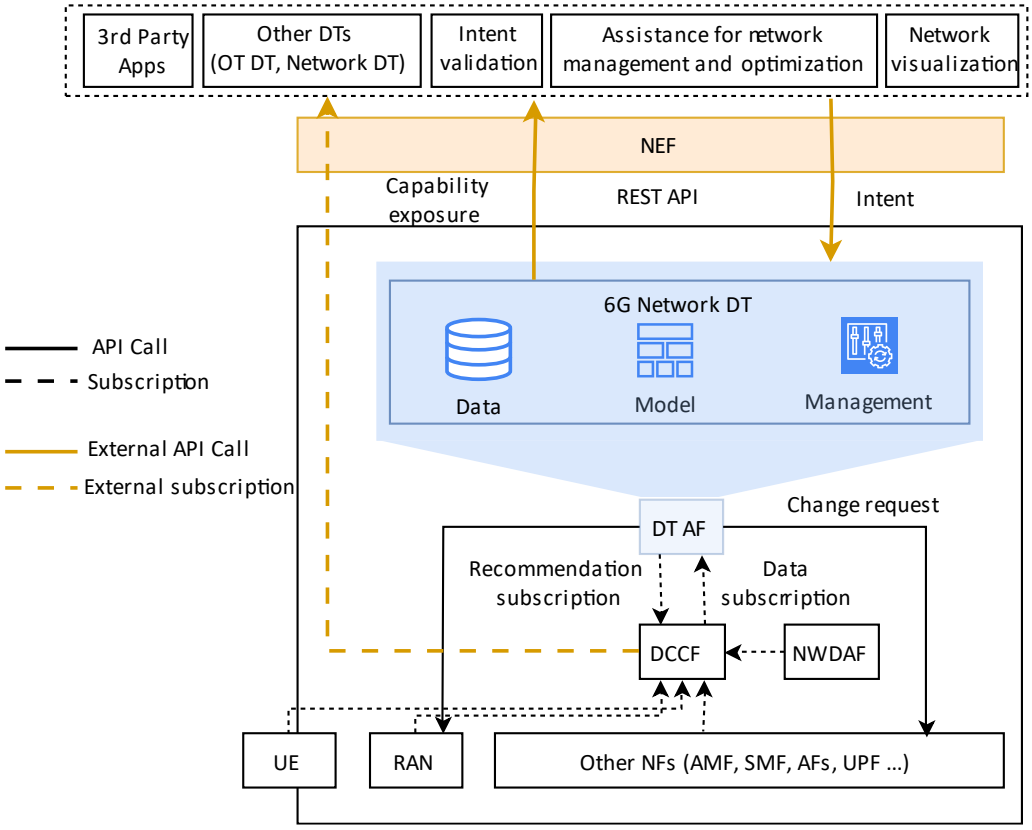


Figure 3.8: Architecture option with 6G DT applications outside of 6GS.

3.2.4 Maturity gap compared to OT DT

In industrial applications, the AAS serves as a standardized digital representation of physical assets, enabling seamless integration, data exchange, and lifecycle management. In contrast, the adoption of DT technology in the networking domain is still evolving, with varying levels of standardization, implementation, and real-world deployment. The current state of DT in the network focuses mainly on architectural aspects and use cases; gaps and the need for standardization are still to be investigated. One question can be whether certain practices and advancements of DT technology in industrial application are relevant and can be adopted in the DTN. Specific AAS submodels for wireless communication could indicate a need for a standardized description of networking assets. Current AAS network models are defined for the industrial domain; if AAS models would accordingly be defined to describe the network domain, this would simplify interactions in between an OT DT and a 6G DT. The granularity of an AAS model for a 6G network requires further consideration. While it could comprise a model on node level (such as components for RAN, 6GC, UE), it would likely enable more

meaningful interactions with the OT domain if it would describe the network regarding its capabilities and characteristics as useable by the applications. This could, e.g., be a map of supported performance levels, network utilization, etc. Such models could be populated with timely network information obtained from the network operation.

3.3 Synergy between OT DT and 6G DT

In the previous deliverable [DET24-D33], the benefits of interaction between OT DT and 6G DT are explained. Different forms of situational awareness could be obtained, and different optimizations of network and production could be applied. Interactions between different DTs involve the exchange of contextual data and can leverage AAS to ensure interoperability across various subsystems. Rather than transmitting raw data, this exchange focuses on SA information, which must capture the mutual impact between systems while enhancing the overall reliability and adaptability of the CPS. Although individual DTs can access large volumes of data, the collection of contextual and situational information should be purpose-driven and value-oriented, ensuring that only relevant insights are shared across domains. This approach enables optimization, where the OT system accounts for 6GS in its planning, and vice versa, fostering a more synchronized and efficient interaction between both domains, as illustrated in the Figure 3.9. Similar thinking was found in the OT community as explained in earlier sections about AAS and wireless communication submodels. They also envision that this interaction could lead to finer control of industrial assets but also network assets such as RAN. Additionally, in the ETSI [ETSI24-015] report about zero-touch networks and service management, the synergy between industrial DT and network DT is mentioned. The focus there is on the predictive behavior of networks that could result in planned network reconfigurations or modification of automation control rate in order to support critical services. This leads to the conclusion that it is the right time to explore the practicality of proposed interactions, and in the following section, we try to close the communication circle between OT DT and 6G DT with existing standardized tools covered in previous sections.

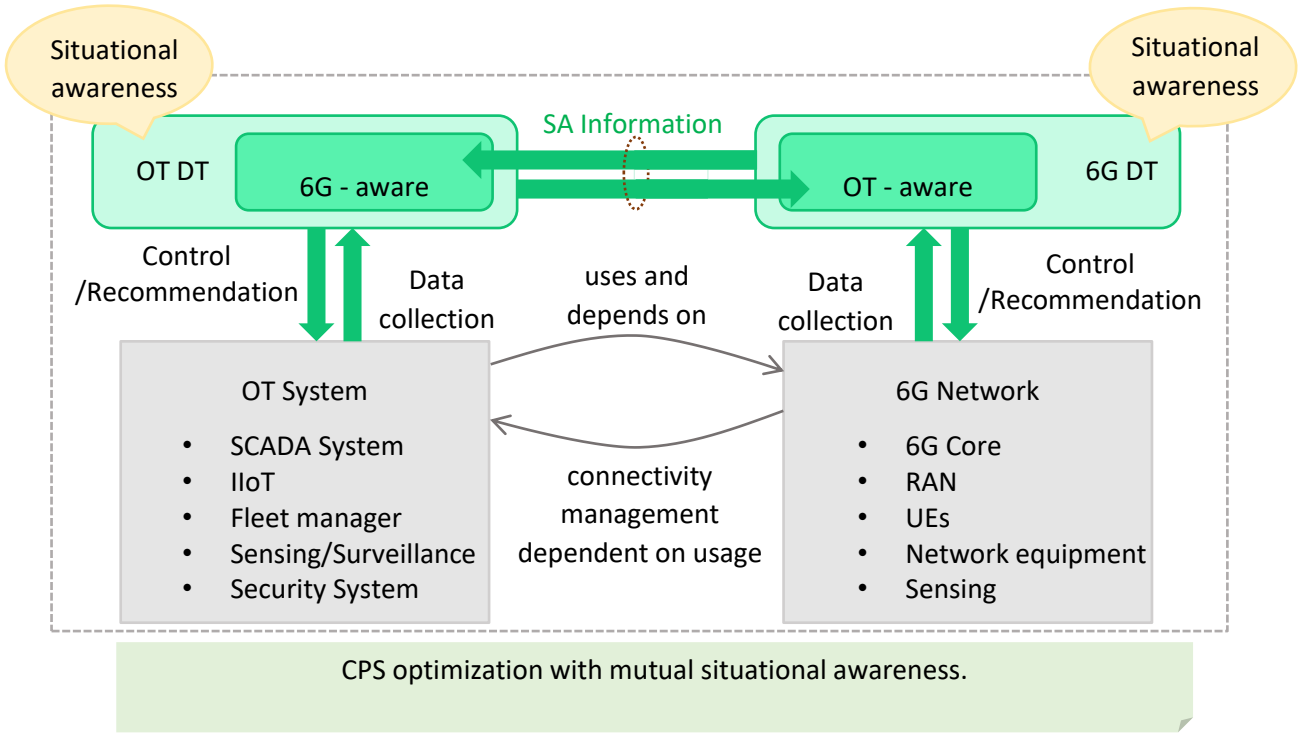


Figure 3.9: Interaction between OT DT and 6G DT

3.3.1 Data exchange on an example

Following the previous work in [DET24-D33], this section shows an example of a full circle communication loop including architecture aspects we discussed in the previous sections. We assume that the OT system has to execute a new task which requires the operation of 4 more AGV/Autonomous Mobile Robots (AMR) in the area of interest as shown in Figure 3.10.

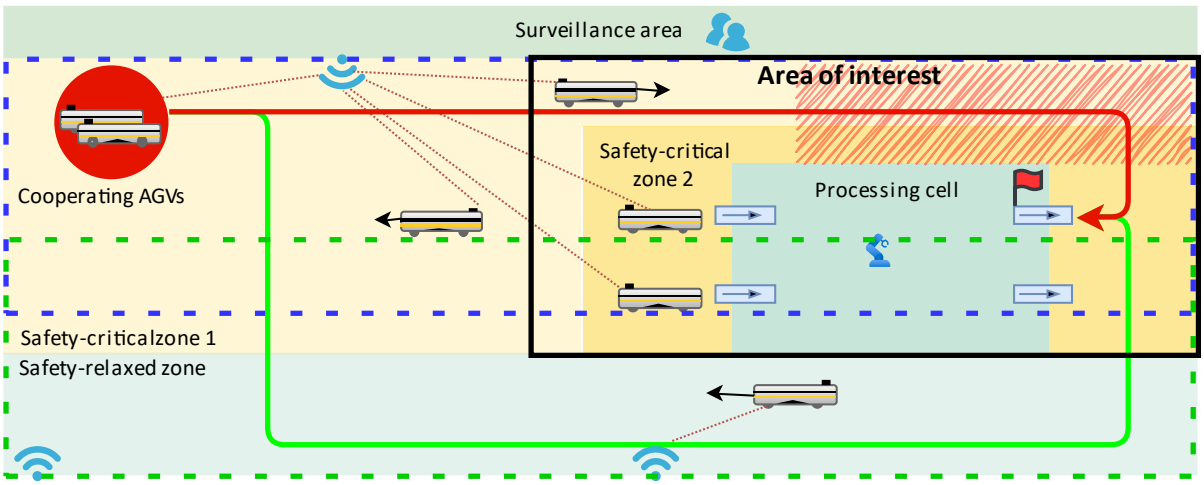


Figure 3.10: Use case illustration

As it is shown in Figure 3.11, which represents a flow diagram of this use case, the OT planner uses its OT DT to estimate the outcome of a new task. By assuming the architecture shown in Figure 3.7 the OT DT has to use the services of NEF to access and use the services of 6G AF DT. Prior to this, the DT AF service endpoints must be registered with NEF. To the best of our knowledge, this cannot be done

through dynamic signaling, but instead via configuration of OAM. OAM administrator registers DT AF with NEF by providing application ID, endpoint specification, access rules and API definition. The API call can then be forwarded to the DT AF via the NEF and in this use case the OT DT could use the Intent validation service. One of the most important assets that 6G DT brings to 6G network is the ability to simulate the outcome of different operation and reconfiguration scenarios. Therefore, intent validation service is often mentioned as a service to simulate or predict the impact of intended changes on the running network. The intent validation service can be one of many services offered by the 6G DT. For its operation, location dependent network insights are relevant, which could be based on parameters such as: area of interest, UL and DL throughput requirements, UL and DL latency requirements, reliability requirements. In the future, such network information insights could be provided by, e.g., NWDAF. A prerequisite for 6G DT and OT DT interactions is a common understanding of the area of interest. In 3GPP the area of interest can be represented by a list of Tracking Areas, list of cells or list of RAN node identifiers or with geographical area with multiple longitude-latitude pairs that define a convex polygon [3GPP24-28537]. The OT system has no notion of network specific parameters such as tracking areas, list of cells and therefore the area of interest would be specified with longitude-latitude coordinates. The intent validation application could use an DT REST API to request an estimation of the maximum number of UEs that could be supported in the area of interest based on the current network state and resource utilization in the focused area. A 6G DT can use different data to make the estimation or even a prediction and the basic ones are shown in the Table 3.2 together with data sources.

Table 3.2: Parameters for estimating number of supported UEs in an area of interest

Parameter	Source	Description
Packet delay	OAM	OAM can provide average packet DL/UL delay or distribution of DL/UL delay between NG-RAN and UE
Physical Resource Block (PRB) usage	OAM	OAM provides total available PRB in DL and UL as well as total PRB usage or distribution of total DL/UL PRB usage
UE throughput	OAM	OAM can provide average DL and UL throughput in gNB or distribution of DL throughput
Packet loss	OAM	gNB measure the DL and UL packets loss including any packets not successfully transmitted or packets successfully received but delayed more than a delay threshold at Uu transmission
Channel Quality Indicator (CQI) / Signal to Interference Noise Ratio (SINR)	OAM	OAM can provide CQI related measurements including CQI distribution, MCS distribution PDSCH and PUSCH, SINR Distribution in gNB [3GPP25-28552]
Number of UEs	AMF	Number of UEs in an area of interest
Location of UEs	LMF/OT DT	Current geodetic or local location of target UE obtained from OT DT or Location Management Function (LMF)
Throughput requirement	OT DT	Data from current analysis request but also historical data from previous requests
Latency requirement	OT DT	Data from current analysis request but also historical data from previous requests
Reliability requirement	OT DT	Data from current analysis request but also historical data from previous requests

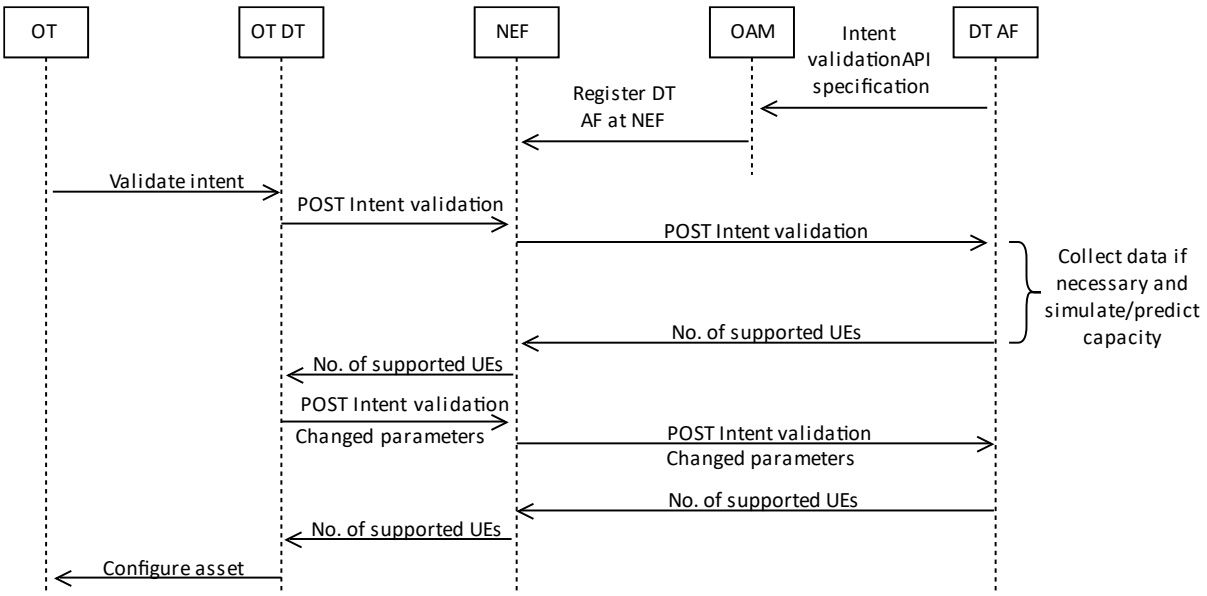


Figure 3.11: Call flow for intent validation.

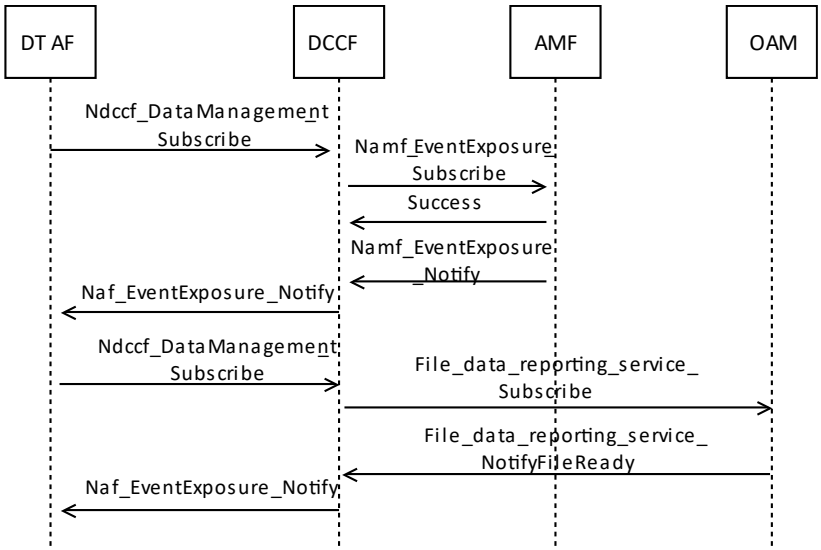


Figure 3.12: Data collection from DCCF

In the following we exemplify the data collection and network analytics to provide appropriate network insights, following the design of NWDAF. The desired data is being continuously collected via DCCF. To trigger the data collection the DT AF uses Ndccf_DataManagement_Subscribe service and provide parameters to identify data or analytics to be provided together with optional formatting and processing instructions that specify how data will be delivered as shown in Figure 3.12. The following parameters are required:

- Service operation – identifies the services to request the data from:
  - Namf_EventExposure_Subscribe – for collecting the number of UEs data from AMF.
  - Nlmf_Location_DetermineLocation – for collecting current UE location data from LMF.
    - Nnef_EventExposure_Subscribe – for collecting asset location from OT DT. EventProducer has to be registered in NEF profile via OAM configuration.
  - Subscribe – for OAM data, subscribe API of different OAM services such as generic performance assurance and fault supervision management services, performance management services, fault supervision, management data analytics services or file data reporting services can be used. [3GPP24-23288].
- Data specification – service operation required and optional input parameters to identify the data to be collected (EventID, Target of event reporting, Event filter, etc.).
- Notification Target Address – DT AF function IP address as the data should be notified to the DT AF.

There are also optional parameters such as:

- Time Window – start and stop time when the data should be collected. The period can also be in the past which indicates the collection of historical data from Analytics Data Repository Function (ADRF) or NWDAF.
- NF ID(s) – specific NFs from where to collect the data.

The area of interest can be arbitrary, and the OT DT can initiate multiple intent validation requests. This means that the total factory floor can be partitioned into smaller areas of interest. By combining these smaller chunks, the OT DT can request the number of supported UEs on different paths through the factory as shown in Figure 3.13.

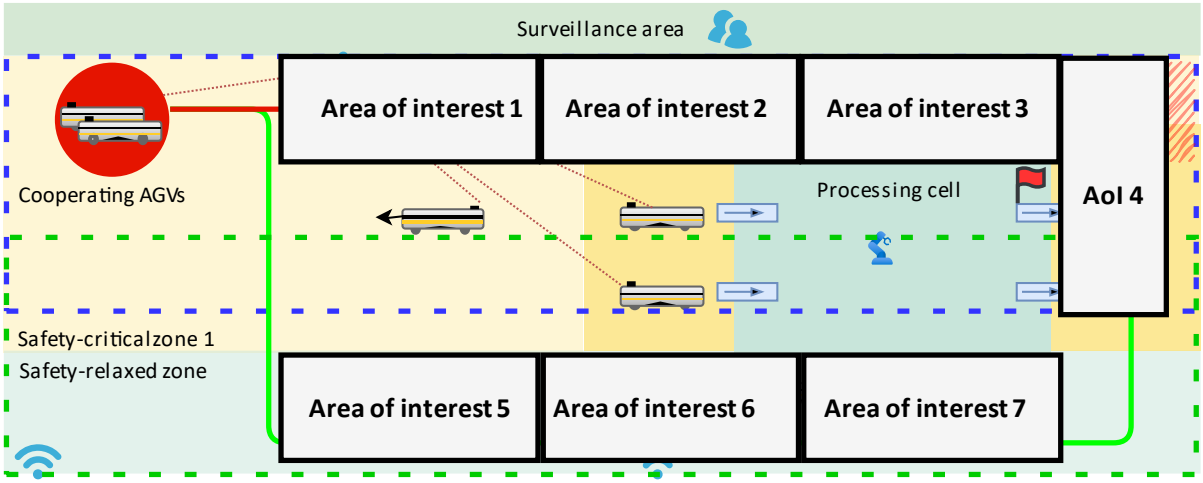


Figure 3.13: Areas of interest - partitioning of the factory floor

First, to determine the number of supported UEs in the area of interest, the available resource capacity must be determined. Used and total number of PRBs can be good indicator of network utilization. However, to translate available PRBs into a number of supported UEs, UE communication requirements have to be accounted for. Based on the reliability requirement, DT AF can use the same algorithm as 6GS to determine the required QoS or 5G Quality of Service Indicator (5QI) mapping. The next step is to use CQI measurements to estimate the average signal quality in the area of interest and determine the maximum Modulation and Coding Scheme (MCS) that can be used. The measurements do not necessarily need to be taken from the physical network but instead could be obtained from the DT service which simulates the radio propagation in the modeled environment, such as a ray-tracing simulation. The benefit of using the results from the simulated radio propagation is that the channel quality parameters can be estimated in the current and future time instances and in locations where there are no available measurement data. In case that the past measurements values are used directly, it could happen that the physical environment was different at the time of collecting the measurements compared to the environment at the time instance of interest in the future while with the simulated radio propagation, the predicted environmental changes could be taken into account as well. Combining the maximum MCS with determined 5QI, the required number of PRBs for a new UE can be calculated and therefore also the number of UEs that can be supported.

Similarly, the output of simulated radio propagation can be used as an input to a network simulator as part of a DT service for predictions to simulate the network behavior in future time instances, areas of interest with desired number of UEs in order to estimate the capacity of the network in terms of predicted latency and TP parameters.

As shown in Figure 3.11, the OT DT can initiate multiple analysis requests with different communication requirements, especially if the asset supports different operation modes. The OT DT uses the analysis response from 6G DT to configure the task execution in the best possible way. The

results obtained from 6G DT can show preference of one navigation path over another or indicate the need for modified operation in order to reduce the load on communication system.

4 Conclusions

As demonstrated in the first report on Edge computing [DET24-D33] for time-critical applications, Mobile Edge Computing (MEC) offers a substantial advantage in reducing latency compared to traditional centralized cloud solutions. This reduction in latency is crucial for applications that demand real-time responses and can be sufficient to meet the stringent requirements of certain time-sensitive applications. By bringing computational resources closer to the end-users, MEC minimizes the distance that data needs to travel, thereby enhancing the speed and efficiency of processing tasks. This capability makes MEC an indispensable component for applications where every millisecond counts, such as autonomous driving, and industrial automation.

In scenarios where the IEEE 802.1Qbv scheduled traffic standard is either required or beneficial, robust support for data plane operations, such as those offered by technologies like eBPF, is essential within the edge domain. The integration of Time-Sensitive Networking (TSN) applications necessitates precise application scheduling and traffic handling to ensure deterministic latency and minimal delay variation. Implementing support at the data plane level in the edge domain allows for seamless packet scheduling and the effective management of time-sensitive traffic, thus supporting the stringent requirements of TSN applications.

To effectively manage and configure the specific internals of cloud deployments for TSN support, it is necessary to abstract the control plane of the cloud ecosystem. This abstraction facilitates the integration of legacy TSN domains with cloud compute domains, overcoming the current lack of standardized methods for exposing cloud characteristics to TSN control planes. By modeling cloud host deployments according to IEEE standards, a TSN-aware cloud management entity can explore cloud deployment characteristics and configure tailored traffic handling solutions based on 802.1Qbv scheduling plans. This approach ensures that the cloud infrastructure can efficiently support TSN applications by aligning cloud-specific processes with TSN requirements.

We explored the involvement of mobile network operators, specifically the scenario of hosting TSN applications within the mobile operator-hosted edge domain. This integration of edge and TSN domains presents multiple options for effectively configuring both 6G and edge cloud components. Among these options, the SEAL Network Resource Management framework-based solution emerged as the most promising for facilitating TSN control plane configurations across both domains. This method offers a cohesive approach to manage and optimize network resources without imposing major standardization impacts. While detailed design considerations for this scenario remain an area for further development, it is crucial to ensure that proposed solutions align with existing standards and industry practices to promote widespread adoption and interoperability.

Regarding the Digital Twin (DT) topic, the current state of the standardization and development in Operational Technology (OT) and network domain, especially 6G, has been explored. Due to a decade

of difference in the research and development between the OT DT and Digital Twin Network (DTN), the maturity gap between those two domains has been summarized. The focus of this deliverable was on the detailed description of the interaction of the 6G DT with the physical network (data collection via southbound interface) and the interaction with the external systems via other DTs (northbound interface). The architectural options of those interfaces were presented and a detailed description of the data flow in case of interaction of OT DT with 6G DT for the combined performance optimization has been presented.

Following the work in the previous deliverable [DET24-D33] where it was explained why there is a need for situational awareness and interaction between the OT domain, i.e., industry domain and network domain, this work focused on how this interaction could exist. The obvious maturity gap in the advancements of DT technologies between the two domains shows that OT domain could lead the adoption of DT technologies in network and future 6G systems. Through the research of state-of-the-art solutions on network DT, we concluded that most of the focus is put on the architecture aspects, while some crucial parts such as data provisioning are lagging. Therefore, the applicability of well-established data analytics framework with focus on Network Data Analytics Function (NWDAF) and Data Collection and Coordination Function (DCCF) for data collection from 6G network to run the 6G DT was explored. Our approach is that the 6G DT, if seen as a trusted Application Function (AF), can access 6G system data via DCCF and send recommendations or control feedback to 6G Network Functions (NF). However, in the current state not all required data can be obtained through the standardized event exposure interfaces of NFs, but rather manual configuration is required in the Operations, Administration and Maintenance (OAM) system in the management plane. Manual data provisioning is also required for preconfigured static data related to configuration of Radio Access Network (RAN), 6G Core, User Equipment (UE), and network elements. Nevertheless, the data analytics framework is constantly evolving and can become a centralized or distributed data collection solution for 6G DT AF.

References

[3GPP18-23434]	3GPP TS 23.434," Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows", technical specification, Dec 2018
[3GPP23-22261]	3GPP TS 22.261, Service Requirements for the 5G System, v19.4.0, 2023.
[3GPP23-23558]	3GPP TS 23.558," Architecture for enabling Edge Applications", technical specification, March 2023
[3GPP24-23288]	3GPP TS 23.288, Architecture enhancements for 5G System (5GS) to support network data analytics services, v18.7.0, October 2024
[3GPP24-23502]	3GPP TS 23.502, Procedures for the 5G System (5GS), v18.7.0, October 2024
[3GPP24-26531]	3GPP TS 26.531, Data Collection and Reporting; General Description and Architecture, v18.2.0, July 2024
[3GPP24-28537]	3GPP TS 28.537, Management and orchestration; Management capabilities, v18.2.0, July 2024
[3GPP24-29574]	3GPP TS 29.574, Data Collection Coordination Services; Stage 3, v18.7.0, September 2024
[3GPP25-23501]	3GPP TS 23.501, 5G, System Architecture for the 5G System, v18.8.0, 2025.
[3GPP25-28552]	3GPP TS 28.552, Management and orchestration; 5G performance measurements, v18.9.0, January 2025
[3GPP25-29520]	3GPP TS 29.520, Network Data Analytics Services; Stage 3, v18.8.0, January 2025
[5GA21]	5G-ACIA, "Using Digital Twins to Integrate 5G into Production Networks," white paper, February 2021, https://5g-acia.org/whitepapers/using-digital-twins-to-integrate-5g-into-production-networks/
[DET23-D11]	DETERMINISTIC6G, Deliverable 1.1, "DETERMINISTIC6G use cases and architecture principles", Jun. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D21]	DETERMINISTIC6G, Deliverable 1.3, "DETERMINISTIC6G first report on 6G centric enablers", Dec. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET23-D31]	DETERMINISTIC6G, Deliverable 3.1," Report on 6G convergence enablers towards deterministic communication standards", Dec. 2023, https://deterministic6g.eu/index.php/library-m/deliverables
[DET24-D12]	DETERMINISTIC6G, Deliverable 1.2," First report on DETERMINISTIC6G architecture", January 2025, https://deterministic6g.eu/index.php/library-m/deliverables
[DET24-D33]	DETERMINISTIC6G, Deliverable 3.3," Report on Deterministic edge computing and situational awareness via digital twinning security solution", Jun. 2024, https://deterministic6g.eu/index.php/library-m/deliverables
[DET25-D13]	DETERMINISTIC6G, Deliverable 1.3," Report on Dependable Service Design", April 2024, https://deterministic6g.eu/index.php/library-m/deliverables

[DET25-D35]	DETERMINISTIC6G, Deliverable 3.5," Report on Multi-domain End-to-End Schedules".
[ETSI19-MEC003]	ETSI ISG MEC ETSI GS MEC 003 V2.1.1 (2019-01), "Multi-access Edge Computing (MEC); Framework and Reference Architecture"
[ETSI24-015]	ETSI GR ZSM 015 V1.1.1 (2024-02) Zero-touch network and Service Management (ZSM); Network Digital Twin
[Gri14]	M. Grieves, Digital twin: manufacturing excellence through virtual factory replication, white paper, 2014
[IDTA]	https://industrialdigitaltwin.org/
[IDTA-01001-3-0-1]	IDTA Specification of the Asset Administration Shell, Part 1: Metamodel, June 2024
[IDTA-01002-3-0-3]	IDTA Specification of the Asset Administration Shell, Part 2: Application Programming Interfaces, October 2024
[IDTA-01003-a-3-0-2]	IDTA Specification of the Asset Administration Shell, Part 3a: Data Specification – IEC 61360
[IDTA-02022-1-0]	IDTA Specification, Submodel Template of the Asset Administration Shell, Wireless Communication, May, 2024
[IETF10-RFC5810]	IETF RFC 5810: Forwarding and Control Element Separation (ForCES) Protocol Specification https://datatracker.ietf.org/doc/rfc5810/
[IEEE8021-1Q]	IEEE standard for local and metropolitan area networks–bridges and bridged networks, IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018), pp. 1–2163, 2022.
[IEEE09-802.1Qay]	IEEE Standard for Local and metropolitan area networks-- Virtual Bridged Local Area Networks Amendment 10: Provider Backbone Bridge Traffic Engineering https://standards.ieee.org/ieee/802.1Qay/4191/
[IEEE15-802.1Qbv]	IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic https://standards.ieee.org/ieee/802.1Qbv/6068/
[IEEE17-802.1Qci]	IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 28: Per-Stream Filtering and Policing https://standards.ieee.org/ieee/802.1Qci/6159/
[IEEE18-802.1Qcc]	IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks -- Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements https://standards.ieee.org/ieee/802.1Qcc/5784/
[IEEE24-802.1Qdj]	IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks Amendment 38: Configuration Enhancements for Time-Sensitive Networking https://standards.ieee.org/ieee/802.1Qdj/7669/
[IIC20]	Industrial Internet Consortium and Plattform Industrie 4.0, "The digital twin and Asset Administration Shell Concepts and Application in the Industrial Internet and Industrie 4.0." Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.pdf

[IPROUTE2]	S. Hemminger and D. Ahern, “iproute2.” [Online]. Available: https://git.kernel.org/pub/scm/network/iproute2/iproute2.git, 2024. Accessed:06 November 2024.
[ITU-TY3090]	Digital twin network – Requirements and architecture, February 2022
[KHP+22]	N. P. Kuruvatti, M. A. Habibi, S. Partani, B. Han, A. Fellan, H. D. Schotten, Empowering 6G Communication Systems with Digital Twin Technology: A Comprehensive Survey, IEEE Access, September 2022.
[LKD+23]	X. Lin, L. Kundu, C. Dick, E. Obiodu, T. Mostak, 6G Digital Twin Networks: From Theory to Practice, IEEE, July 2023
[Mal21]	Somayeh Malakuti, “The digital twin: from hype to reality,” ABB Review, March 2021, https://new.abb.com/news/detail/80770/the-digital-twin-from-hype-to-reality
[ONF15-TS025]	Open Networking Foundation TS-025: OpenFlow Switch Specification https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
[PI4022]	Platform Industrie 4.0 (in collaboration with Industrial Digital Twin Association IDTA), “Asset Administration Shell Reading Guide,” January 2022, https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/AAS-ReadingGuide202201.html
[SPS+23]	G.P.Sharma, D. Patel, J. Sachs, et. al., “Toward Deterministic Communications in 6G Networks: State of the Art, Open Challenges and the Way Forward”
[TZL+19]	F. Tao, H. Zhang, A. Liu, A. Y. C. Nee, Digital Twin in Industry: State-of-the-Art, IEEE Transactions on Industrial Informatics, vol. 15, no. 4, April 2019
[VGJ+22]	C. Von Arnim, G. Gessner, M. Jarwitz, A. Lechler, and O. Riedel, “Updating the linux taprio scheduler in deterministic time,” in 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–7, 2022.
[VSP+24]	R. Verdecchia, L. Scommegna, B. Picano, M. Becattini, E. Vicario, Network Digital Twins: A Systematic Review, IEEE Access, October 2024

List of abbreviations

5GC	5G Core
5GS	5G System
5QI	5G Quality of Service Indicator
6GC	6G Core
6GS	6G System
AAS	Asset Administrative Shell
ADRF	Analytics Data Repository Function
AF	Application Function
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AMF	Access and Mobility Function
AO	Application Owner
AP	Application Provider
AS	Application Server
BSS	Business Support System
CNC	Centralized Network Controller
CNF	Cloud-Native Function
CP	Control Plane
CPS	Cyber Physical System
CQI	Channel Quality Indicator
CUC	Centralized User Controller
DCCF	Data Collection and Coordination Function
DetNet	Deterministic Networking
DT	Digital Twin
DTN	Digital Twin Network
EAS	Edge Application Server
eBPF	extended Berkeley Packet Filter
ECN	Edge Computing Node
EDF	Earliest Deadline First
EP	Edge Provider
EST	Earliest Start Time
FCAPS	Fault, Configuration, Accounting, Performance, Security
ForCES	Forwarding and Control Element Separation
GCE	Each Gate Control Entry
GCL	Gate Control List
GPP	General Purpose Processor
IDTA	Industrial Digital Twin Association

INT	In-band Network Telemetry
KPI	Key Performance Indicator
kTN	kernel Transport Network
LMF	Location Management Function
MCS	Modulation and Coding Scheme
MEC	Multi-access Edge Computing
ML	Machine Learning
MNO	Mobile Network Operator
MnS	Management Service
MO	Mobile Operator
MQTT	Message Queuing Telemetry Transport
NE	Network Element
NEF	Network Exposure Function
NETCONF	Network Configuration
NF	Network Function
NIC	Network Interface Controller
NMS	Network Management System
NRF	Network Repository Function
NRM	Network Resource Manager
NSACF	Network Slice Admission Control Function
NWDAF	Network Data Analytics Function
OAM	Operations, Administration and Maintenenance
ODA	Open Digital Architecture
ODP	Open Distributed Processing
OPC-UA	Open Platform Communications Unified Architecture
OS	Operating System
OSS	Operational Support Systems
OT	Operation Technology
PCF	Policy Control Function
PCP	Priority Code Point
PDU	Protocol Data Unit
PHM	Prognostic and Health Management
PLC	Programmable Logic Controller
PRB	Physical Resource Block
PTP	Precision Time Protocol
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
SA	Situational Awareness

SDN	Software Defined Networking
SEAL	Service Enabler Architecture Layer
SINR	Signal to Interference Noise Ratio
SMF	Session Management Function
SNMP	Simple Network Management Protocol
SPoF	Single Point of Failure
TAPRIO	Time-Aware Priority Shaper
TAS	Time-Aware Shaper
TSA	Time-Sensitive Application
TSN	Time-Sensitive Networking
UE	User Equipment
UDM	Unified Data Management
UPF	User Plane Function
YANG	Yet ANother lanGuage
VNF	Virtual Network Function
VV&A	Verification, Validation and Accreditation
WSM	Wireless System Manager