szgezu_296_24

Smart Network Solutions for Industry 4.0: Leveraging 5G Network Exposure and Adaptive Industrial Applications

DSc theses

Géza Szabó, Ph.D.

Research Fellow

Budapest, Hungary

2024

szgezu_296_24

Contents

Chapter 1

Introduction

Current Operational Technology (OT) in discrete manufacturing is focused on mass production with few options for product customization. The majority of wired networks used in factories are made up of a proprietary physical layer and other communication protocol technologies. These networks need time and money to build and maintain. Dealing with wires during production cell reconfiguration is a labor- and time-intensive task. This task can be made much easier with wireless technologies, which can also greatly speed up reconfiguration and open up new use cases. No wireless technology has yet been able to provide reliability and low latency that are on par with wired connections. With the arrival of Fifth generation of mobile communications (5G), this is changing, and operational technology companies are preparing to use 5G cellular networks to accomplish their Industry 4.0 vision [1].

Similar to how they do it now with wired network infrastructure, enterprises must connect the private 5G network, also known as the 5G Non-Public Network (NPN), with their present OT or Information Technology (IT) systems, Industrial Internet of Things (IIoT) platforms, and control systems. The entire 5G system is intended to function as an essential component of the existing wired OT or IT communication infrastructure [2].

The IIoT exposure interface that 5G networks provide must be significantly easier to use than any of the current processes that rely on manual methods involving Customer Service Request (CSR) sent to the Communication Service Provider (CSP) or those that rely on CSP-oriented exposure interfaces that assumes in-depth familiarity with the internals of cellular systems. The interface must provide the proper level of abstraction so that factory operators may carry out their routine operational duties without the need for specialized assistance from the service and network provider [3].

This dissertation focuses on integrating the Network Resource Management (NRM) capabilities of network exposure into industrial deployments of private radio networks. The

objective is to enable a seamless transition from wired to wireless communication for Internet of Things (IoT) devices within agile production cells. In the subsequent two sections, I examine two cases in detail: how different communication layers facilitate the shift from fixed control loop frequency or streaming requirements with static NRM to dynamic control loop frequency or streaming requirements with dynamic NRM. It is also important to note that the network exposure process still requires in-depth network knowledge from the Vertical Application Layer (VAL) developer, as demonstrated in [3]. This work presents complex use cases that introduce dynamic VAL behavior and dynamic NRM initiated from the User Equipment (UE), along with methods to automate requests within the network exposure process.

1.1 Static NRM

Static NRM refers to the fixed allocation of network resources–such as bandwidth, processing power, and storage–without adapting to changes in traffic demand or network conditions over time. In this approach, the allocation is determined based on pre-defined assumptions about the network's expected load, and these resources remain dedicated to specific applications or services regardless of actual traffic fluctuations.

This concept ties into the practice of over-provisioning, where application developers request more network resources than may be necessary to ensure that application traffic can be handled even under conditions of high network load. Over-provisioning simplifies the development process by avoiding the need to precisely calculate the minimal resources required, thus mitigating the risk of performance degradation during peak traffic periods. While this approach offers reliability and eases the burden on developers, it can lead to inefficient resource usage, as underutilized network capacity remains reserved even when it is not needed.

1.1.1 The design of the applications

Static NRM approach resembles a control loop in application design, where the application's resource usage operates at a fixed frequency. The application regularly consumes a set amount of network resources at a constant rate, regardless of the actual workload or traffic conditions. Similar to how a control loop functions with a steady frequency, the application operates under the assumption that the allocated resources will always be sufficient to meet its needs, even during fluctuations in traffic.

1.1.2 The design of the middlewares

In this work, I examine two industrial protocols: Robot Operating System (ROS) 2 and Open Platform Communications (OPC) Unified Architecture (UA) (see Section 1.3 for further details). The subsequent discussion focuses on these protocols. Static NRM is well-supported by ROS 2, OPC UA applications with the respective Quality of Service (QoS) settings.

ROS2. As ROS1 was not designed with QoS in mind, the QoS policies of ROS2 were a huge step toward feasible setup of more complex network settings. As the ROS2 design article summarizes, there was no common method of overriding QoS settings while constructing a node prior to ROS2 Foxy [4]. Reusing nodes that other people have created is popular in the ROS2 ecosystem, and in many use cases allowing QoS profiles to be altered makes sense.

OPC UA. [5] states that both the OPC UA publishers and subscribers can set their required `QoSCategory` and `PrirotiyLabel`. If the pairing of these values is not found in the mapping table [6], the communication cannot be established. Standard values for `QosCategory` with the required structures in the `DatagramQos` array are pre-defined. This list can be extended by specifications built on top of OPC UA `PubSub`.

1.1.3 The design of the transport protocol

Data Distribution Service (DDS). DDS provides fine-grained control over the QoS setting for each of the entities involved in the system. Common entities whose behavior can be modified via QoS settings include: Topic, DataReader, DataWriter, Publisher and Subscriber. QoS is enforced based on a Request vs Offerer Model, however Publications and Subscriptions will only match if the QoS settings are compatible.

Given the complexity of choosing the correct QoS settings for a given scenario, it may make sense for ROS2 to provide a set of predefined QoS profiles for common use-cases (e.g., sensor data, real time, etc.), while at the same time give the flexibility to control specific features of the QoS policies for the most common entities. ROS2 uses several QoS policies provided by FastDDS [7] to manage communication between nodes.

Message Queuing Telemetry Transport (MQTT). [8] illustrates how an application implements QoS mapping from OPC UA to MQTT. This involves a direct transformation, which maps the 4-level OPC UA QoS to the 3-level MQTT QoS while omitting the best-effort QoS class in OPC UA. This mapping alters the role of OPC UA QoS settings. Since this

process is typically performed during session initialization, the mapped MQTT QoS is expected to remain unchanged throughout the topic's lifespan.

1.1.4 The design of 5G exposure

Static NRM type of operation that the 5G network exposure in Third Generation Partnership Project (3GPP) Rel. 17 is designed for either with the Network Exposure Function (NEF) in Technical Specifications (TS) 23.501 [9], TS 23.502 [10] or Service Enabler Architecture Layer (SEAL) in TS 23.434 [11] (see Section 1.3 for further details). During session initiation, the necessary network resources are allocated and are valid for the whole lifecycle of the application.

3GPP TS 29.549 [12] Rel. 17 defines 3 types of operation for unicast resources, that is creating a new session with POST, providing information of an existing resource with GET or stopping an existing resource with DELETE.

1.1.5 The design of 5G core

A common method to implement a QoS for a certain flow for the NRM request in 5G is to establish a dedicated bearer with the requested QoS Class Identifiers (QCI) between the UE and the base station then make a routing rule to route a certain flow through the established bearer.

1.2 Dynamic NRM

Dynamic NRM refers to the adaptive allocation of network resources that responds in real-time to changing traffic conditions, application demands, and network load. In contrast to static NRM, dynamic NRM continuously adjusts the allocation based on current requirements, optimizing the efficiency and reliability of resource utilization while ensuring that applications receive the necessary resources as conditions fluctuate.

This concept is particularly relevant when comparing Mobile Broadband (MBB) applications and industrial applications. MBB applications are often designed to dynamically adapt to network conditions. For example, a File Transfer Protocol (FTP) can adjust its speed to utilize the available bandwidth, and video streaming applications can modify the resolution to suit changing network capacity. These applications are flexible in their resource consumption and can continue operating even as network conditions vary.

Industrial applications, on the other hand, have strict network requirements that must be consistently met with high reliability. These applications often operate in environments

where real-time data transmission and precise control are critical to maintaining safety and performance. If the network cannot guarantee the required level of service, the operation may fail, posing potential risks, especially from a safety perspective. Unlike mobile broadband, industrial applications cannot adapt to fluctuating resources without compromising their functionality or safety.

1.2.1 The design of the applications

In my previous work [13, 14, 15], I initiated efforts to integrate network exposure and NRM into industrial setups. In these setups the dynamic nature of the operation appeared, however the NRM is controlled centrally by the network by knowing every necessary information of the application process. In this work my focus is on the UE initiated NRM that is requested by added internal logic to the IoT devices to enhance the radio efficiency of the uplink-heavy sensor traffic.

1.2.2 The design of the middlewares

ROS2. In ROS2 there are major issues with QoS reconfiguration during run-time. The main one is that only when the publisher and the subscriber QoS profiles are compatible can a link between a publisher and a subscription be established. A "Request vs. Offered" model is used to determine whether two QoS profiles are compatible. Publishers supply a QoS profile that is the "maximum quality" they can provide, while subscriptions want a QoS profile that is the "minimum quality" they are prepared to accept. Connections are only established if no requested QoS profile policy is stricter than any offered QoS profile policy. Publishers and subscribers need to have compatible QoS settings otherwise they either cannot publish or cannot subscribe. Overriding QoS settings run-time can end up losing the connection. In ROS2 Galactic, developers agreed on a way to support QoS overriding which is done via the `QoSOverridingOptions`[16] class that can be setup during initialization, run-time and have a callback function if it is running. There are a few well-known examples in the ROS2 community using QoS overriding. The most common one is ros2bag [17], but there can be more upcoming as the latest ROS2 releases become more widespread in the community.

The other requirement of a feasible dynamic NRM use case on ROS2 is the possibility of creating executors [18] with varying wake up frequency. ROS2 is designed to initialize executors with given frequency through the `rclcpp::Rate` object, but it is certainly possible to overcome the limitations by e.g., specifying multiple rate objects for the same executor and switching between them.

OPC UA. OPC UA specification does not provide a direct mechanism to dynamically change the QoSCategory [5] and PriorityLevel [6] during runtime. These settings are typically configured during the initialization phase of the communication and remain static throughout the lifetime of the communication session. If the application developer needs to change the QoS settings during runtime, the developer would typically need to establish a new communication session with the desired QoS settings. This would involve creating a new OPC UA session with the updated QoSCategory and PriorityLevel values. While the specification and Application Programming Interfaces (APIs) do not offer direct support for this feature, application developers with low-level access to the PubSub code can still configure QoS on a per-packet level if necessary within the MQTT layer when publishing MQTT messages.

1.2.3 The design for the transport protocol

DDS. QoS policies can be grouped into two categories: Static (Immutable) Policies and Dynamic (Mutable) Policies [19]. The static policies directly affect the internal structure of how the data is exchanged, while the dynamic ones deal with behaviors that are more about performance tuning or operational management. The static QoS policies are the following: DestinationOrderQosPolicy, DurabilityQosPolicy, DurabilityServiceQosPolicy, HistoryQosPolicy, LivelinessQosPolicy, OwnershipQosPolicy, PresentationQosPolicy, ReliabilityQosPolicy, ResourceLimitsQosPolicy. Once data writers/readers are created, altering these could break the ongoing communication model, so they must be fixed at initialization.

The dynamic QoS policies are the following: DeadlineQosPolicy, EntityFactoryQosPolicy, GroupDataQosPolicy, LatencyBudgetQosPolicy, LifespanQosPolicy, OwnershipStrengthQosPolicy, PartitionQosPolicy, ReaderDataLifecycleQosPolicy, TimeBasedFilterQosPolicy, TopicDataQosPolicy, TransportPriorityQosPolicy, UserDataQosPolicy, WriterDataLifecycleQosPolicy. These policies are more flexible and are generally associated with the behavior of the entities during operation, which can be adjusted without breaking the underlying communication setup.

MQTT. The MQTT's QoS level can be dynamically set on the fly during the publisher or subscriber's lifetime. MQTT allows publishers and subscribers to specify the desired QoS level on a per-message basis, providing flexibility and adaptability in message delivery guarantees. This means that publishers can choose different QoS levels for different messages based on the importance or reliability requirements of the data being sent.

The MQTT publisher inherits the behavior of the OPC UA publisher. The same applies for MQTT that I discussed for OPC UA. The MQTT broker has the capability to alter the

traffic characteristics, though it has not got a well-defined queuing mechanism according to the standard [20]. When a message is published to a topic, the broker will immediately attempt to deliver the message to all subscribers that have subscribed to that topic based on the respective QoS levels. Some MQTT brokers may offer optional features to support buffering and message persistence for offline clients. These features can cause the alternation of the traffic pattern, but they are typically broker-specific and not part of the MQTT standard.

1.2.4 The design of the network exposure

Though the 5G network has the capability to support a form of low granularity dynamic NRM, the design principle dit not contain the support for dynamic NRM from the exposure APIs. This is evident from the Representational State Transfer (REST) APIs, as neither SEAL NRM [12], nor CAMARA QoD [21] provides an UPDATE-type REST API for unicast subscriptions or QoS sessions, respectively. Nevertheless, the current design conventions for OPC UA, 3GPP SEAL and CAMARA QoD tend to configure NRM during connection initialization and maintain it unchanged throughout the connection's lifespan.

1.2.5 The 5G network

Dynamic NRM update is performed by updating the QCI, which triggers a new bearer establishment and a new routing rule to route the flow through the newly established bearer. Establishing a new bearer takes time in a magnitude of a few seconds. Rerouting happens within a second. Updating a bearer means a tear down and a new bearer establishment. Though it is important to note that some default bearers are always present while connectivity is provided to the UE, connection outage does not happen during the bearer switching.

This means that NRM updates can be dynamic in a few seconds granularity. However, as current design patterns for both ROS2 and 3GPP SEAL are formed to make the NRM at the connection initialization and do not modify it during the lifetime of the connection, this is a low granularity NRM (even without in-band signaling).

1.3 Background

In this section I collected the relevant background information on several main topics. These topics are discussed in the following subsections.

1.3.1 Network Resource Management

NRM is vital for efficient network operation, optimal resource allocation, and improved quality of service. It adapts to evolving user needs. In traditional practices, network operators managed NRM. But in programmable networks, developers can customize NRM for specific use cases, meeting their application needs precisely.

1.3.2 Network exposure

The 3GPP [22] Service-Based Architecture (SBA) for 5G Core (5GC) specifies a functional architecture and standardized interfaces as 5GC control plane Network Functions (NFs) expose Service-Based Interfaces (SBIs). The NFs register their services in the NEF [23] and services can then be discovered by other NFs. This enables flexible deployment, where every NF allows the other authorized NFs to access the services, which provides possibilities for external third parties to use the services and capabilities provided by 5GC. It is expected that 5GC will be deployed on software-defined infrastructure, however, the choice of implementation rests with the operator or vendor.

The traditional way of configuring cellular networks is via their Operations and Maintenance (OM) interfaces which are command-line tools or based on NetConf [24]. These interfaces can be used by network operators only, hence the factory needs to issue a CSR to the CSP for each change. Handling of CSRs typically involves several manual steps and hence it is a time-consuming procedure. Some simpler configuration steps can be automated by the factory using the NEF APIs. These are specified by the 3GPP in TS 23.501 [9], 23.502 [10] and 29.522 [23]. While the NEF was a novel feature of 3GPP Release 15, it has an inherent issue due to the offline configuration of the Northbound APIs: API discovery was not supported and the APIs behavior led to fragmentation. Soon after the introduction of NEF, the Common API Framework (CAPIF) was defined in to establish a single and harmonized platform for all 3GPP Northbound APIs. The CAPIF related work happened in the 3GPP Technical Specification Group (TSG) System Aspect (SA) Working Group (WG) 6 (SA6) which is the application enablement and critical communication applications group for vertical markets. The main objective of SA6 is to provide application layer architecture specifications for 3GPP verticals, including architecture requirements, functional architecture, procedures, information flows, interworking with non-3GPP application layer solutions, and deployment models as appropriate. While NEF provides a low-level programmability of the network, there was room to improve its user or application developer-friendliness and the common set of capabilities for the 5G verticals was identified and the SEAL 3GPP TS 23.434

[11] is introduced.

The SEAL for Verticals was introduced by 3GPP in Release 16 to enable factories and other verticals to automate system integration and 5G network configuration operations without deep network knowledge. The APIs for provisioning, connection management, device management, connection monitoring, group management, user profile retrieval, identity and key management, location reporting, events, and network resource management are specified in 3GPP TS 23.434 [11]. SEAL exposure interface demonstrated a drastically simplified system integration of industrial 5G devices [3].

1.3.3 Robot Operating System 2

A collection of software libraries and tools called the Robot Operating System are used to create robot applications. ROS contains the open-source resources that are needed for any upcoming robotics project, including drivers, cutting-edge algorithms, and robust development tools. The robotics and ROS communities have undergone significant development since the founding of ROS in 2007. By utilizing what worked well in ROS 1 and enhancing what is not, the ROS 2 [25] project hopes to adjust to these developments.

A wide range of QoS policies are available in ROS 2 and let the developer fine-tune node connectivity. With the correct combination of QoS settings, ROS 2 can have a wide range of potential states, ranging from best-effort User Datagram Protocol (UDP) to Transmission Control Protocol (TCP)-like reliability. ROS 2 selected DDS for transport protocol driven by its maturity, real-time capabilities, scalability, interoperability, and wide adoption in various industries.

1.3.4 OPC UA

OPC UA serves as a versatile and open-source framework, adhering to the IEC62541 set of standards. My topic of interest, the network model is defined in [6], to facilitate seamless data exchange between sensors and cloud applications. Developed by the OPC Foundation, it offers a range of notable features, including standardized data models for industrial equipment, adaptable security profiles encompassing authentication and encryption, support for diverse communication patterns, and protocol independence with mappings to various protocols. OPC UA is designed to use for industrial data exchange, but it has expanded its utilization to encompass building automation, cloud applications, and more. My main interest is the communication modes of operation in OPC UA. My focus is on the PubSub mode. In this mode, data is published by a publisher and subscribed to by one or more subscribers. This allows for efficient distribution of data to multiple recipients, without

requiring each recipient to establish a separate connection to the publisher. I selected this approach to pursue because my objective was to create a proposal that could be embraced across both OPC UA pubsub operation and MQTT, even without the OPC UA layer. This approach scales effectively and eliminates the need for modifications to the existing code base.

1.4 Related work

In this section I collected the relevant related works from several main topics. These topics are discussed in the following subsections.

The analysis of IIoT use cases and the definition of the requirements of the OT industry on 5G network have been done in the 5G Alliance for Connected Industries and Automation (ACIA), with broad participation from the OT and telecommunication industries. The results are documented in a whitepaper [26], which describes the detailed requirements related to device management use cases, network management use cases, and security aspects. The whitepaper serves as an input to standardization efforts in the 3GPP. The ongoing specification work in 3GPP is targeting a generic approach so that other industrial verticals can also use the exposed capabilities. These include automotive, rail-bound mass transit, electric power distribution, central power generation, health care, and smart cities.

One of the first documented prototype is built and discussed in [1] to verify the exposure capabilities of a real private 5G network via a cloud-based digital automation ecosystem. Authors investigate how a commercially available 5G non-public network can be extended with exposure capabilities to automate its operational configuration and customization from an industrial automation system. They demonstrated in practice an interoperable and easy-to-use environment for managing and monitoring 5G connectivity of networked industrial devices.

An examination of a Cyber Physical Production System (CPPS) for a simulated robot with simulated network effects was published by the authors of [27]. The concept of a wireless resource allocation approach that is Quality of Control (QoC)-aware (QoCa) was presented. The method is based on classifying the robotic arm's movement phases into those that require high and low QoC. The arm movements that do not require high precision are those requiring low QoC. For instance, getting to a position where a robotic arm needs to do a task should not require extreme precision. Contrarily, arm movements necessitating high QoC are those where precise joint movements are necessary in order to successfully complete the necessary tasks. The proposed approach was compared against the locally controlled-based

solution from the Agile Robotics for Industrial Automation Competition (ARIAC). The analysis shows that, on average, 54% of the radio time can be saved without reducing the robot cell's productivity.

1.5 Organization

The organization of the dissertation is structured as follows. Chapter 2 introduces a method for determining the appropriate timing for making NRM requests by sensors enhanced with Artificial Intelligence (AI)-agents. Chapter 3 presents a methodology for automating network exposure requests, focusing on parameterization of 3GPP SEAL NRM with collected data from the ROS2 framework. Chapter 4 extends the work in ROS2 to the OPC UA domain, proposing a comprehensive solution for cross-domain service discovery. Chapter 5 outlines a feedback mechanism designed to provide observability capabilities for specific industrial processes. This mechanism supports either the network or other VAL applications and is elaborated in detail for three distinct industrial processes. The chapters of this dissertation are based on the author's previous publications [28, 29, 30, 31, 32], which are referenced in the bibliography.

Chapter 2

AI-Powered Sensor-Driven Timing Strategies for Optimized NRM

2.1 Introduction

A common solution in current factory environments is to deploy sensors that stream their status constantly. It is the case for e.g., temperature sensors, light beam sensors which are connected to a Programmable Logic Controller (PLC) Input/Output (IO) device and stream their status in 10 ms granularity [33] and also it is the case with sensors having more data like cameras or Laser Imaging, Detection, And Rangings (LIDARs) devices. The sensor data in such frequency, particularly high-resolution images or videos can cause significant load on communication networks especially in the case of a wireless network [34]. In this chapter my goal is to relax the radio load in a way that is easy to deploy and maintain and scales with the number of IoT devices in the system. My goal is to propose a system with full radio coverage of the production cell, achieving always-on connectivity for those cameras that are important for the operation, cost optimized in terms of energy and spectral efficiency and performs the same productivity level Key Performance Indicators (KPIs) as a system using cables to communicate.

Sensing efficiency as known as Quality of Sensing (QoSensing) [35] can be evaluated by considering the following performance metrics: coverage rate, connectivity ratio, energy consumption, end-to-end delay, and throughput. To achieve this, I considered the following stepping stones.

- I take an operational agile production cell using sensors with wired communication.
- I replace the wires with wireless communications and aim for the minimal number of

access points (radio dots) that keep the sensors operational in the production cell by checking the radio connectivity during the operation of the production cell.

- I apply Reinforcement Learning (RL) [36] to optimize the layout and minimizing the number of the required dots in the production cell.
- I introduce a novel AI agent that enables cooperative interaction between the cameras and radio dots to ensure efficient utilization of camera streams and optimal radio coverage. This agent proactively selects transmission quality and timing, while also enhancing sensor capabilities by incorporating AI functionalities and addressing specific sensing aspects.

Note that support of NRM capabilities of the UE, which is attached to the cameras is a novel feature standardized recently [11]. So far, UEs had no capabilities to influence the operator's network by any means. I built upon this novel feature and analyzed how automatic this solution can be made by deploying the capable AI agents onto the UEs.

The operation of the production cell can result in temporal connection outages due to the moving actuators which may block the connection between a transmitter and a receiver. As my goal is to consider the dynamics of the production cell during radio coverage design and to ensure full functionality of the cameras, my approach is unique in terms of running the optimization algorithms during the operation of the production cell. This requires long training times as training collects samples during the entire operation of the production cell. To enhance the process, I introduced several improvements:

- I do the optimization in a simulation environment that has both detailed rigid-body physics and radio propagation simulation capabilities.
- The radio propagation modelling utilizes the novel fast raytracing capabilities of recent Graphics Processing Units (GPUs) to speed-up the radio propagation simulation part.
- I introduce a novel task-specific curriculum-learning based solution that prepares the RL policy in a static and later in the dynamic environment.
- I introduce a novel physical twin of the radio connected camera with self-adjusting scene setup capabilities within the production cell, enabling us to establish an interconnected environment that optimizes the efficiency and accuracy of the trained AI agents.

In summary, NRM is a novel feature of programmable 5G systems that capable devices can support from Release 17. My extension of novel cooperative AI agents of the cameras

and dots can utilize the 5G radio efficiently. Both these agents are trained on a novel self-configuring physical twin with a novel curriculum learning based method that uniquely considers the dynamics of the radio propagation of a production cell under operation.

My proposal is unique in that it enhances modeling accuracy through computationally intensive calculations: radio propagation modelling, physical twin, operating production cell. These operations typically slow down AI training, but I also introduce novel solutions to significantly improve training speed without sacrificing any level of detail of the high-resolution modeling.

The demo video of the proposed system in action can be seen at [37].

2.2 Background

In this section I collected the relevant background information on several main topics. These topics are discussed in the following subsections.

2.2.1 Digital, Cyber and Physical Twins

This chapter employs the concepts of digital twin and physical twin. A physical twin represents a tangible real-world object, while a digital twin is a virtual counterpart created using digital models. The authors of [38] introduce the concept of a cyber twin, which behaves like its physical counterpart but primarily exists in the digital realm, using sensor data for calculations and simulations. In this chapter, I primarily refer to physical twins, which can be seen as a specific form of cyber twin with functionality tailored to customer needs.

2.2.2 Edge and fog computing

IoT devices like sensors and actuators are generating vast amounts of real-time data, making them attractive for AI systems. However, implementing machine learning on these devices is challenging. Offloading data to external systems, like cloud servers, is common but affects latency, costs, and privacy. To address this, efforts are made to bring more computing power closer to IoT devices or the network's edge. This enables edge computing devices to handle Machine Learning (ML) tasks, solving the mentioned problems. In a research paper [39], successful applications of intelligent edge systems are discussed, primarily focusing on compression techniques, tools, frameworks, and hardware. These solutions often emphasize large-scale object recognition, but my main interest is in communication-related NRM tasks.

2.2.3 Reinforcement learning

I used ML, particularly RL, to create a solution for a complex environment with many autonomous sensors in a production cell, considering radio conditions for NRM. RL, inspired by trial and error learning, helps an agent make sequential decisions in an environment to maximize rewards. In single-agent RL (SARL), one agent interacts with the environment. Multi-agent RL (MARL) deals with scenarios where multiple agents interact, impacting not only their rewards but also those of others [36, 40].

2.2.4 Radio propagation simulation

In my MARL approach, the reward function accounts for radio propagation effects, which inherently affect all agents' rewards, without explicitly considering other agents' statuses.

Electromagnetic waves, governed by Maxwell's equations, lack general analytical solutions in realistic propagation environments. Propagation modeling estimates signal strengths based on system parameters like frequency, terrain, and antenna heights. This includes models like the Friis equation for open space and empirical models such as the Hata-Okumura model for urban areas [41]. Handling Multiple-Input Multiple-Output (MIMO) and non-free space, like indoor environments with dynamic objects, proves challenging for empirical models.

Ray tracing employs ray optics to solve Maxwell's equations in high-frequency ranges, offering path loss, arrival/departure angles, and time delays [41]. Rays are intuitive from everyday experiences with sunlight, and they can be categorized as direct (Line of Sight (LoS)), reflected, transmitted, diffracted, or scattered. Determining rays from a source to a field point is complex, especially in urban environments. Fast ray-tracing algorithms and accurate field calculations are essential for this research area, dating back to the 1990s. Raytracing employs methods like Fermat's principle, the image method, and the shooting and bouncing ray method, or combinations thereof.

Raytracing is a computation heavy and there are several acceleration methods in literature. There are several algorithmic solutions [41] and there are hardware based solutions like application of GPUs [42] or computation clusters-based solutions [43]. My work applies a hardware based raytrace calculation enhancement solution.

2.3 Related work

In this section I collected the relevant related works from several main topics. These topics are discussed in the following subsections.

2.3.1 Video Surveillance System

The authors of [44] offer a surveillance system that automatically records high-resolution videos of pedestrians as they go through a defined area and consists of wide Field of View (FOV) passive cameras and pan/tilt/zoom (PTZ) active cameras. While they introduce the concept of high- and low-resolution data, they use it for processing optimization and the amount of network data that is utilized during transmission is not considered.

Authors of [45] discuss Wireless Fidelity (WiFi) operated surveillance camera system. The ubiquity of cameras limits the amount of video that can be sent to the cloud, especially on wireless networks where capacity is at a premium. Their solution splits video processing between edge computing nodes co-located with cameras and the cloud to save wireless capacity, which can then be assigned to WiFi hotspots. With new traffic scheduling and video frame prioritization algorithms, the solution optimizes bandwidth use. This work introduces the concept of network load during video stream transmission, but the radio environment is considered static.

In [46], authors provide a survey on event cameras. These bio-inspired sensors are different from traditional frame cameras in that they measure per-pixel brightness changes asynchronously and output a stream of events that contain information about the brightness changes' timing, location, and sign. Compared to regular cameras, event cameras have appealing features such high temporal resolution, high dynamic range, low power consumption, and high pixel bandwidth, which reduces motion blur. Due to their high speed, low latency, and excellent dynamic range, event cameras have a great deal of potential for robotics and computer vision in difficult situations for standard cameras. To fully utilize these sensors, however, new processing techniques are needed to handle their unusual output. My aim is to keep the current streaming mode of operation of brown-field factory deployments and make improvements within these constraints.

According to [47] the size of data offloaded to edge computers is directly proportional to the delay incurred by the offloading. Delay caused by the offloading method is therefore minimized by using data reduction techniques to reduce the offloadable data. Their proposed method downsizes the data and can reduce the delay due to network traffic. My proposed method performs data compression inherently and with no delay.

2.3.2 Radio cell deployment

Authors of [48] review the literature about Radio Access Network (RAN) Base Transceiver Station (BTS) placement, describing classic approaches, and then concluding with automated

placement planning without the supervision of human domain expert. During the planning phase, BTS placement locations (either optimal or sub-optimal) are selected in order to maximize the intended RAN quality metrics. Future automated cell placement will primarily focus on macrocell (high power, long range) BTS placement for small cells (low power, short range cells), as these BTSs will be used on a bigger scale in future (5G and beyond) multi-tier RANs. Although the deployment of small cell BTSs is simple and inexpensive, it is believed that RAN operators will not move macrocells as part of the switch to 5G RAN because doing so would be both difficult and expensive.

2.3.3 Digital Twins

Digital Twins (DTs) are widely used in industry for a variety of applications. As the IoT is gaining ground and becoming increasingly popular in smart city applications such as smart energy, smart buildings, smart factories, smart transportation, smart farming, and smart healthcare, the DT concept is evolving as complementary to its counter physical part.

[49] discusses DTs in the relation of Sixth generation of mobile communications (6G) also identifying several open research challenges and future directions that need to be addressed before the end-to-end deployment of DT technology in 6G communication systems. My work concentrates on the utilization of DT technology in three main areas within the context of 6G communication systems. Firstly, I explore DT technology for constructing platforms that facilitate training of AI models designed for 6G communication systems. Secondly, I investigate how DT technology can simplify and expedite the configuration of site deployment. Lastly, I explore the potential of DT technology in enhancing radio resource management in 6G communication systems.

2.4 Radio coverage and QoSensing optimized transmission of the camera stream in a factory environment

Current AI-enabled cameras usually focus on the recognition or the positioning of the object [50], but I argue that a significant improvement in the operation of current cameras could be that I make them aware of their environment, make them learn the spatial and temporal changes around them and manage their data sending resolution, rate, or timing accordingly.

Figure 2.1 displays my system in action, where AI agents in radio dots and cameras collect and respond to environmental data. In my AI deployment scenario, using a single policy for all dots and cameras might overfit to specific setups. Therefore, a multi-agent setup is preferred, aligning with reasons detailed in [40]. This setup is beneficial for privacy

in scenarios where dots and cameras are from different vendors. Multi-agent systems scale well, suitable for dense deployments in Industry 4.0 settings. In my approach, I employ the same policy for all dots and another for all cameras. The system is designed as heterogeneous communicating agents with scheduled coordination to optimize resource management.

The multi-agent AI system is deployed in a RL setup. The task that I aimed for is to make a static deployment of the dots by the AI. The regular grid deployment is conservative in the sense that radio coverage is provided for the whole cell. My intention is not to alter it completely, just fine-tune it. Providing the AI agent with the task to cover the whole production cell is a significantly bigger state space consuming more training than to optimize a grid-like deployment. My approach is to start from a grid like deployment and make the AI move the dots slightly to improve the connectivity of the whole production cell incorporating the dynamics of the actuators. The AI agents for the cameras aid the dots to ensure radio coverage by proactively selecting the quality and the timing of the transmission.

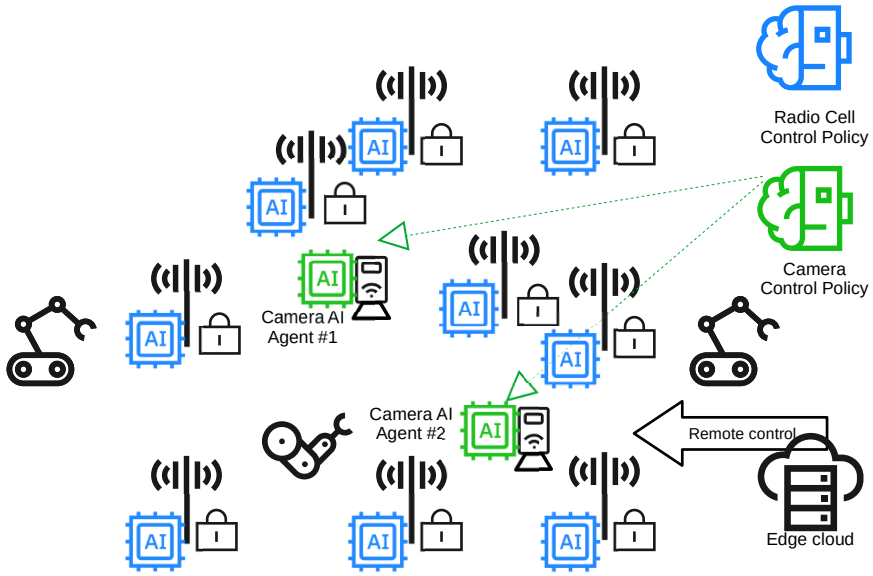


Figure 2.1: Stationary dots and sensors with cooperative AI agents

Figure 2.2 presents the system architecture. This RL-based solution produces optimized policy graphs for radio dots and cameras. The environment is the production cell, operational in various scenarios including simulation, real hardware, hardware in the loop, or DT [13].

2.4.1 Observation spaces

In RL, the observation space refers to the set of all observations that an agent can receive from its environment at each time step. These observations provide information to the agent

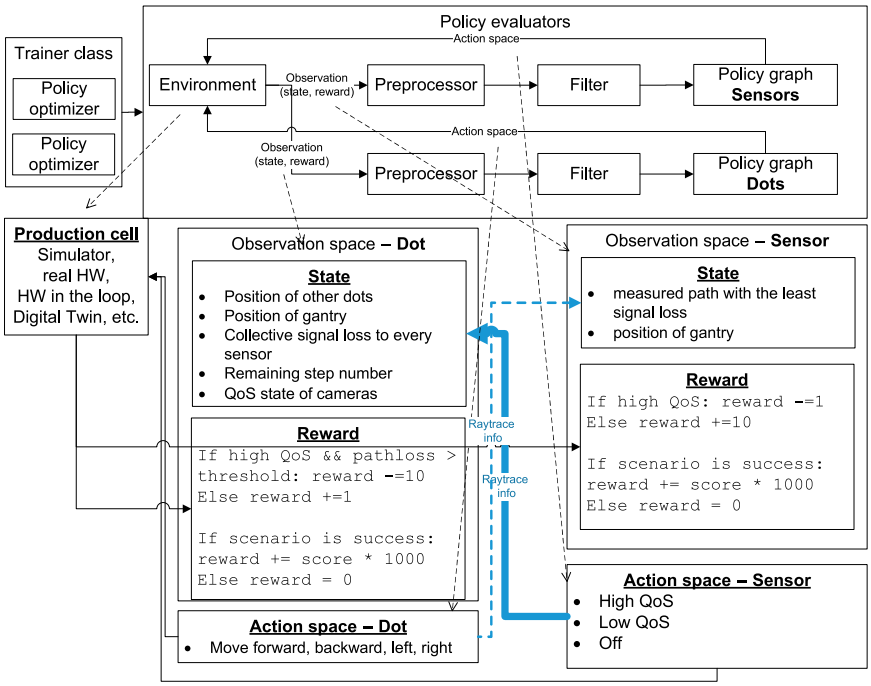


Figure 2.2: The AI architecture of the proposed system

about the state of the environment, and the agent uses this information to make decisions and take actions. In the following sections, I discuss the observation spaces (inputs) of the two agents.

2.4.1.1 Radio dot

In the radio dot's observation space, the *state* includes the positions of other radio dots, overhead gantry, collective signal loss to all sensors, individual signal losses for that dot, remaining steps, and cameras' QoSensing state. Loss is computed via ray tracing and pertains to the entire network, not just individual dots. The *reward* in the observation space enforces learning. It should include elements influenced by actions, with other information in the observation space. Weighting the reward elements affects the learning rate, impacting convergence.

The radio dot's reward function includes scenario scores from [51], which depend on part placement accuracy determined by camera resolution. Camera resolution is tied to QoS. High QoS with high path loss results in penalties for the agents, as maintaining a high-resolution data connection becomes challenging.

2.4.1.2 Camera

In the observation space of the camera, the *state* is the measured path with the least signal loss, and the position of gantry.

Note that the utilized observation spaces do not take into account any information on the specific layout of the production cell and the method can be considered to be universal in this respect.

2.4.2 Action spaces

In RL, the action space represents the set of all actions that an agent can take in a given environment. The action space defines the choices available to the agent at each time step, and the agent's goal is to select actions that maximize its cumulative reward over time. The action spaces of the two agents are discrete action spaces, meaning that the agents can choose from a finite set of distinct actions. In the following sections, I discuss the action spaces (outputs) of the two agents.

2.4.2.1 Radio dot

Optimizing radio dot positions can be challenging due to the large action space when selecting absolute positions on the ceiling grid. This approach can slow down exploration and result in noisy decisions. To improve this, I propose a more efficient strategy. The AI agent considers relative positions or movement directions, with four possible actions: forward, backward, left, and right on a planar structure. Adding the capability to modify dot height aligns with classical RL, where the agent makes step-by-step decisions based on each observation. This simplifies the learning process and reduces decision complexity, making observations more Markovian.

2.4.2.2 Camera

The action space of the camera consists of the selection of connectivity QoS: high or low QoS connectivity. The QoS has a direct effect on the camera resolution (QoSensing) and later the positioning accuracy for the pick and place scenario. During the training process a policy is taught to make data transmission in time, in order to minimize radio load considering the operation of the production cell and in-cooperation with the radio dots.

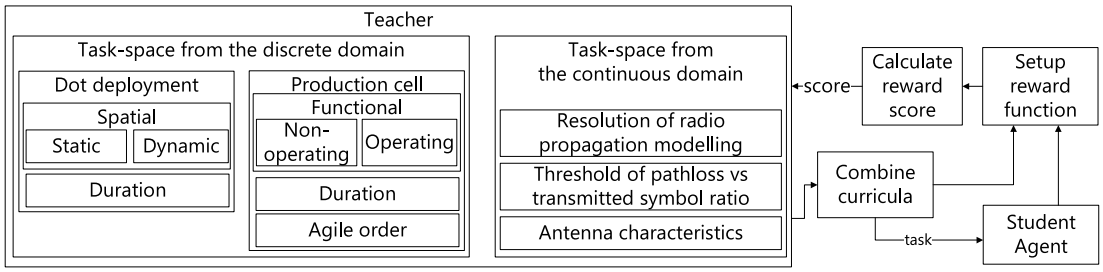


Figure 2.3: Architecture to setup both the task and reward curricula

2.4.3 Considering radio connectivity

The action space of one agent is connected to the observation space to another agent when communication between the two agents is required. In practice, the decision of the sensor agent is the following: if it requires to transmit high- or low-resolution data, it seeks for high- or low-QoS connectivity. In case of a connection between the two agents, the radio propagation modelling provides a valid path between the two. If the pathloss is smaller than a given threshold, I assume that communication is possible, and it provides enough symbol rate for high- or low- resolution sensor data (see Figure 2.6b).

2.5 Curriculum learning for radio coverage deployment in a Digital Twin

Providing the AI agent with the task of covering the whole production cell is a significantly bigger state space consuming more training than optimizing a grid-like deployment. I propose a curriculum learning based solution to speed up a reinforcement learning based AI agent that's main function is to make optimal radio dot deployment in a production cell. The AI agent receives increasingly complex tasks defined on static and dynamic radio dot deployments, operating, non-operating production cell and the radio propagation model resolution.

Curriculum learning involves setting the environment to different difficulty levels or tasks, enabling a controlled progression of learning from easier to more challenging stages. The task space refers to the set of tasks or learning objectives that a machine learning model needs to learn. Figure 2.3 shows the workflow of the curriculum learning for the radio coverage deployment. The different curricula can enable or change different elements of the RL environment. It can enable or disable agents or change the reward function of the agents.

2.5.1 Curricula for the task space

I establish two operation modes for radio dots and the production cell: static and dynamic. In the static dot scenario, dots remain fixed without movement or the ability to seek improved coverage. The static cell scenario means the production cell remains non-operational. In the dynamic dot scenario, radio dots can move freely to optimize coverage but remain static during the inference phase. In the dynamic production cell scenario, the production cell is in operation with robots actively controlled and in motion. Based on these four options, I introduce four curricula in the subsequent paragraphs.

2.5.1.1 Task-space from the discrete domain

Static radio dot deployment in a static production cell. This is the state-of-the-art. It does not require RL. Other methods like evolutionary algorithms can do this.

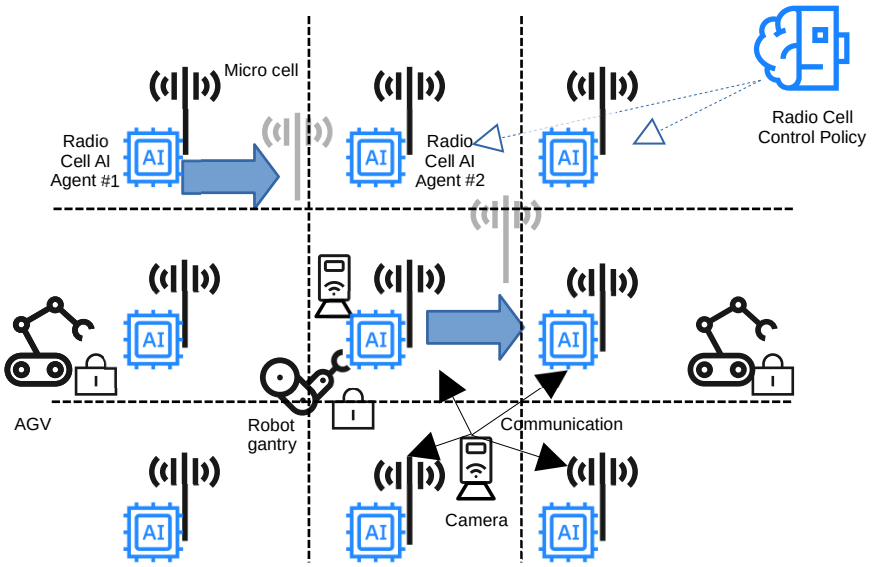


Figure 2.4: Dynamic radio dot deployment in a static production cell

Dynamic radio dot deployment in a non-operating (static) production cell. In this *curricula* I apply the radio propagation modelling deployed in the production cell, in which all the elements are static in an initial position, there is no control on the actuators (see Figure 2.4). Application of a curriculum in a static robot cell relaxes two issues: 1) the accurate rigid body simulation of the production cell used in the DT is processing heavy, 2) the actuators make temporal and spatial changes in the radio coverage which makes it more difficult for the agent to learn. The *action space* of the AI is to move the radio

dots in the surface above the production cell in four possible directions. The *observation space* contains the simulation step counter. The *reward* is designed to encourage radio dot re-positioning during the static production cell phase.

Static radio dot deployment in a dynamic production cell. Following the static phase, the production cell begins dynamic simulation with robot orders (see Figure 2.5). The *reward* is adjusted to penalize radio dot re-positioning during this phase. To prevent overfitting for the initial phase's exact duration, a random seed is generated within a specified range for the length parameter.

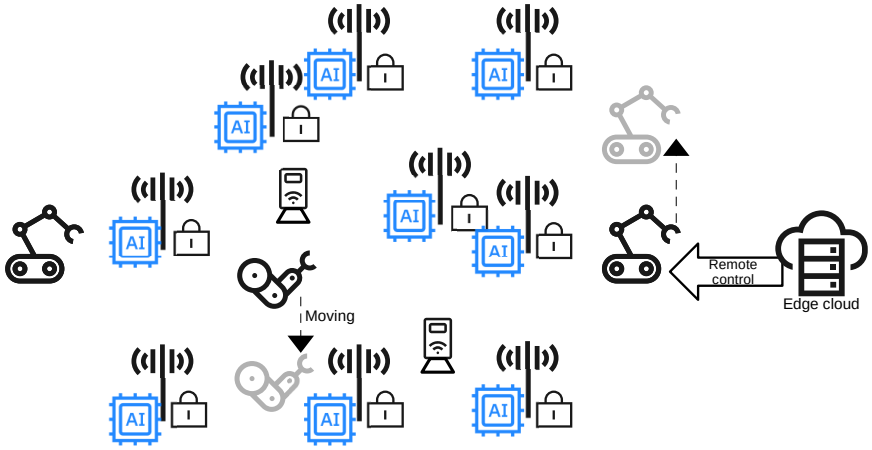


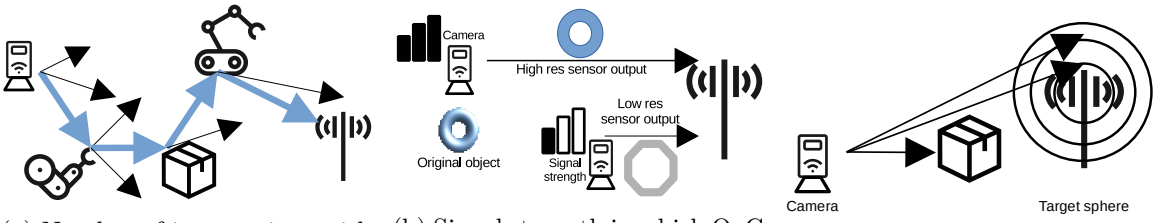
Figure 2.5: Static radio dot deployment in a dynamic production cell

Dynamic radio dot deployment in an operating (dynamic) production cell. In this scenario, radio dots can adjust to the operational production cell. I acknowledge that implementing this through physical actuators is costly, but a feasible alternative is Software Defined Antennas capable of altering antenna characteristics.

2.5.1.2 Task-space from continuous domain.

Each of the above scenarios are trained with the following embedded curricula. The radio propagation modelling has several parameters that affect the radio coverage map, and it can be related to the complexity of the task for the AI agent. In this case the task-space is from continuous domain.

Resolution of raytrace. Radio propagation modelling has several parameters that affect the resulting radio coverage map significantly. Three main parameters are the following, but



(a) Number of intersections with and reflections on objects

(b) Signal strength in which QoC phases are considered

(c) Basic antenna characteristics

Figure 2.6: Task-space from continuous domain

not limited to these:

- Number of rays per transmitter
- Maximum number of interactions: Figure 2.6a shows the number of intersections with and reflections on objects.
- Spatially bias toward expected location of receivers

Signal strength in which Quality of Service or Quality of Sensing phases are considered. The radio propagation modeling calculates path loss for each radio ray, with a correlation to signal transmission rate. Various simulators, such as multi-level models like [52] or models like Shannon capacity [53], can be used to capture this effect. I employed a physical-twin setup for radio-connected cameras, where channel quality impacts object detection accuracy. Figure 2.6b provides a high-level overview.

Antenna characteristics. Figure 2.6c shows the effect of the antenna characteristics on the radio propagation modelling. Different antennas significantly impact the AI agent's radio coverage map learning.

2.5.2 Curricula for the reward function

Reward function encodes the expert knowledge to direct the learning pace. Its main function is to minimize loss, in a limited number of steps. The reward function can be linked to curricula. I propose to consider the reward function independently from the task space curricula thus the reward function can be fine-tuned as a separate curriculum.

Setting up the reward function manually can be challenging, especially when encoding non-differentiable factors like antenna characteristics. The reward function curricula have the following traits:

1. Progression may depend on various factors, like previous rewards.
2. It does not regress to simpler curricula to avoid fluctuations.
3. In the dynamic deployment in a static production cell in Section 2.5.1.1, it receives a reward for the signal losses of the receivers if they approach an expected minimum. It receives penalty either if the radio dots remain in the same position or if the radio dots moved after the set number of steps (see Section 2.5.1.1).
4. To avoid overfitting for a specific number of steps during the dynamic radio dot deployment phase in Section 2.5.1.1, the reward function randomly varies the number of allowed steps.
5. In the dynamic radio dot deployment in an operating production cell in Section 2.5.1.1, the penalty for moving radio dots after the predefined number of steps is reduced to encourage optimization during cell operation

Figure 2.3 shows a practical approach to make the teacher node setup both the task and reward curricula in parallel for a given learning setup. For the implementation details see [28] Section 7.

2.6 Experimental results

In this section I discuss the measurements performed on the implemented system.

2.6.1 Discussion on the training

In the following subsections I discuss the behavior of the AI policies of the radio dots and the camera agents during the training phase.

I train across all 15 ARIAC competition scenarios, randomly selecting one for each policy evaluation episode. The policy is queried every 1 second based on the current observations, and an action is selected from the discrete *action space*. The 1 sec granularity is applied due to 1) its fine-enough granularity for this use case; 2) it is not connected to an event in the robotic cell, which is difficult to track requiring watching various topics and the environment, and 3) it is not too fine-grained that may intervene with a channel switching method on the radio that can cause packet reordering in certain implementations. Every sensor got its own receiver, and I added two radio dots over the environment that can be moved on a grid. Two radio dots are chosen for the evaluation due to the fact that 1) one radio dot cannot provide full connectivity for the whole production cell; and 2) providing more radio dots

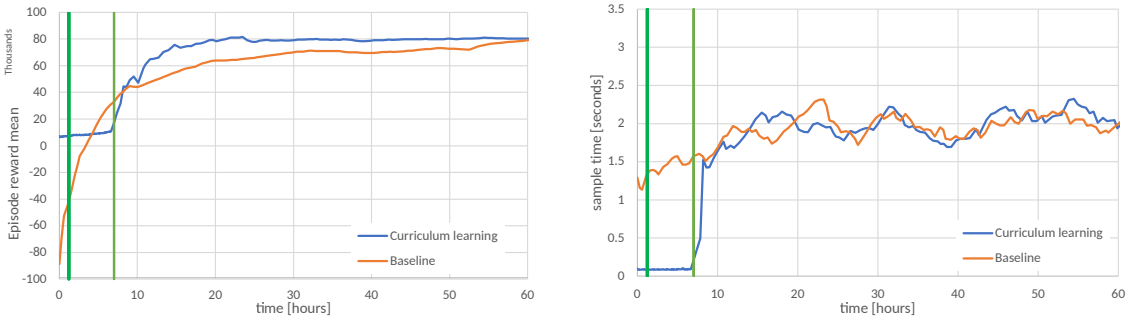
than the absolute necessary results in high connectivity ratios (low pathloss) on multiple paths which are difficult to differentiate in the reward function.

2.6.1.1 The AI policy of the radio dots

What can we see on the episode reward mean plots? Figure 2.7a shows the episode reward mean in the function of the training duration. The measurement data is collected in the RL's tensorboard. The red line shows the baseline case when no curriculum learning is applied. The blue line shows the case when curriculum learning is applied to speed up the learning rate. The green vertical lines indicate the introduction of a new curricula.

- Curriculum 1: The 0 to the first green vertical line, 0-1 hour: Dynamic radio dot deployment in a non-operating (static) production cell (described in Section 2.5.1.1 in detail)
- Curriculum 2: The first to second green vertical line, 1-7 hours: Relaxing the overfitting for a specific length of the dynamic radio dot deployment phase (described in Section 2.5.1.1 in detail)
- Curriculum 3: After the second green vertical line, 7-60 hours: Static radio dot deployment in an operating (dynamic) production cell (described in Section 2.5.1.1 in detail)

Note that the remaining curricula – selection of various agile demands scenarios (described in Section 2.5.1.1 in detail) and task space from continuous domain (described in Section 2.5.1.2 in detail) are applied during every training phase as a way of domain randomization. The *explanation* is the following. In practical implementation, curriculum learning (referred to as "curriculum 3") and the baseline case share the same training environment. However, there is a fundamental difference in their initial conditions. While the baseline case starts with an empty policy and learns from scratch, curriculum learning begins with a pre-trained policy obtained from earlier curriculum stages (curriculum 1-2). It is important to note that the rewards in the curriculum learning phase do not include the large bonus values awarded upon successful completion of ARIAC (an evaluation metric), which makes direct comparison with the baseline case difficult. Curriculum 3 quickly surpasses the performance of the baseline case within just two hours of training. It continues to improve and reaches a final policy after approximately 20 hours of training. On the other hand, the baseline case takes around 60 hours of training to achieve the same level of reward. This demonstrates that curriculum learning, with its incremental learning approach, enables accelerated learning and quicker convergence compared to the baseline case.



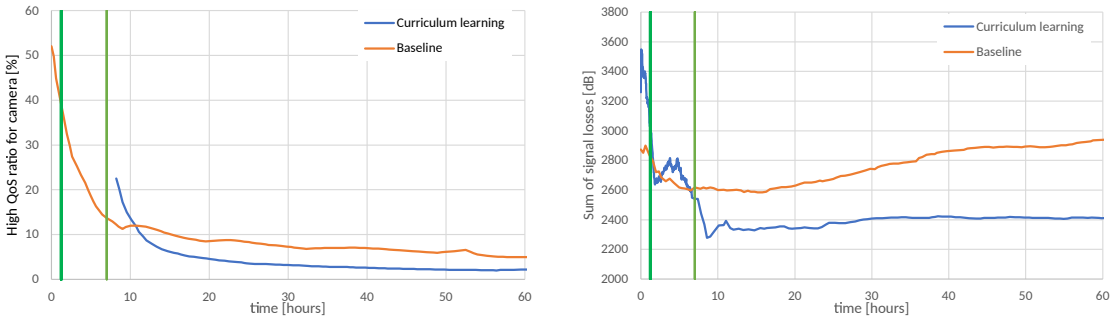
(a) Episode reward mean (b) Sample time in function of training duration

Figure 2.7: Evaluation of the AI policy of the radio dots

What can we see on the ARIAC sample time plots? Figure 2.7b shows the sample time in the function of the training duration. In curriculum 1-2 only the radio propagation calculation is performed on the initialized production cell. One run of the whole scene takes approx. 50-100 ms on average. There are 15 steps on average to let the radio dots move around to find the proper deployment. Consequently, the total duration of these phases is 15*100=1.5 sec. In curriculum 3, the radio dots' policy is queried in every simulated second, but due to the computer load, it runs at approx. 50% of real time factor resulting in a sample time around 2 secs. The episodes in curriculum 3 can last a maximum of 300 secs encompassing complex trials with multiple faulty products. As a result, it takes approx. 300*1*0.5=10 mins to collect an episode in curriculum 3. The *explanation* is the following. Comparatively, curriculum 1-2 operates at a significantly faster pace, approx. three orders of magnitude faster than curriculum 3, since the production cell control is not active during these stages.

2.6.1.2 The AI policy of the cameras

What can we see on the plot of requested high QoS ratio for the camera? Figure 2.8a shows the high QoS ratio for the camera. Note that there is no camera sensor operating during curriculum 1-2, thus there is no data for that part. The curriculum learning based solution converges to approximately 2.12% high QoS ratio, while the baseline is 4.97%, more than double the high QoS ratio that it requests. The *explanation* is the following. In the analyzed period, the baseline could not achieve the same performance as the curriculum-learning based approach, but with longer training it is expected to approach the same QoS ratio.



(a) The requested high QoS ratio for the camera (b) The sum of signal loss to the cameras

Figure 2.8: Evaluation of the AI policy of the cameras

What can we see on the plot of sum of signal loss to the cameras? Figure 2.8b shows the sum of signal loss between the transmitters and receivers. The *explanation* is the following. In curriculum 1 it learns a lot on the proper positioning of the radio dots and shows a significant drop of signal loss. The baseline case has issues with convergence as there is a trade-off between cameras being in high priority good connection with minimal pathloss, and cameras which are not prioritized –, not having connection even, – with high pathloss. The trade-off is due to the case in curriculum 3, in which some cameras are in high priority, but other cameras – that might not be used for that action even – are not prioritized, resulting in increasing losses. The curriculum learning (blue line) related line shows the benefit of taking the dynamics of the production cell into account during learning. In curriculum 3 there is a significant drop of pathloss from curriculum 2 to 3, which means that a radio deployment optimized only on a static production cell performs worse in the operation environment than the one that learns on the dynamic cell as well.

2.6.2 Discussion on the inference

In the following subsections I discuss how the AI policies of the radio dots and the camera agents behave during the inference.

2.6.2.1 Dots

Figure 2.9 shows the full extent of the simulated factory floor from the top. The light grey spheres are the positions that the dots can occupy. The yellow spheres are the cameras, while the blue spheres are other non-camera sensors. The red sphere in the middle is the start position of gantry and the dots. From this point the dots move to the deep blue and deep red areas. The deep blue (first dot) and deep red (second dot) areas represent heatmaps of

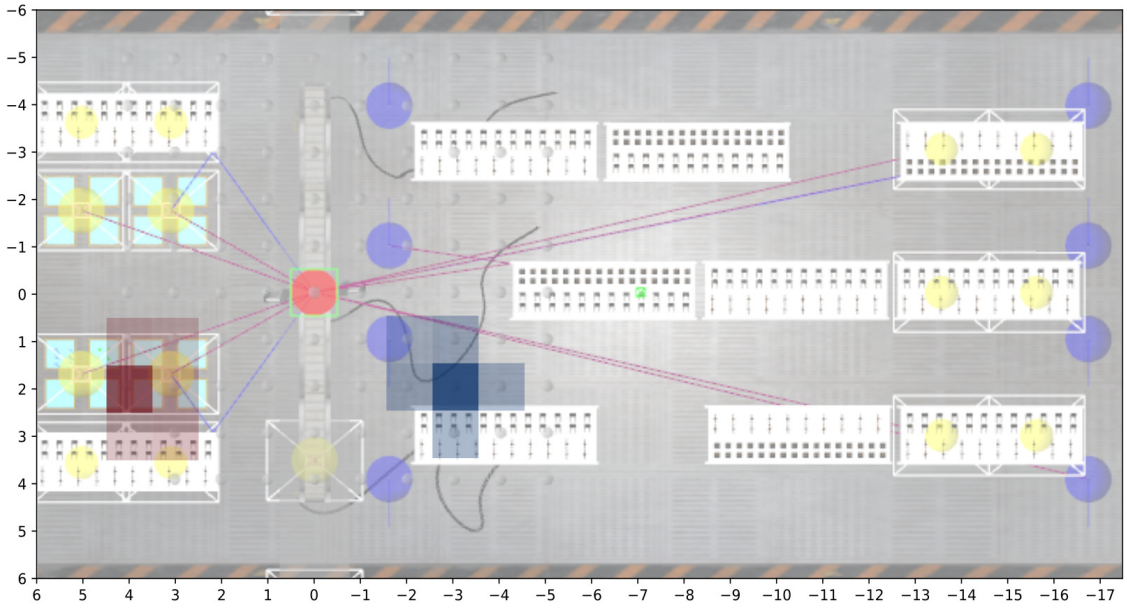


Figure 2.9: The heatmap of transmitter positions in meters (red,blue) in Curricula 1

dot positions that are the result of running an agent that learned on the static curricula. The intensifying color indicates a longer duration of the dot's presence in a specific position.

Compared to a basic dot positioning approach e.g., to move the dots in the geometric center of the production cell, it can be seen that the layout of the shelves, the conveyor belt and mainly the cameras alter the recommended position of the dots more towards the left, lower side of the layout.

In addition to heatmap of the dot positions, Figure 2.10 shows the path of the overhead gantry as it moves over the factory floor during an ARIAC scenario. The two dots are positioned even more to the lower left side of the layout. The gantry is mostly around that position, as the parts it picks up are found in the bottom left quadrant of the image, and only the drop-off points are on the top and bottom left layout.

This shows that introducing dynamic curricula can further improve the system as the agents of the dots found that these locations are more important from connectivity perspective.

2.6.2.2 Cameras

In terms of improving radio time for the AI-driven cameras, I compared the results to a baseline heuristic. This heuristic involves switching the cameras to high QoS mode whenever the gantry is within 5 meters. Figure 2.11 illustrates the operation of the 17 cameras in the

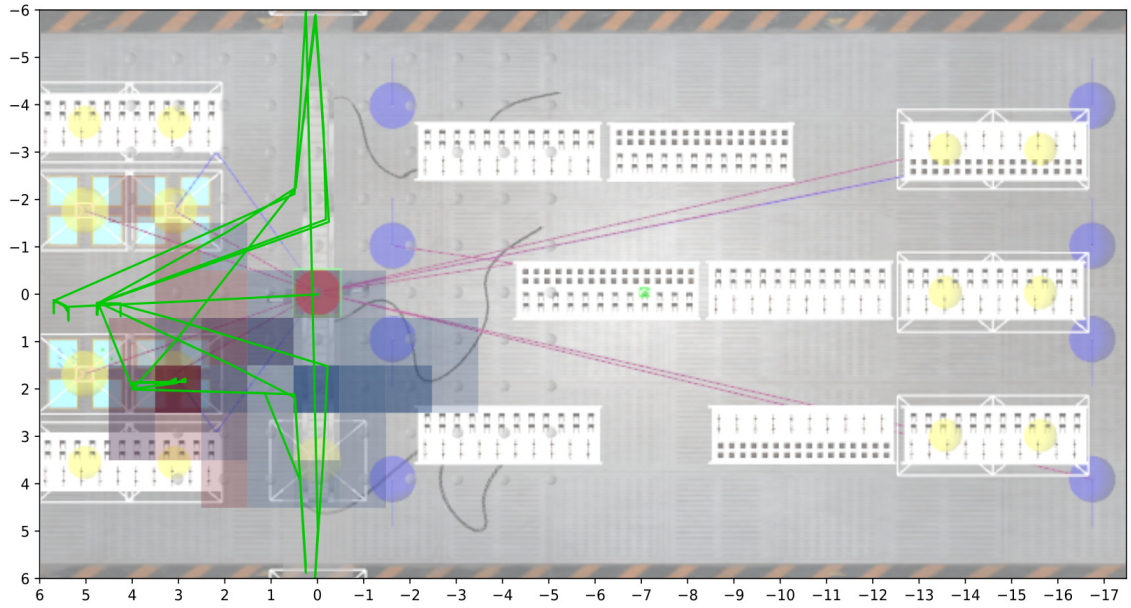


Figure 2.10: The heatmap of transmitter positions in meters (red,blue) during inference, and the path of the gantry (green) in Curricula 3

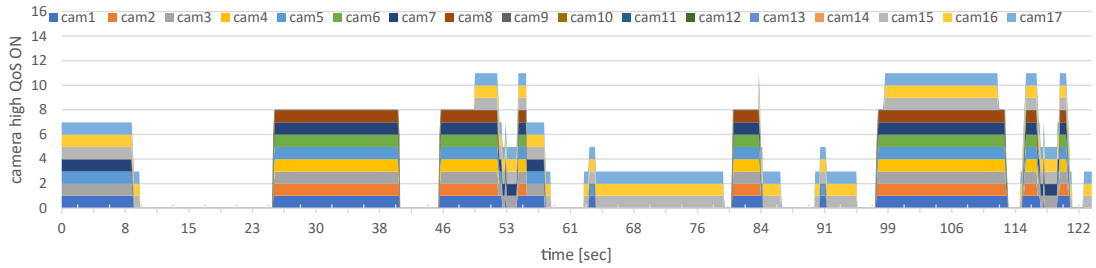


Figure 2.11: The activity of the cameras with the baseline heuristic

production cell and their status. While the figure appears as a total bandwidth diagram, it's worth noting that even during the low QoS phase, some bandwidth is consumed, depicted as a 0 on the figure. The total bandwidth of the cameras depends on the ratio of the low and high QoS bandwidth case. The total radio time of the heuristic is 44% of time the cameras are in high QoS mode compared to the case when all cameras are in high QoS all the time (100%). The heuristic results in a bursty on-off period, as it makes all the cameras switch-on around certain areas. The average number of cameras switched to high QoS mode in parallel is 4.4 per second. The average length of the high QoS periods switched on is 52.4 sec, while the low QoS periods last for 192.5 sec on average.

Figure 2.12 shows the activity of the cameras with the trained AI-agent. The total radio time is 2% of the time in high QoS mode compared to the case when all cameras are in high

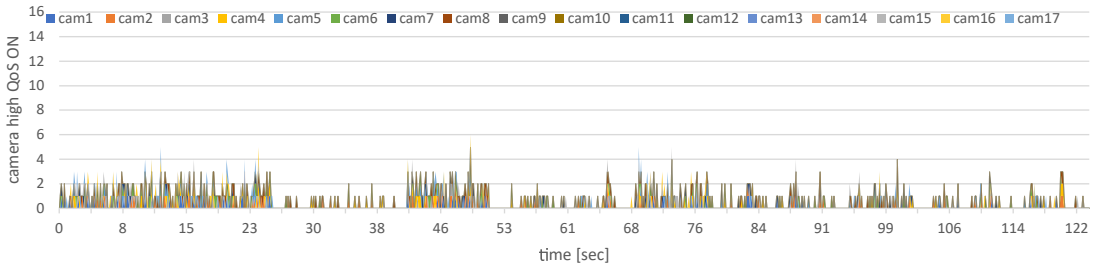


Figure 2.12: The activity of the cameras with the AI-agent

QoS all the time (100%). In this case, the average number of cameras switched to high QoS mode in parallel is 0.77 per second. The average length of the high QoS periods switched on is 1.1 sec, while the low QoS periods last for 22.72 sec on average. I can conclude that the AI-policy significantly improves the basic heuristic. It practically needs to estimate the time when the information is gathered from the part to be picked up.

Chapter 3

Automated Network Exposure Requests within ROS2

3.1 Introduction

The 5G networks are designed to support new use cases beyond consumer-oriented mobile broadband: critical Machine Type Communication (MTC), and massive MTC. Critical MTC use cases require very low bounded latency and high reliability, while Massive MTC favors enhanced coverage and long device battery lifetime for sensor type use cases involving massive amounts of devices. A single 5G network can support the diverse service quality requirements of mobile broadband, critical MTC and massive MTC use cases.

My motivation is to investigate the feasibility to apply SEAL for the industry verticals in an even easier and more automatic way than it is defined in current 3GPP standards. To reach out to a widespread community I design a solution for the users of the ROS2. I implement my solution for ROS2 and FastDDS and evaluate my method in various scenarios and deployments. My proposed solution consists of an information collecting proxy node and a node performing the mapping from ROS2 settings to 3GPP SEAL requests. I can deploy these components in three significantly different ways, a mode supporting that 1) the vertical application is the consumer of network exposure, 2) the network as the consumer of application exposure and 3) a tunnel mode. The second deployment option is a novel concept to improve the action radius of the network. The third deployment option is an aid to overcome some limitations of current network deployments with support on the application side.

I cover three SEAL functionalities: group management, network resource management and connection monitoring. I provide a solution to map the ROS2 application layer information

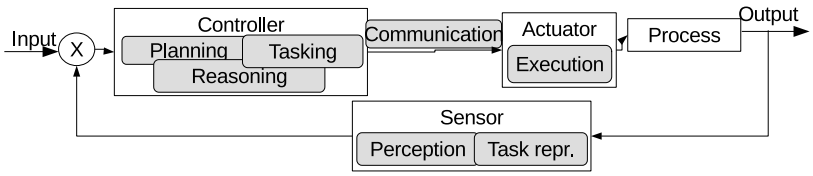


Figure 3.1: The closed-loop control model of the network resource management

elements automatically to these requests. The ROS2 application developer's only job is to tag the ROS2 topics that need special 3GPP QoS handling. I prove that my proposed solution is feasible and lightweight. The code is available as a ROS2 package in [54]. The demo video of the proposed system in action can be seen at [55].

3.2 Vertical applications as consumer of networks exposure

Exposure consumers are the IIoT applications, which are software entities that use the 5G exposure interfaces' services, whereas exposure producers are the 5G functions that provide the services that are exposed to consumers.

In Institute of Electrical and Electronics Engineers (IEEE) P2940 Standard for Measuring Robot Agility, the modeling of the production cell is introduced as a closed loop control (see Figure 3.1) with the ten agility aspects involved in the specific components [56]. The industrial process is considered as a system-in-a-system way, thus the network components i.e., the links between the boxes are considered also as a closed-loop system. The components of this closed-loop system are discussed in the following sections.

3.2.1 Sensor – Connection monitoring

I need to collect information from the ROS2 application to make it feasible to define the network requirements. Figure 3.2 shows the various layers in the network stack that can be utilized to collect the necessary information elements for the network exposure requests. The figure shows the part of the 3GPP SA6 architecture discussed in the chapter with the ROS2 application stack. (Each box is discussed in detail in the further sections.)

I start from the application layer and discuss the options going down in the protocol stack layer by layer. The network requirements are the consequence of the application-level settings of the application developer and the channel characteristics. The first can be set up and queried directly at both the publisher and subscriber side, while the channel characteristics need to be measured on various layers of the protocol stack.

3.2.1.1 Network requirements set by the application developer

The application's bandwidth and packet Inter-Arrival Time (IAT) is the consequence of the ROS2 topics' message type, packets' size, the publishing rate and sampling interval.

Network requirements of the application. The ROS2 application developer defines the behavior of the publisher and the subscriber. This behavior is only valid on the publisher site. It is important to note that the DDS layer has a feature to support filtering data on a DDS topic [57], though it is not exposed for the ROS2 layer. The issue with this feature is that the filtering is happening at the subscriber. This means all topic data must be transmitted first on the radio channel. After everything arrives successfully, the DDS layer can filter the content of the topic with various string-matching methods and provide to the ROS2 application only the filtered content. This can reduce the load on the ROS2 application – though performance-wise it is also user-space code; thus, it is more convenient for the application developer rather than any performance gain –, but it does not provide any gain on the radio channel.

ROS2 QoS settings. A wide range of QoS policies are available in ROS2 and let the user fine-tune node connectivity. With the correct combination of QoS settings, ROS2 can have a wide range of potential states, ranging from best-effort UDP to TCP-like reliability. ROS2 leverages the flexibility of the DDS transport it utilizes, which proves advantageous in scenarios characterized by unreliable wireless networks, where a "best effort" approach is more suitable. It also proves beneficial in real-time computing setups that demand specific QoS profiles to ensure timely completion of tasks [58].

3.2.1.2 Measured connection related KPIs

Essential information in terms of network statistics and the QoS of the connection are latency, jitter and packet loss. While in the previous section the network flow statistics can be collected on the publisher side to estimate the bandwidth of the application, the influence of the subscriber's setting and the transmission channel needs to be measured.

Figure 3.2 shows the ROS2 protocol stack on top of a 3GPP network from SA6 perspective. The gray boxes highlight the possible approaches to gather network statistics in various layers of the system. The information collection method in the specific layers is discussed in the following paragraphs.

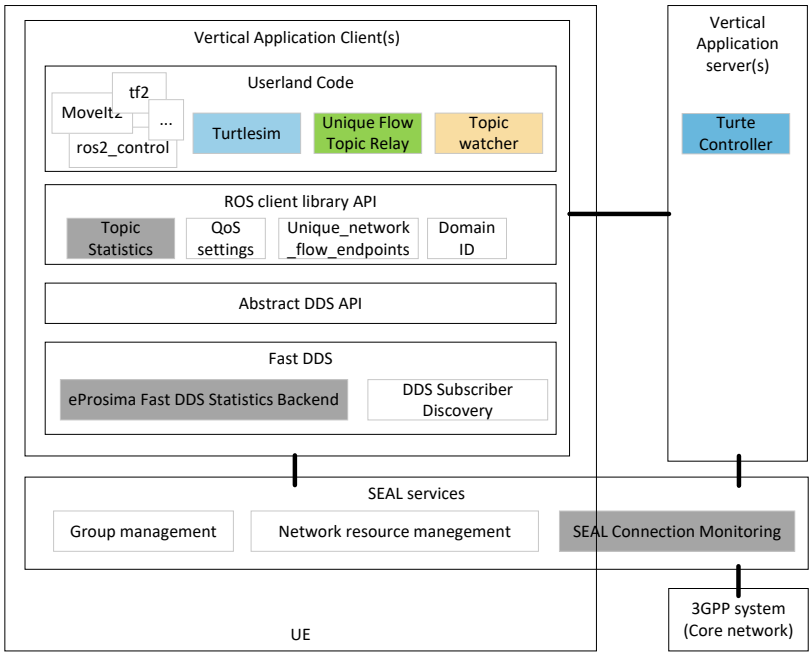


Figure 3.2: The ROS2 protocol stack on top of a 3GPP network from SA6 perspective /gray boxes show the network statistics modules/

Userland code. An option to measure the ROS network statistics can be done in the userland code. This can be done by reading out the settings that the application developer set up for the application. As it is not a generally applicable solution, I aimed to perform the network statistics collection which is transparent for the application developer. I need the ROS topics bandwidth and publishing rate which can be similarly extracted as the command line ROS topic tools, the `hz, bw` operates [59]. The statistics collection is done by summing up the message sizes and their IAT time in the subscriber. The tool developed to perform the collection of required information is discussed in Section 3.2.2.1 in detail.

ROS client library API. The integrated statistics measurement for messages received by any subscription is provided by ROS2 via a statistics topic [60]. Allowing users to get subscription statistics gives them the ability to assess the system's performance or help with problem identification. The received message age and received message period are provided by the measurements.

Topic Statistics measures are disabled by default. Both the received message age and the received message period measures are enabled for that particular subscription once this feature has been activated for a certain node via the subscription configuration options. Data age can be calculated for those subscriptions which message type contains timestamp

in the header field, otherwise empty data is published. Only C++ is currently supported for this capability in ROS2 Humble (rclcpp).

The benefit of this approach is that we are in the user space application, and this method measures the same QoS KPIs as the application would perceive. The main issue with this approach is that it is not a transparent solution. It requires the user to modify its code by enabling the subscription feature and forcing them to use time stamped messages.

I tried many designs to use this feature in the Unique Flow Topic Relay (UFTR) (see Section 3.2.2.1 in detail). The difficult part is that the subscription node must have a predefined message type during compilation. I checked the examples of the statistical library which made format checking by templates still in the compilation time. I also examined the source code of "RelayField" from ROS2 "topic tools" [61] which showed that the message type is still a rigid requirement just not in compilation time as it is in Python. I considered the application of Deep Packet Inspection (DPI) and parsing the headers of the messages as if there were timestamps. I decided that this could give misleading results and dropped the idea. All in all, this solution does not seem to be a viable candidate to include in UFTR as it cannot support general message types.

DDS. A C++ library called eProsima FastDDS Statistics Backend [62] enables users to gather and retrieve performance statistics from a FastDDS network, such as the topology of the DDS network and the configuration of each DDS entity (i.e., QoS). Additionally, it has a FastDDS statistics module that enables data collection from the DDS layer and dissemination under certain DDS subjects. This library provides detailed statistics on the DDS level communication: latency, throughput, re-sent packets, sub-messages, meta-traffic packets, discovery time can be directly collected. The statistics module's built-in DataWriters are used to publish the collected data using DDS across certain topics. As a result, FastDDS does not compile this module by default, because doing so could slow down the application. To enable it in the ROS2 applications, the FastDDS needs to be recompiled with a CMake configuration step's `-DFASTDDS STATISTICS=ON` switch. After the successful compilation, the ROS2 abstract DDS API needs to be pointed from the default binary package install to the compiled version.

The advantage of this approach is its detailed KPIs and accuracy. The drawback is the number of steps required to enable this module. Also, full deployment is needed of these modules as during the interaction with other ROS2 nodes, which do not use the statistics enabled DDS version, will not provide any statistical data.

In my experiences, compilation of FastDDS and modifying the abstract DDS layer does

not work with all the ROS packages. Some of them have a dependency on the binary packaged DDS. The whole recompilation of ROS2 is a safe solution, but it takes a long time.

Transport network. There are three message types in TS 23.434 on retrieving information about the connectivity status. The common way is to subscribe for QoS monitoring and the network pushes down monitoring information periodically. TS 23.434 [11] §14.3.2.22 "Unicast QoS monitoring notification" is the information flow for unicast QoS monitoring notification from the NRM server to the VAL server. QoS monitoring data is an aggregate of QoS measurements obtained from the 5G System (5GS). TS 29.549 [12] §7.4.2.4.2.3 describes the measurement data. It consists of the following attributes: 1) downlink packet delay in milliseconds, 2) uplink packet delay in milliseconds, 3) round-trip packet delay in milliseconds, 4) average packet loss rate, 5) average data rate, 6) maximum data rate, 7) average traffic volume for downlink in bytes, 8) average traffic volume for uplink in bytes.

The following two messages are event notifications from the network. TS 23.434 [11] §14.3.2.15 "QoS downgrade indication" is a message from the NRM client to the NRM server. The report includes the expected or actual QoS or Quality of Experience (QoE) parameters which were downgraded (i.e., latency, throughput, reliability, jitter).

TS 23.434 [11] §14.3.2.16 "Application QoS change notification" is a message from the NRM client to the NRM server. It contains information on the updated or requested QoS parameters for the end-to-end session based on the QoS change on one or both links involved in the network-assisted end-to-end communication.

Note that the way the network retrieves the information is not defined in TS 23.434 [11]. It is possible to involve the Network Data Analytics Function (NWDAF) [63] or make active probing on the established connections.

The advantage of this approach is that it is generally applicable on the 5G network. The drawback is that these measurements refer to the whole connection and not application, ROS2 node specific.

3.2.2 Proposed ROS2 data collection node – Implementation of the sensor

In this section I discuss the proposed proxy node that is capable of collecting the necessary information from the ROS2 and transport layers for the SEAL request.

3.2.2.1 The method to retrieve the 5-tuple identifier of the ROS2 topic

To fill in the Internet Protocol (IP) address field of the API, or a more detailed 5-tuple identifier (transport protocol, source IP, source port, destination IP, destination port) of

the IP flow on which the QoS management request is valid I need to obtain the ROS2 topic flow identifiers. [64] discusses the issue of the default ROS2 topics. The same flow identifiers (5-tuple or 3-tuple) are used by all publishers and subscriptions in communication nodes. This prevents communication nodes from explicitly differentiating network QoS for publishers and subscriptions. Thus, only the same network QoS can be assigned to publishers and subscriptions in communicating nodes.

Source IP, port. Since ROS2 Galactic, the "Support for unique network flows" is among the features [65]. To enable QoS specifications for these IP streams in network architectures that support such a feature, like 5G networks, applications may now demand that UDP or TCP and IP-based ROS Middleware (RMW) implementations provide unique network flows i.e., unique Differentiated Services Code Points (DSCP) and/or unique IPv6 Flow Labels and/or unique ports in IP packet headers.

A further feature connected with the setting up of unique network flows is a related getter interface for the RMW. ROS2 Galactic provides an API called `get_network_flow_endpoints()` for publishers and subscriptions to understand ports and IP addresses assigned for their messages by the RMW implementation. Note that Network Flow Endpoints (NFEs) established only on the publisher-side are represented by the NFEs supplied by `get_network_flow_endpoints()` for a publisher (local). It does not include details on NFEs for matching subscriptions. Similarly, the NFEs for a subscription returned by the getter function are localized and lack information about matched publishers. This is a design choice for ROS2.

Destination IP, port. The missing information elements from the 5-tuple identifier can be obtained in a FastDDS specific way. The DDS layer is aware of the destination IP and ports, it is just not exposed towards the ROS2 layer. ROS2 topics consist of 1) DDS topics with the `rt/` namespace, 2) a topic transporting meta traffic information and 3) a topic providing statistical information if it is enabled. The meta traffic topic is used by ROS2 for discovery and configuration purposes. During the DDS subscriber discovery, the `SubscriberEvent` [66] can be queried for the destination IP address and ports. I check the DDS topics and collect the 5-tuple data when the `ReaderDiscoveryInfo` changes regarding those topics which are indicated by the user to special QoS handling.

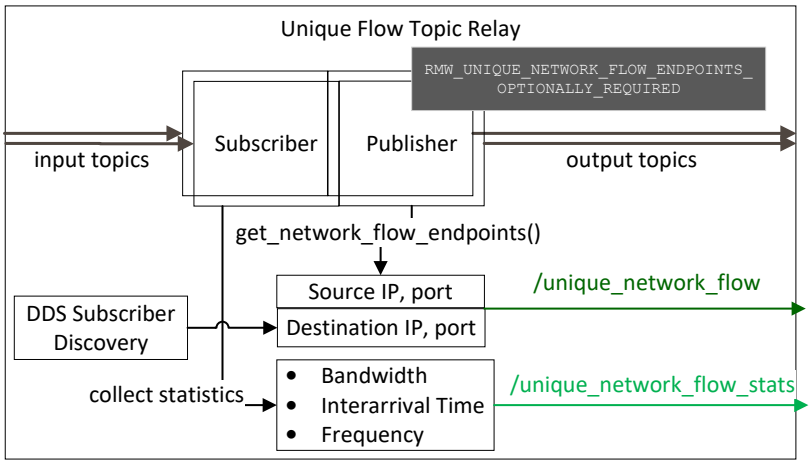


Figure 3.3: Unique Flow Topic Relay (UFTR)

3.2.2.2 Unique Flow Topic Relay

The required features are collected and realized in my tool called "Unique Flow Topic Relay". Figure 3.3 shows the architecture and information flow of the node. This tool is the extension of "ros topic tools" [61]. "ROS topic tools" are tools for meta-level guiding, throttling, choosing, and other manipulation of ROS2 topics. These utilities work on generic binary data utilizing rclcpp's `GenericPublisher` [67] and `GenericSubscription` [68], rather than serializing the streams that are being manipulated. The tools in the package are offered as composable ROS2 component nodes, which enables them to be launched from launch files, initiated from the command line, or spawned into an already running process. I extended the "relay" command line tool. The ROS2 "Relay" node subscribes to a topic and publishes all incoming data to another topic. It is compatible with all message types.

My extension for the relay node is that the re-published topics are created with the enabled unique flow option, resulting in that these topics are no longer multiplexed on the same 5-tuple as the other and I can request the 5-tuple identifier of these topics.

The intended usage of the node is that the user configures its launch file by tagging those topics that require QoS handling by my mechanism. The user can add a "_QoS" tag to the original topics of the application, and the relay node subscribes for these topics then re-publishes them, removing the tag as well. An option for the application developer is to simply remap the existing topic names with the added "_QoS" tag. In this case there is no need for any modification to the original ROS2 application.

The 5-tuple related information for each QoS handled topic is published on the "unique_ _network_flow" topic, while the network statistics related information is published on the

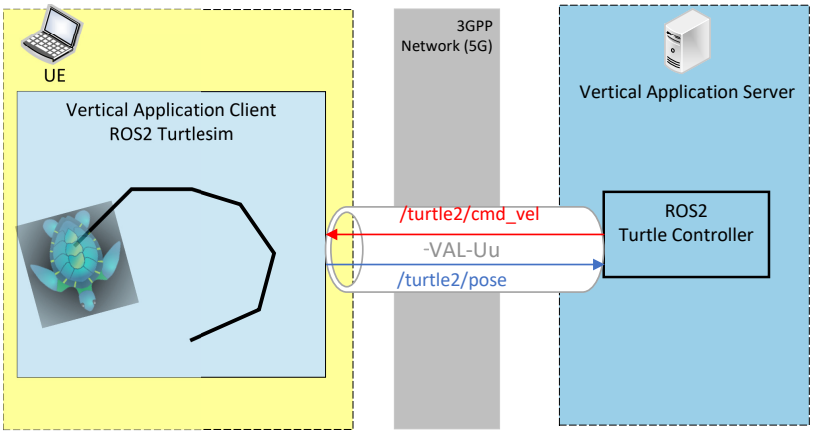


Figure 3.4: ROS2 Turtlesim with an external controller connected via 5G

"unique_network_flow_stats".

The method that provides more detailed network statistics requiring different ROS2 connection architecture is discussed in Section 3.4 and 3.5.

3.2.3 Controller – Network Resource Management

I need to map the sensor information from Section 3.2.1 to a control process.

TS 23.434 [11] §14.3.2.13 defines the "end-to-end QoS management request" from the NRM client to the NRM server. The essential information elements are the "list of VAL UE" for whom the end-to-end QoS management occurs, the "IP address" of the VAL UE and the "end-to-end QoS requirements" of the application including latency, error rate, etc. for the end-to-end session. Note that the Rel-17 definitions enable the UE-level setup of the QoS parameters. To make it future-proof with further releases, and to follow the recommendations given in [26] §4.2.3 Note 2,3 to support multiple IP flows, I provide UDP/IP-flow level identification of the ROS2 topics.

3.2.3.1 The 'Hello world' in ROS2, Turtlesim

A simple simulator for learning ROS2 is called Turtlesim [69]. It gives the user an overview of what ROS2 accomplishes at its most fundamental level so that they can subsequently use a real robot or a robot simulation. Figure 3.4 shows the basic architecture of Turtlesim with an external controller connected via 5G. The use case is that I spawn a turtle, and it is controlled via closed-loop control by an external control node to move along a circle. In this basic setup there are two unidirectional topics established between the controller and the Turtlesim. Turtlesim publishes the position information; the controller subscribes to

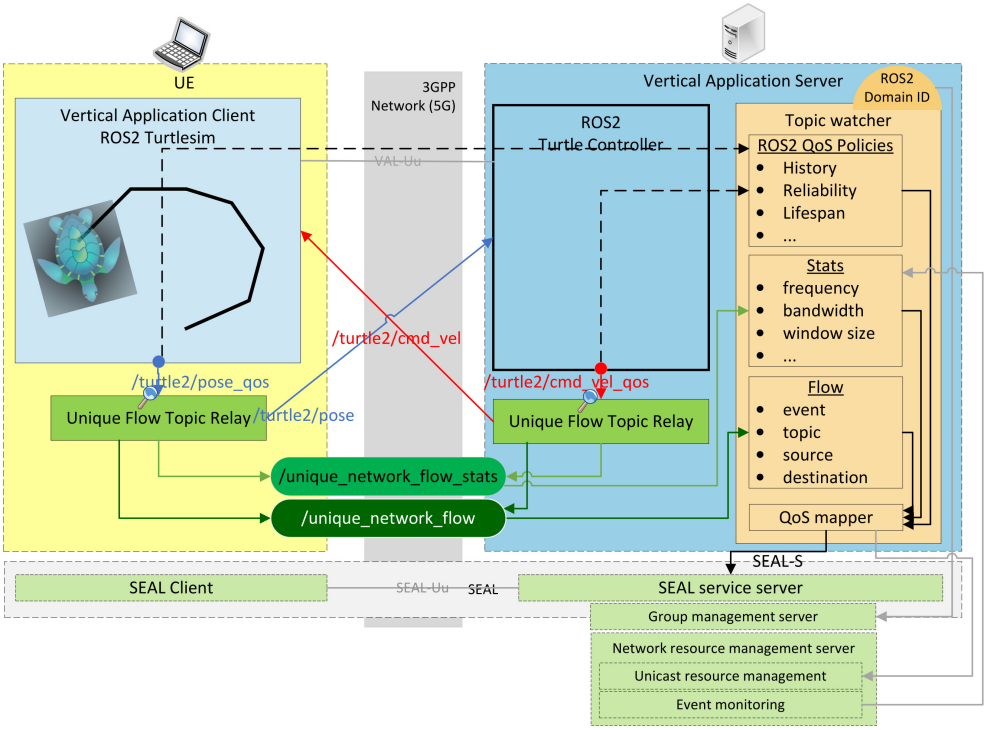


Figure 3.5: ROS2 Turtlesim with UFTR deployment

this topic ("/turtle2/pose"). Meanwhile the controller publishes the velocity command topic ("/turtle2/cmd_vel") and the Turtlesim subscribes to it.

Figure 3.5 shows the extension of the basic use case with the UFTR nodes that enables the QoS handling of both the position and the velocity topics. The following section discusses the orange box in the figure.

3.2.3.2 The method to define QoS out of the connection monitoring data

Until now, I identified the flow identifier of the topic which needs QoS handling. In the following steps I discuss how the QoS handling is achieved.

3GPP TS 23.203 [70] Table 6.1.7 enumerates the standardized QCI characteristics in terms of Resource Type (RT), Priority Level (PL), Packet Delay Budget (PDB), Packet Error Loss Rate (PELR), Maximum Data Burst Volume (MDBV) and Data Rate Averaging Window (DRAW) with given example services also. The QCI is a parameter used in Fourth generation of mobile communications (4G) and 5G wireless networks to classify and prioritize different types of data traffic. Each QCI value represents a specific class of service, with different characteristics and priorities. The QCI value is assigned to each user session or data flow within the network, allowing the network infrastructure to treat different types of

traffic differently. The mapping of a QCI for a certain input parameter set is given by the table. This step is indicated by "QoS mapper" in Figure 3.5. The open question remains how to set up the input parameters based on the available connection information. Listing 3.1 shows the important part of the QoS mapping code that I discuss in detail.

```
def __init__(self):
    super().__init__('subscriber')    self.create_subscription(UniqueTopicFlow,'
    unique_topic_flow', self.listener_callback, 10)
    self.create_subscription(UniqueTopicFlowStats, 'unique_topic_flow_stats', self.
    listener_callback_stats, 10)
def listener_callback(self, msg):
    if msg.event_type == 1: #ADDED
        # src, dst, hz, bw
        active_topics[msg.topic] = (msg.source, msg.destination, 1, 1)
    if msg.event_type == 2: #REMOVED
        active_topics.pop(msg.topic)
def listener_callback_stats(self, msg):
    for i in msg.stats:
      res = active_topics.get(i.topic)
        if res is not None:
            src = res[0] \n\t dst = res[1]
            current_rate = res[2] \t current_bw = res[3]
    if qos_profile.reliability is QoSReliabilityPolicy.RELIABLE: PELR = 1e-5
    elif qos_profile.reliability is QoSReliabilityPolicy.BEST_EFFORT: PELR = 1e-2
    PDB = 1. / i.window_rate * 10e3
    if PDB > qos_profile.lifespan.nanoseconds / 10e9:
        print("WARN: ROS2 lifespan setting is too low")
    if abs(current_rate/i.window_rate-1)<0.1: RT = "GBR"
    else: RT = "Non-GBR"
    if i.window_duration_secs < 2:
        print("WARNING: DRAW is not enough")
        continue
    MDBV = i.num_bytes/i.num_messages
    if RT == "GBR" and PDB <= 100 and PELR <= 10e-2:
        QCI = 1
        ExampleServices = "Conversational Voice"
        hit = hit + 1
    ...
    active_topics[i.topic] = (res[0], res[1], i.window_rate, i.window_bandwidth)
```

Listing 3.1: QoS mapping code

"Topic watcher" subscribes to both the "unique_flow" and "unique_flow_stat" topics (see Figure 3.5). The topic "unique_flow" provides information in case a new topic becomes active or if it has ceased to exist i.e., there are no more publishers or subscribers for the topic. In case these events occur, an update is published with the 5-tuple identifier of the topic's IP flow. A dictionary stores the list of active topics. The "unique_flow_stat" topic

receives network statistics data on the active topics in every 10 secs. The 5-tuple for the active topic and the statistic from the previous window is retrieved from the dictionary.

The PELR is mapped from the ROS2 QoS reliability. ROS2 have two QoS reliability settings:

- Best effort: make an effort to deliver samples, but if the network is not reliable, they could be lost.
- Reliable: provides a delivery assurance and several retries.

3GPP TS 23.203 [70] Table 6.1.7-B has QCI values for automation in which the most reliable values are in 10^{-5}, which is given for the reliable ROS2 topics and 10^{-2} is considered for the best effort ROS2 topics.

PDB is set directly by the measured packet generating frequency of the ROS2 application. The lifespan duration set by ROS2 QoS policy is the longest period that can pass from the time a message is published and when it is received without the message being considered stale or expired (expired messages are silently dropped and are effectively never received). I give a warning if the measured PDB is higher than the lifespan duration as it means that the ROS2 application considers them as expired.

I compare the data sending rate of the ROS2 topic in the current and previous measurement window. If their difference is within 10% I consider the topic as Guaranteed Bitrate (GBR) otherwise as a Non-GBR topic which is set to the RT for the QCI mapping.

The DRAW has to be at least 2000 ms according to 3GPP TS 23.203 [70] Table 6.1.7-B. If the measurement window is smaller than that I pop up a warning.

The MDBV is estimated from the average packet size in the measurement window. Extending the statistics message could directly provide this information. 3GPP TS 23.203 [70] Table 6.1.7-B use it rarely. Basically, two types of MDBV packets are considered: one for small packets below 255 bytes, and one with Maximum Transferable Unit (MTU) with 1354 bytes.

An optional information that can be used is the lease duration of the ROS2 topics to provide indication to the publisher that it is alive before the system considers it to have lost liveliness. Losing liveliness could be an indication of a failure. A callback is triggered by the ROS2 system in case the maximum time of the lease duration is reached (see [71] as an example). The 3GPP system can utilize this to remove the flow related requests of the ROS2 topic e.g., NRM requests, network monitoring subscriptions explicitly. It is done automatically by the 3GPP system, but in large deployments, it can reduce the load on the 3GPP nodes.

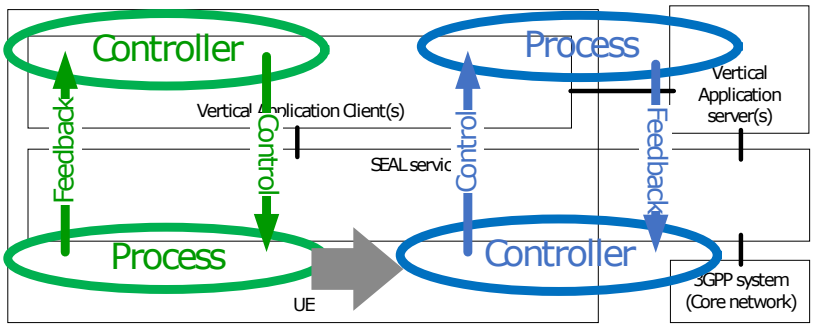


Figure 3.6: The control loops in the "vertical application as consumer of network exposure" (Section 3.2) vs "the network as consumer of application exposure" (Section 3.3)

The rest of the algorithm goes through the 3GPP TS 23.203 [70] Table 6.1.7 as a set of conditional switches. Note that the rest of the ROS2 QoS parameters are DDS specific or ROS2 application specific (e.g., deadline) and have no relevance in the radio transport network setup. The following ROS2 QoS policies are not considered: 1) history: the strategy how the samples are stored, 2) depth: the number of samples to be stored, 3) durability: check if the publisher is the responsible for persisting samples for “late-joining” subscriptions, 4) deadline: the maximum time that should pass between posting new messages to a topic. Finally, the stats are updated in the dictionary.

3.3 The network as consumer of application exposure

The current design of 3GPP SEAL provides both information and control on the network towards the vertical application, but eventually to be able to provide smart network solutions, the other way of information and control is a desired state in the future. In this way the exposure consumer is the 5G network, whereas exposure producers are the vertical application functions that provide the services that are exposed to consumers (see Figure 3.6).

In another interpretation, the previous section discussed rigid VAL applications and adaptive networks, while this section extends the capabilities of VAL applications with adaptive behavior in cooperation with the network.

The NRM requested by the VAL application worked in an open-loop manner so far. We had no sensor to check the effects of the NRM request on the VAL application performance. In this section my goal is to close this control-loop.

In this section I introduce the Vertical Application Service Enabler Architecture Layer (VASEAL) that exposes the vertical application layer towards the network. Figure 3.7 shows the concept. Similarly, as in SEAL, VASEAL has four main components: VASEAL Service

Server and Client and their respective network side nodes, the Network Layer Service Server and Client. VASEAL Service Server runs next to the Vertical Application Server hosting the Resource Management Server and Event Monitoring for the production cell or other VAL application. VASEAL Service Client runs next to the Vertical Application client running the Graphical User Interface (GUI) for the data collection of the event monitoring. The Network Layer Service Client runs next to the SEAL Client. The Network Layer Service Server runs next to the SEAL Service Server.

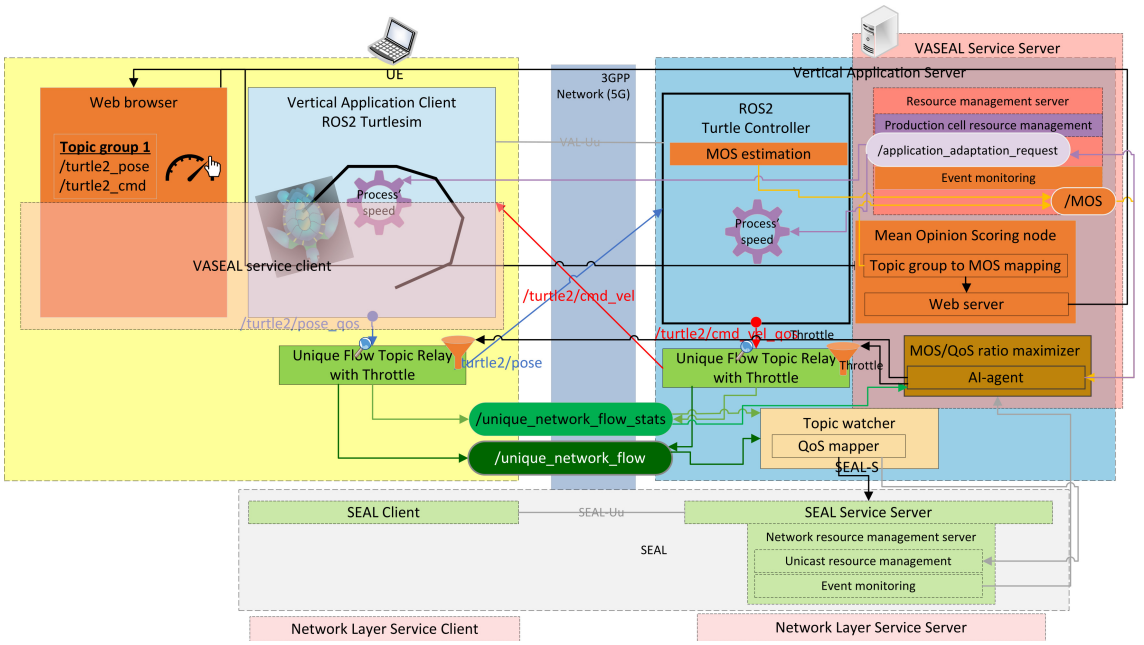


Figure 3.7: The architecture of the ROS2 Turtlesim with UFTR with throttle and Mean Opinion Score (MOS) deployment

In connection with P2940, the VAL process, like the Turtlesim, is considered as a closed-loop control process in which the certain elements are discussed in the following sections.

3.3.1 Sensor – ROS2 process related KPIs

The main idea to reduce the needed infrastructure can be the assumption that if the product does not have significant quality impact due to the looser network requirements (see mentioned examples), then significant cost savings can be achieved by relaxing some QoS parameters of the network.

To achieve this, I introduce an interface, a "MOS" topic which can be populated in the following two ways.

3.3.1.1 MOS provided by the user

I intend to demonstrate in this section that industrial processes can be examined one-by-one based on their network performance requirements, and their performance can be evaluated by fast expert opinion leaving out the tedious low-level process specific KPI measurements. This option involves a human expert in the evaluation. The "Mean Opinion Scoring" node hosts a web server which communicates with a browser via websocket. The user is provided with an interface on which all the topics relayed by the UFTR in the system are enumerated. The user can group the topics by a self-defined ID which represents that a set of topics is responsible for a certain perceived MOS. After the grouping of the topics, the user can set a MOS value for the process on a scale 1-5. The websocket sends back the selected value associated with the list of topics and the "Mean Opinion Scoring" node publishes it on the "MOS" topic.

3.3.1.2 Application with built-in MOS estimation

Beside the human focused mean opinion scoring, it is also common to estimate the MOS automatically to provide high level feedback to the content provider based on measured network QoS related KPIs e.g., [72],[73]. As a demonstration I included an automatic estimation for the controlled turtles in the Turtlesim. I extended Turtlesim with a function to compare the Proportional-Integral-Derivative (Proportional-Integral-Derivative (PID)) error or movement resolution of the two controlled turtles. After every 10 Hz difference in the control loop of the two turtles, the MOS score of the slower control loop is reduced by 1 unit. This is calculated in the background and published on the "MOS" topic.

In case both the human expert mean opinion scoring and automatic MOS estimation publishes on the MOS topic, the human-based is considered only. The "MOS" topic publisher has an identifier field to signal whether it comes from the human scoring system or from the automatic one.

For the MOS estimation of further use cases, see Chapter 5.

3.3.2 Controller – Application behavior influencing interface

From an application's i.e., a robot controller's perspective, the bandwidth is a consequence of the control packet size. While in audio and video encoding there are various codecs and compression rates which significantly alter the required bandwidth, in terms of control messages e.g., ROS2 standard twist message there are two vectors only which is difficult to compress further. The resolution can refer directly to the control frequency e.g., a haptic

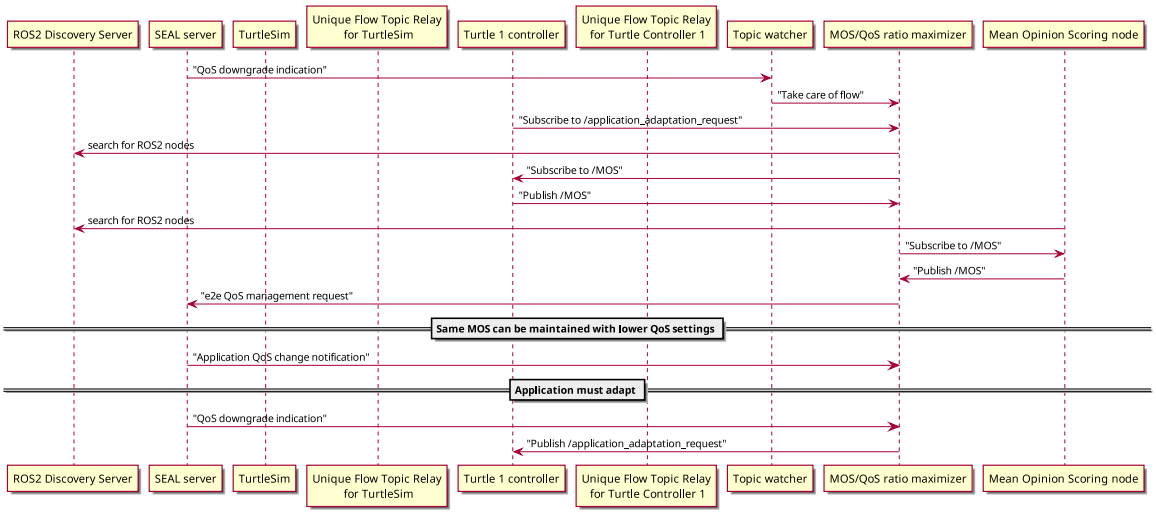


Figure 3.8: The communication steps between the ROS2 and SEAL nodes when communication resources are depleted

device has higher control frequency or resolution than an industrial heavy duty robotic arm. The frame rate in video codecs requires various playback speeds for the specific frame rate. In case there is inadequate network bandwidth to download the 50 Frames Per Second (FPS) video in real-time then playing out with 25 FPS frame rate makes more room for the network to buffer with the doubled play-out time. We can interpret this from the VAL applications point of view as slowing down the application till there is no significant degradation of production KPIs, e.g., cycle time can relax the network requirements. Assuming the VAL application can work similarly to the Dynamic Adaptive Streaming Over HTTP (DASH) client, it would mean that the VAL application adapts to the network characteristics. This behavior is only required if the network's resources are depleted. To achieve this the network requires clean interfaces on the VAL application for which the VASEAL architecture provides an approach.

3.3.3 Network resources are depleted

TS 23.434 [11] §14.3.2.16 "Application QoS change notification" types of messages can arrive from the NRM server to the NRM client.

3.3.3.1 ROS2 implementation

Figure 3.8 shows the communication steps between the ROS2 and SEAL nodes when notification events occur from the network. In case the network resources get depleted, the SEAL server sends "QoS downgrade notification" to the "topic watcher". The "topic

watcher" notifies the "MOS/QoS ratio maximizer" node to spur into action which opens a topic "/application_adaptation_request" for which the "Turtle 1 Controller" subscribes. The "MOS/QoS ratio maximizer" node subscribes to the "/MOS" topic on which the "Turtle 1 Controller" publishes estimated MOS values automatically without human interaction. The "Mean Opinion Scoring node" is launched which also publishes on the "/MOS" topic for which the "MOS/QoS ratio maximizer" node subscribes. These MOS scores are set up by human experts. The "MOS/QoS ratio maximizer" calculates actions on keeping the MOS acceptable but reducing the requested QoS and the "e2e QoS management requests" are sent towards the SEAL server.

There are two outcomes of these requests. One, the SEAL server can fulfill the requested QoS and responds with "Application QoS change notification". In this case we reached a new operation point. Or the second case, when the QoS requests still cannot be fulfilled and a further "QoS downgrade notification" arrives from the SEAL server. In this case the "MOS/QoS ratio maximizer" node publishes an adaptation request towards the "Turtle 1 Controller" via the "/application_adaptation_request" topic to slow down and lower the frequency of the control loop.

3.3.3.2 The MOS/QoS ratio maximizer in the VASEAL architecture

The right middle brown box represents the MOS/QoS ratio maximizer in the VASEAL architecture in Figure 3.7. It is hosted in the VAL server, but it is also possible to implement as a NF. The node has access to the event monitoring and resource management features of the SEAL and VASEAL servers. In this way it can collect information and have control of both the VAL and SEAL servers. An implementation of the MOS/QoS ratio maximizer is discussed in Section 3.5.

3.4 Tunnel mode

In this section I describe a third architectural concept and a further operation mode of the UFTR. The two main features why I introduced this architecture are 1) to support the detailed network KPI measurements in the userland code and 2) to provide enhanced dynamic QoS handling for the ROS2 applications. I need to measure latency, jitter and packet drop on any general message type. I examined the code of ROSBag [17] as its recorded data would make it possible for an offline evaluation in terms of the QoS KPIs of the topics' traffic. The command line utility "ros2 bag" is used to record data published on subjects in the ROS2 system. It compiles the information shared on a variety of subjects and stores it

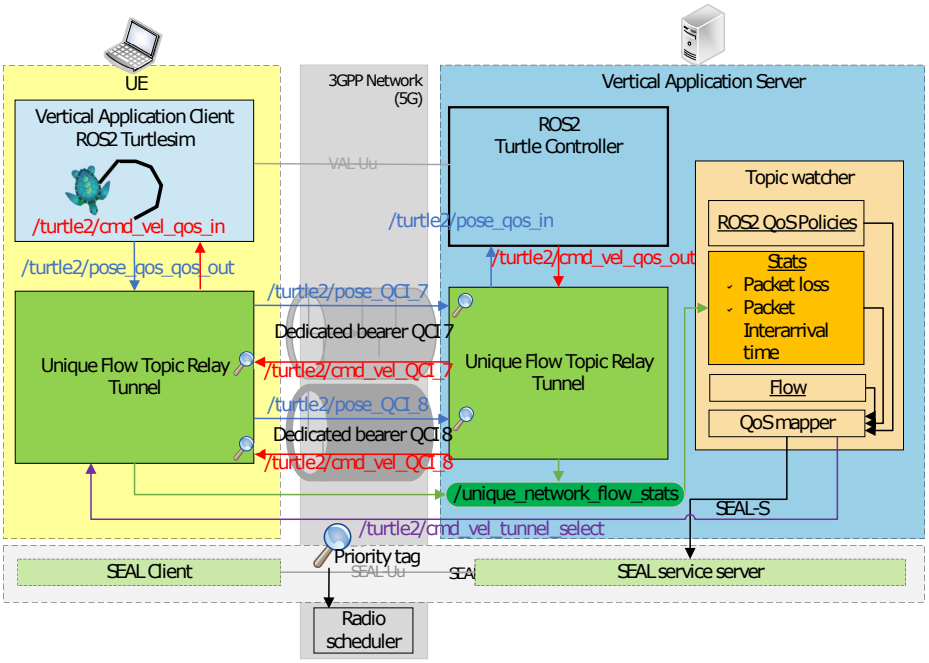


Figure 3.9: ROS2 architecture with tunnel

in a database by extending the topic messages with a header containing a timestamp. After that, the user can replay the data to get the same outcomes as they test and experiment.

3.4.1 Unique Flow Topic Relay with Tunnel

My design choice was to extend and modify the original functionality of UFTR. I connect the UFTRs' of the ROS2 application instances then the UFTRs publish the relayed topics to another UFTR instance which subscribes to it creating a tunnel between the two (see Figure 3.9). This design choice allows us to extend the transferred messages with custom headers, as if we make an encapsulation of the original topics with ROS2 message headers.

Figure 3.10 shows the extended architecture of the UFTR. There are two modes of operation of the subscriber and publisher based on the function of the ROS topic. The function of the ROS topic is indicated with an additional tag (beside the `_qos` tag that identifies the flows needed for the unique flow relay features). The additional tag represents the user intended direction of the topic. First, the application publishes a topic tagged with `_qos_out`. The UFTR subscribes to it with a general subscriber meaning that no message type is needed to be set, as it is not parsed by this node. Second, compared to the way rosbag works – in which the compiled message type library is dynamically loaded to parse the message content – , a publisher is initiated with a custom message type including a

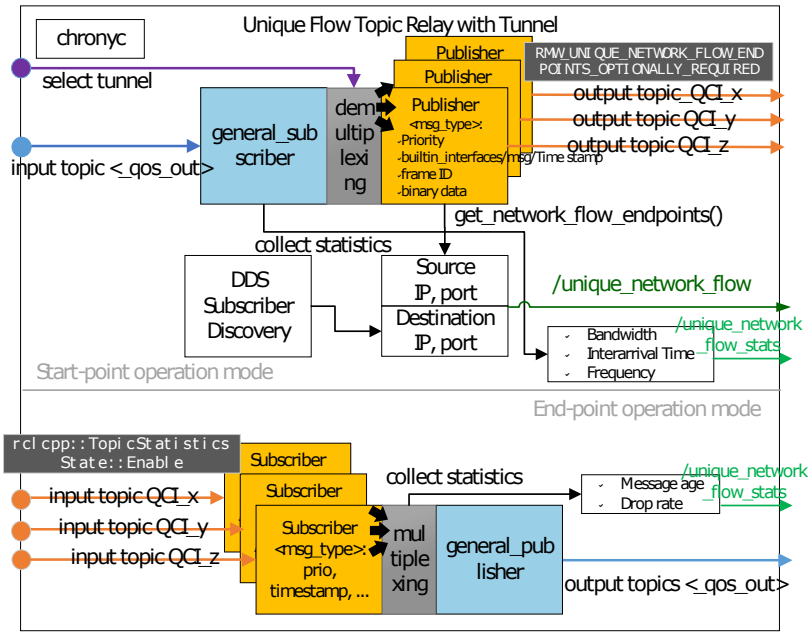


Figure 3.10: Unique Flow Topic Relay with Tunnel

timestamp, a message counter, and a binary field. The timestamp is filled with the clock, the message counter is an ever-increasing integer, and the binary data is the whole message received on the general subscriber topic. Note that there is no parsing happening in this case, thus it is message type agnostic, an encapsulation occurred only. In this operation mode of the publisher, the topic is transmitted via QoS handling, thus the unique network flow endpoint option is set.

The same UFTR has the second operation mode, in which it subscribes to the encapsulated topic. As this topic has a timestamped header, the `TopicStatistics` (see Section 3.2.1.2) feature can be enabled. The header is parsed, and the binary data stream is forwarded through a general publisher. This step is indicated in the topics naming with a `_qos_in` tag that this topic is decapsulated and can be forwarded towards the application.

Time synchronization is crucial in the estimation of message age. I used chrony [74], which is a Network Time Protocol (NTP) implementation. Chrony provides features to make the system clock be synchronized using NTP servers, reference clocks, like a Global Positioning System (GPS) receiver, or just manually entered time using a wristwatch and a keyboard. In order to offer a time service to other computers on the network, it can also function as an NTPv4 (Request for Comments (RFC) 5905 [75]) server and peer. It is built to function well under a variety of circumstances, including erratic network connections, overloaded networks, changing temperatures (regular computer clocks are sensitive to temperature),

and systems that do not run continuously or on virtual machines.

3.4.2 QoS handling support

QoS handling can be done with packet marking or in separate tunnels. The UFTR can be extended with one more feature as it is used in topic tools "mux" [61]. First, the application publishes a topic tagged with `_qos_out`. The UFTR subscribes to it with a general subscriber (see Figure 3.10). Second, several publishers are initiated functioning as the tunnel start-point. The topic messages are encapsulated in a new message header. In this operation mode of each publisher the topics are transmitted via QoS handling, thus the unique network flow endpoint option is set. The published topic tunnels are tagged with the priority level or QCI value "_QCI_8". The UFTR receives from the publisher either via an external control plane topic when to enter certain traffic priority handling modes and the UFTR sends the incoming topic messages to the specific outgoing topic with the unique flow id and QCI tagging. The QoS mapper establishes dedicated bearers with NRM requests to the known 5-tuples, thus proper QoS handling is ensured for the QCI tagged topics (see Figure 3.9).

The same UFTR has the second operation mode, in which it subscribes to all the encapsulated topics with the various "_QCI" tags. The header is parsed, and the binary data stream is forwarded through a general publisher. This step is indicated in the topics naming with a `_qos_in` tag that this topic is decapsulated and can be forwarded towards the application.

3.5 Evaluation

As described in Section 3.2.3.1 the use case on which I tested the proposed method is a simple simulator for learning ROS2 is called Turtlesim [69].

To search for the minimal required network resources that can still work well for the user, the MOS/QoS ratio maximizer is a key component (see Section 3.3.3.2 for the concept).

MOS/QoS ratio maximizer. The implementation of the MOS/QoS ratio maximizer is done as a ROS2 node.

Figure 3.11 shows the architecture of the proposed system. The proposed solution is a reinforcement learning based solution. I implemented my solution using Ray 2.2.0 with RLlib [76]. Ray is a general-purpose and universal distributed compute framework to flexibly run any compute-intensive Python workload including distributed training, hyperparameter tuning or deep reinforcement learning. RLlib is an open-source library for reinforcement

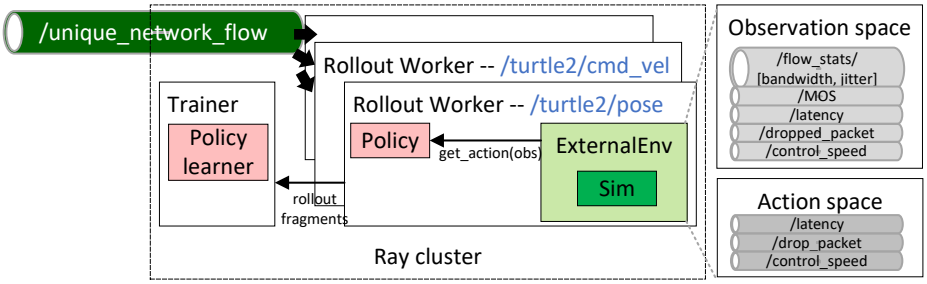


Figure 3.11: The architecture of the MOS/QoS ratio maximizer

learning supporting highly distributed RL workloads. Within RLlib I used a multi-agent environment. The policy graph optimizer used the Proximal Policy Optimization (PPO) algorithm. The output of the system is optimized policy graphs for the ROS2 topic controlling agents.

RLlib supports environments for external agents and applications [77]. It is frequently unnecessary for an environment to be "stepped" by RLlib. For instance, it makes more sense for an agent to query a service that serves policy decisions and for that service to grow via experience over time if a policy is to be employed in a web serving system. This situation also occurs with external simulators that operate freely outside of RLlib's control but may still need to use RLlib for training. The ExternalEnv class from RLlib serves this function. ExternalEnv has its own control thread, in contrast to other environments. Agents on that thread are always able to query the current policy for decisions via `self.get_action()` and report rewards, done-dictionaries, and other information elements via `self.log_returns()`. This is also possible for multiple concurrent episodes.

Every topic that is handled by the UFTR has an associated AI-agent, a rollout worker in RLlib terms. The external *environment* is the ROS2 setup. The *observation space* is collected by subscribing to various ROS2 topics: bandwidth and jitter values of the flow statistics, MOS, latency, dropped packet and control speed.

The *action space* of the agents consists of the increasing or decreasing the latency, dropped packet, control speed with one minor step in the specific value set. The decided action of the agent is published through the above ROS2 topics.

Besides AI, many other types of controllers and heuristics could be used. My aim is to prove the viability of the concept and give a universal solution that can be fine-tuned easily in an application specific way. With this RL-agent it is as simple as providing the application-specific MOS-scores and fine-tuning the weights in the reward function.

In the observation space, the *reward* is the function that enforces the learning algorithm to optimize onto it. The reward should contain information elements on those parts that can

be influenced from the action space, everything else should go into the observation space. The weights on the reward elements influence the learning rate speed. If I set up the weights biased, then the learning rate could slow down so much that I do not see the convergence. Thus, the learning rate influences the success of learning in the end. The reward function for each topic is the following:

$$reward_{topic} = w_1 * mos + latency_{topic} - drop_{topic} \tag{3.1}$$

, where

- w_1 is selected to be 200 in my measurements,
- mos is score is defined as the square error between the desired position and current position of the turtle scaled with the process speed,
- $latency_{topic}$ is considered as the number of times the "UFTR with Throttle" delays the packet. Its effect depends on the topic update rate.
- $drop_{topic}$ means that "UFTR with Throttle" drops every n^{th} packet of the topic. A higher value results in less drop.

Note that the reward function does not contain the VAL process speed. It is included in the MOS score provided by the custom turtle controller.

UFTR with Throttle. An option to collect the network statistics in userland code if the relay node itself adds the network effects to the topic. I aimed at extending UFTR to provide two new features: 1) introduce drop in the topic as it is done in topic tools "throttle" and "drop" [61] and 2) introduce network delay. With these features the relay node is aware of the introduced drop and delay, thus it is possible to know the degradation of the network statistics compared to the existing connection features. Obviously, this approach cannot enhance the network connection by any means.

Figure 3.12 shows the architecture and information flow diagram of the UFTR with Throttle function. The throttle function has an external interface to control the drop rate and delay length. Note that as much as the UFTR is a sensor to collect network statistics, it also becomes a controller with the introduced throttle mechanism. This capability is exploited by the AI-agent to influence the traffic characteristics of the topic.

Learning phase. The PPO is configured in the following way. I set the `no_done_at_end` to true to keep the same simulation environment running and do not reset or exit. This

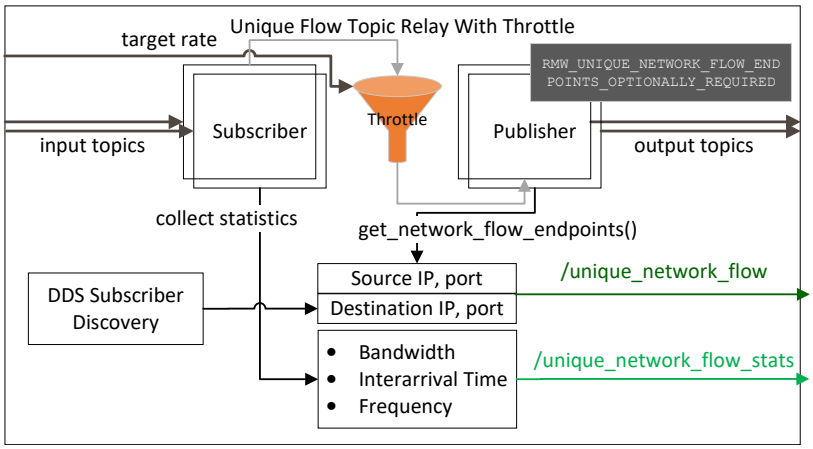


Figure 3.12: Architecture and information flow diagram of the Unique Flow Topic Relay with Throttle function

induced the `soft_horizon` setting to true as well while the horizon size is 128. Detailed description of these parameters can be found in [78].

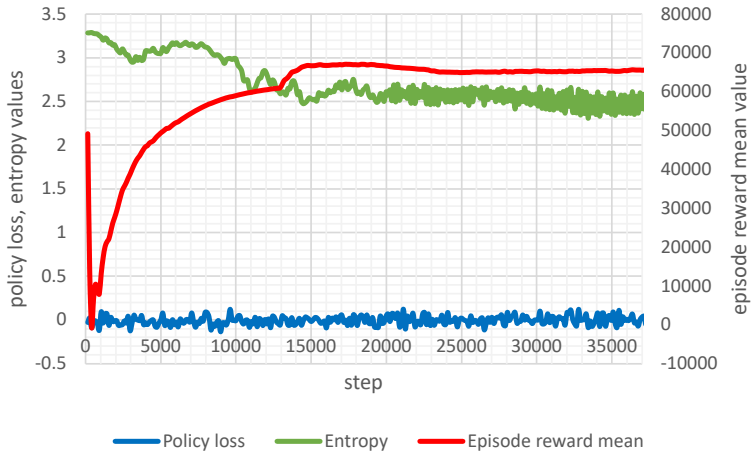


Figure 3.13: The training process

Figure 3.13 shows the training process. The episode reward mean shows the achieved average reward of the policy across all the agents. In the first few hundred steps as the agents start to explore it drops significantly but soon performs better. At about 15k steps it achieves its maximum value during the training and at 20k steps the entropy shows that no further information can be learnt from the available action space. The policy loss function correlates to how much the policy (process for deciding actions) is changing. These values oscillate during training, but they usually fall below 1.0. The entropy shows how random the decisions of the model are. It should slowly decrease during a successful training process.

The values in the figure show successful learning. Note that though each topic has a different rollout worker, the policy for all the agents is common. Therefore, it is the performance of the common policy that can be tracked on Figure 3.13.

Inference phase. Figure 3.14 shows the final setup for each topic that the agents decided to apply after learning. Note that the two controllers of the turtles ended up in different strategies.

	bandwidth	jitter	MOS	latency	dropped packet	control speed
/turtle1/pose	1484	0.0163	4.9405	0	248	
/turtle1/cmd_vel	525	0.0999	4.9405	1	254	0.5
/turtle2/pose	1500	0.0161	4.6904	4	252	
/turtle2/cmd_vel	131	0.4	4.6904	0	184	2

Figure 3.14: The final setup for each topic after learning

Turtle 1 halved its original speed (default speed value is represented with 1, the control speed values should be considered relative to this value) and decided to take only minor packet loss and minor latency achieving higher MOS score than Turtle 2 which took bigger packet loss and latency but doubled the speed of the turtle. These two turtles represent different use cases.

3.5.1 Number of Lines of Code (LOC) for configuration

In Section 3.2.3 I discussed what are the required information elements and how to collect them. The estimated LOC per SEAL function can be seen in Figure 3.15 with the corresponding LOC ratio. TS 23.434 [11] §14.3.2.13 defines the "end-to-end QoS management request"

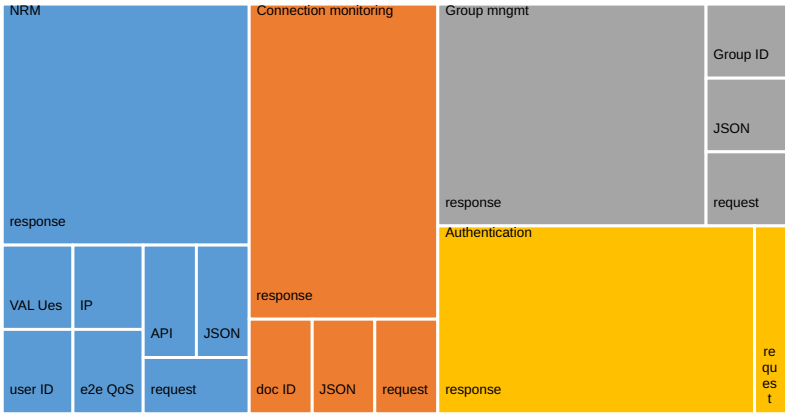


Figure 3.15: Estimated Lines of Code for the SEAL functions

from the NRM client to the NRM server. The request contains mandatory and optional information elements. Though to create a valid API call, only the mandatory elements are required, but as the end-to-end QoS requirement field defines the requested QoS and it is optional, a meaningful NRM request needs an optional field as well. The following 4 information elements need to be set: list of VAL UEs, user ID, IP address, end-to-end QoS requirements. I can assume that each information element requires at least one LOC to collect. An HyperText Transfer Protocol (HTTP) handling library needs to be imported to make the API call. The POST messages need to be set up in proper JavaScript Object Notation (JSON) format. Also, the responses need to be parsed or event-based callbacks need to be implemented.

If connection monitoring and group management are important for a certain use case, they require at least one API call per function towards the SEAL server whose response needs to be parsed. Usually, authentication is also required towards these servers, which is at least one more API call.

The proposed method's QoS mapper node executes the mapping results via an abstract class which makes it possible to connect various API's by implementing various API calls for the same request, thus making SEAL API calls exchangeable for e.g., NEF APIs calls (TS 23.501 [9], 23.502 [10] and 29.522 [23]) easily. The SEAL API handler for the above functions is implemented in approx. 200 LOC. The proposed method provides the described functionalities with a few extra characters onto the existing code at the topic names, or a few lines to remap the topic names in a launch file or command line.

3.5.2 Execution speed of the network exposure requests

To evaluate the execution speed of the network exposure requests I need to discuss how the bearer management occurs based on TS 23.501 [9] and TS 23.434 [11].

Initially, a default bearer with a certain QCI, e.g., QCI 83 is created. The terminal and the base station become connected. When a new NRM request is performed with a different QoS requirement, the network establishes a new dedicated bearer towards the UE. Update of an existing unicast resource is not supported by Rel-17 SEAL, there are create and delete procedures defined only. To overcome this, the UFTR tunnel mode can be used to set up all the possible bearers during the lifetime of the ROS2 application and the tunnel mode emulates the QoS updates by switching between the various tunnels.

Figure 3.16 shows the comparison of the cumulative execution time of the operational phases when the basic UFTR mode and when UFTR with tunnel mode is applied. I performed the measurements in a private 5G Non-Standalone (NSA) network without any background

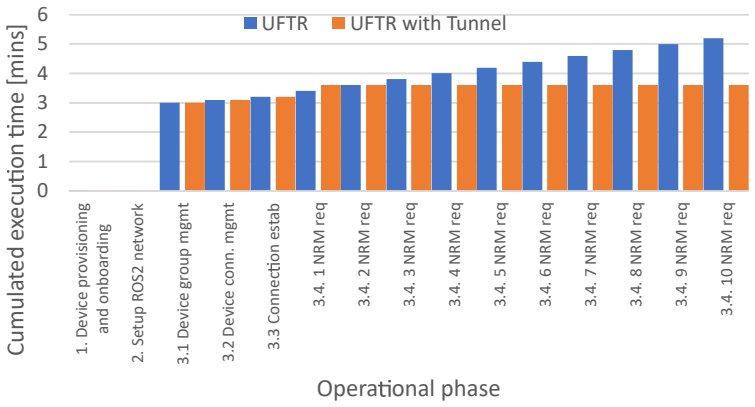


Figure 3.16: The cumulative execution time of the operational phases

traffic or load on the system apart from the tested UE. The NRM update is performed by a unicast resource deletion followed by a unicast resource creation procedure. Figure 3.16 shows that in the initial setup phase the two modes require the same time to execute a NRM update, but after the UFTR with tunnel creating all the required bearers during the initial setup, it does not perform any further bearer or IP routing reconfiguration. The QoS updates are managed by the UFTR itself with in-app routing in the ROS2 layer. The required time for the NRM switch is sub-second level while the bearer setup requires several seconds. Note that the exact values are terminal and network dependent, the magnitude of duration is the most important factor that I should consider.

Chapter 4

Automated Network Exposure Requests in OPC UA and Cross-Domain Service Discovery

4.1 Introduction

In the previous chapter, I focused on ROS2. To expand its applicability across a broader range of industrial use cases, I now turn my attention to another widely adopted industrial protocol, OPC UA [79]. While both protocols share similarities in their overall operational principles, their underlying architectures and functional specifics diverge significantly, necessitating a distinct design approach for seamless integration and optimized performance. My proposed solution focuses on the publish-subscribe operation mode of OPC UA and involves a custom setup of MQTT brokers extended with information collecting nodes regarding traffic statistics and flow identifiers. I also propose a mapping node that converts OPC UA settings to 3GPP SEAL requests. I cover two SEAL functionalities: the NRM and connection monitoring an provide a solution to map the OPC UA application layer information elements automatically to these requests. The OPC UA application developer's only job is to tag the OPC UA topics that need special 3GPP QoS handling. I propose a solution of integrating non-adaptive applications into the 3GPP NRM domain, including those that do not automatically adjust to the channel conditions. My solution leverages CAPIF and its extensions beyond the 3GPP domain to achieve this. I evaluate my comprehensive proposal using a ROS 1.x DT, showcasing how the proposed components operate in practice. One specific use case I explore is 3D printing, where I investigate the potential for dynamic NRM to optimize

network resources and reduce their overall requirements while maintaining the quality of 3D printed objects. I prove that my proposed solution is feasible and light-weight. A series of demonstration videos showing the proposed methods are available at [80]. The related code base is intended to be open-sourced and made available in [81].

4.2 Vertical applications as consumer of networks exposure

Exposure consumers refer to IIoT applications, which are software entities utilizing services provided by the 5G exposure interfaces. Exposure producers encompass the 5G functions that offer the exposed services to consumers. Similarly as in Section 3.2, I elaborate the components of the closed-loop industrial system is detail in subsequent sections.

4.2.1 Sensor – Connection monitoring

We need to collect information from the OPC UA application to make it feasible to define the network requirements. Figure 3.2 illustrates the network stack layers, highlighting potential sources of information for creating network exposure requests. This figure extends the high-level architecture diagram of 3GPP SA6 [82] integrating the OPC UA application stack. Each box is discussed in the further sections. My analysis starts from the application layer, progressively delving into each layer of the protocol stack.

Network requisites emerge from the application-level settings, as determined by the developer, and the inherent channel characteristics. While the former can be directly configured and queried at both the publisher and the subscriber ends, the latter requires measurements across various protocol stack layers.

I want to highlight that the implementation of standards often extends beyond their explicit specifications, addressing aspects that may be underspecified. This is why I discuss implementation details for all network layers, as these details are specific to the implementation and are significant. It is important to note that different implementations may exhibit variations in behavior for certain parts. Apart from these discussions, I avoid repetition of standard content and cite it whenever applicable.

4.2.1.1 Network requirements set by the application developer

Bandwidth and packet IAT within the application are shaped by factors like OPC UA topic message type, size, publishing rate, and sampling interval. These parameters are defined by the OPC UA application developer, influencing the behavior of both the publisher and subscriber components. The OPC 10000-4 UA Part 4: Services section discusses the operation

of the publisher and subscriber. In §5.12.1 [83] the Monitored Item model is described in detail. MonitoredItems are characterized by four key parameters that provide instructions to the Server regarding the sampling, assessment, and reporting of the item. These parameters encompass the sampling interval, monitoring mode, filter, and queue parameter. Beside the regular rate parameters, more advanced features like Data Change Filter (OPC 10000-4 UA Part 4: Services 1.04) §7.17.2 [84] influence data publishing rate. The developer can specify a filter criterion to determine the minimum change in value required for an update to be published.

OPC UA PubSub communication pattern can be supported with various protocols. I chose MQTT for my experiments as the open-source implementations of the OPC UA standard e.g., [85] are usually built upon MQTT. It is important to note that by using MQTT as a transport for OPC UA, the filtering mechanisms of MQTT, such as topic-based subscriptions and wildcard characters can be utilized to selectively subscribe to specific OPC UA topics. However, OPC UA-specific features like Data Change Filter would not be available directly through the MQTT protocol. To apply data filtering in an MQTT-based OPC UA scenario, the application developer needs to implement the filtering logic within the MQTT client or application code.

OPC UA QoS settings. OPC UA provides QoS mechanisms to ensure reliable and secure communication between devices and systems. The combination of the optional QosCategory and the PriorityLabel specifies the priority according to the sender. The network stack uses the PriorityLabel to look up the priority settings for the transport protocol headers. Note that the cited version of OPC does not define specific labels. The engineering process needs to provide them and also build up the PriorityMappingTable in OPC 10000-22 [86] accordingly.

How does QoS settings map from OPC UA to MQTT? In MQTT broker mode the QoS settings are used to control the level of reliability and delivery guarantees for published messages. The empty QoS settings are implemented in a broker specific way in OPC UA. As an example, [87] provides an implementation for mapping the QoS settings of OPC UA to MQTT. Practically, the QoS settings that exist in MQTT are defined as QoS settings in OPC UA.

The following list contains the method how the OPC UA QoS settings map to MQTT QoS levels in broker mode (see [20] §3.3.1.2 for more details):

- At most once, QoS 0 in MQTT: In this mode, messages are delivered once or not at

all, without any confirmation or retransmission. This mode is suitable for non-critical data that can be lost or duplicated without causing significant problems.

- At least once, QoS 1 in MQTT: In this mode, messages are delivered at least once to the subscriber, and the broker ensures that duplicates are eliminated. The subscriber sends an acknowledgment to the broker to confirm receipt, and the broker retransmits the message until the acknowledgment is received.
- Exactly once, QoS 2 in MQTT: In this mode, messages are delivered exactly once to the subscriber, without loss or duplication, and in the correct order. The publisher and broker use a handshake protocol to ensure that each message is received exactly once, and the subscriber sends an acknowledgment to confirm receipt.

Note that in the 'at most once' mode of MQTT, the data is sent without waiting for any acknowledgment from the receiver, so there is a possibility that the message could be lost or dropped if there are network issues or if the receiving end is not ready to receive the data. Even if TCP is used as the transport protocol, which provides reliable and ordered delivery of data, it cannot guarantee that a message sent in OPC UA 'Best Effort' mode will be received successfully by the receiver. These cases can occur especially in case of using the broker mode, not in the point-to-point TCP communication.

4.2.1.2 Measured connection related KPIs

Essential information in terms of network statistics and the QoS of the connection are latency, jitter, and packet loss. While in the previous section the network flow statistics can be collected on the publisher side to estimate the bandwidth of the application, the influence of the subscriber's setting and the transmission channel needs to be measured.

Figure 4.1 shows the possible approaches to gather network statistics in various layers of the system with grey boxes.

OPC UA client library API. OPC UA provides a feature called DiagnosticInfo [88], which provides details on the performance of the OPC UA communication. This includes information on the server, session, and subscription diagnostic information such as PublishingIntervalCount, TransferRequestCount, SecurityRejectedSessionCount and many other information elements [89]. Note that the certain nodes that provide the diagnostic information rely mostly on the underlying network layer if they populate the necessary information. It can be the case that a certain network configuration leaves a lot of these diagnostic information nodes empty.

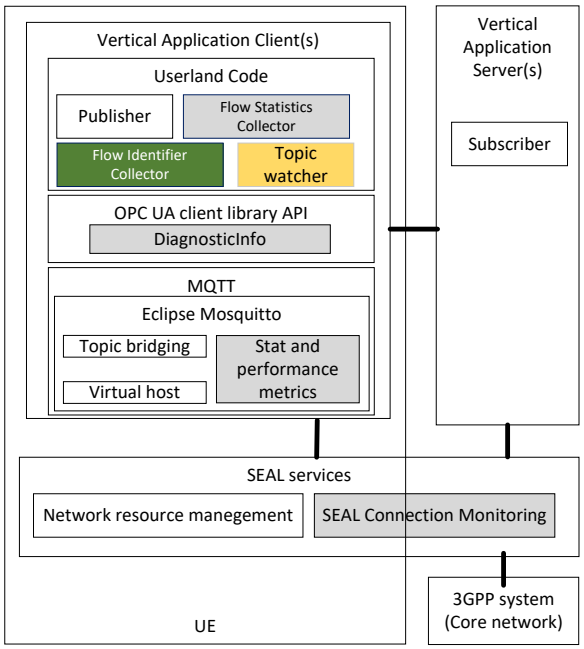


Figure 4.1: Protocol stack /grey boxes show the network statistics modules/

The benefit of this approach is that we are in the user space application, and this method measures the same QoS KPIs as the application would perceive. The main issue with this approach is that it is not a transparent solution. It requires the user to modify its code by enabling the subscription feature and forcing them to use time stamped messages.

I attempted various designs to incorporate this feature into the "Flow Statistics Collector" (for more information, see Section 4.2.2.3). My experimentation revealed that none of the diagnostic nodes offer more insight than the underlying MQTT layer itself. Additionally, there is a possibility of additional aggregation at this level for diagnostic counters, resulting in the loss of finer details. Taking everything into consideration, it appears that this solution may not be a feasible option to include into Flow Statistics Collector.

MQTT. To get statistics on the transferred data from the MQTT layer, I can use the built-in mechanisms provided by the MQTT broker. Most MQTT brokers provide several ways to collect statistics and performance metrics.

The capabilities of various MQTT implementations differ significantly, thus whenever implementation specific details are required, I provide information that is available in Mosquitto [90]. Mosquitto have built-in statistics and performance metrics that can be accessed by subscribing to topics in the $SYS hierarchy. Topics labeled as "static" are transmitted only once per client upon subscription. Conversely, all remaining topics receive

updates at intervals of sys_interval seconds. Two useful topics as examples are the following:

- `$SYS/broker/load/bytes/received/+`: The moving average of the number of bytes received by the broker over different time intervals (1 min, 5 min or 15 min).
- `$SYS/broker/load/publish/dropped/+`: The moving average of the number of publish messages dropped by the broker over different time intervals. This illustrates the message loss rate experienced by clients that become disconnected.

It is important to note that while MQTT brokers offer comprehensive statistics within the $SYS topic, these statistics lack per-topic granularity.

Another way to collect statistics on MQTT communication is to use packet sniffing tools in the broker to capture and analyze the network traffic. This is supported by the plugin system of Mosquitto [91]. A plugin that catches the passing packets on the broker can have detailed information on the communication patterns, including the packet size, data rate, and latency based on accurate timestamping. The advantage of this approach is its detailed KPIs and accuracy. The drawback is the number of steps required to enable this module. The plugin is MQTT broker specific, thus various brokers need their own vendor specific implementation.

A third option to collect statistics is by implementing a custom subscriber that collects the necessary information as [92] does. I went with this option as this is the most transparent deployment-wise. The only drawback is the computational overhead (see Section 3.5 in detail).

For the transport network information see Section 3.2.1.2 paragraph "Transport network".

4.2.2 Controller – Network Resource Management

The sensor information collected from Section 4.2.1 needs to be mapped to a control process. I will explore the different layers of the network stack and discuss how the NRM can be applied at each layer.

4.2.2.1 OPC UA rate limit

When considering the OPC UA layer, it is important to note that this layer does not offer rate control mechanisms on its own. The implementation at the application layer should include rate control as outlined in [93].

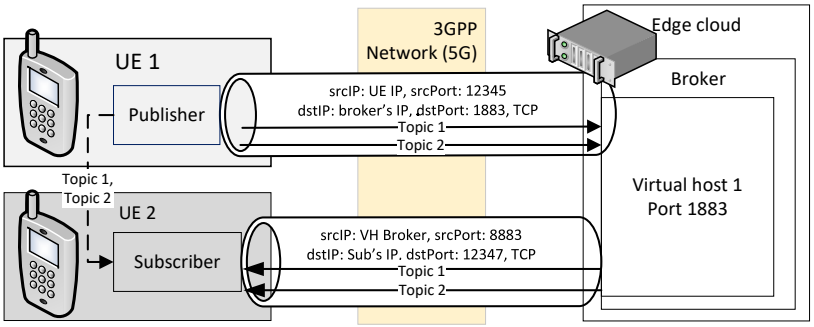


Figure 4.2: The basic architecture for a simple OPC UA publisher and subscriber communicating over 5G

4.2.2.2 3GPP SEAL

Section 3.2.3 collects the components of TS 23.434 [11] §14.3.2.13, the "end-to-end QoS management request". Note that the Rel. 17 specifications facilitate UE-level configuration of QoS parameters. In order to ensure compatibility with forthcoming releases and to be in alignment with recommendations in [26] §4.2.3, specifically Notes 2 and 3, to support multiple IP flows, I introduce TCP/IP-flow level identification for OPC UA topics.

4.2.2.3 The proposed method to retrieve the 5-tuple identifier of the MQTT topic

Figure 4.2 shows the basic architecture for a simple OPC UA publisher and subscriber communicating over 5G. Note that while the figure shows the case when both the publisher and the subscriber are connecting via 5G towards the edge cloud where the broker is hosted, other setups are also possible e.g., the subscriber is in the edge cloud. The proposed approach maintains its applicability across these diverse scenarios, requiring adaptations to the architecture to ensure effective processing of topics transmitted over the radio channel. Note that the proposed method fully adheres to the 3GPP, OPC UA and MQTT standards as there are no modifications made to any of those, rather, they are utilized within my proposed architecture.

Figure 4.3 shows my proposed architecture for automatic handling of OPC UA QoS to SEAL mapping. In this section I discuss the architecture elements one-by-one.

To fill in the IP address field of the SEAL NRM API, or a more detailed 5-tuple identifier (transport protocol, source IP, source port, destination IP, destination port) of the IP flow on which the QoS management request is valid, I need to obtain the MQTT topic flow identifiers. The MQTT protocol uses a publish-subscribe model, where publishers send messages to

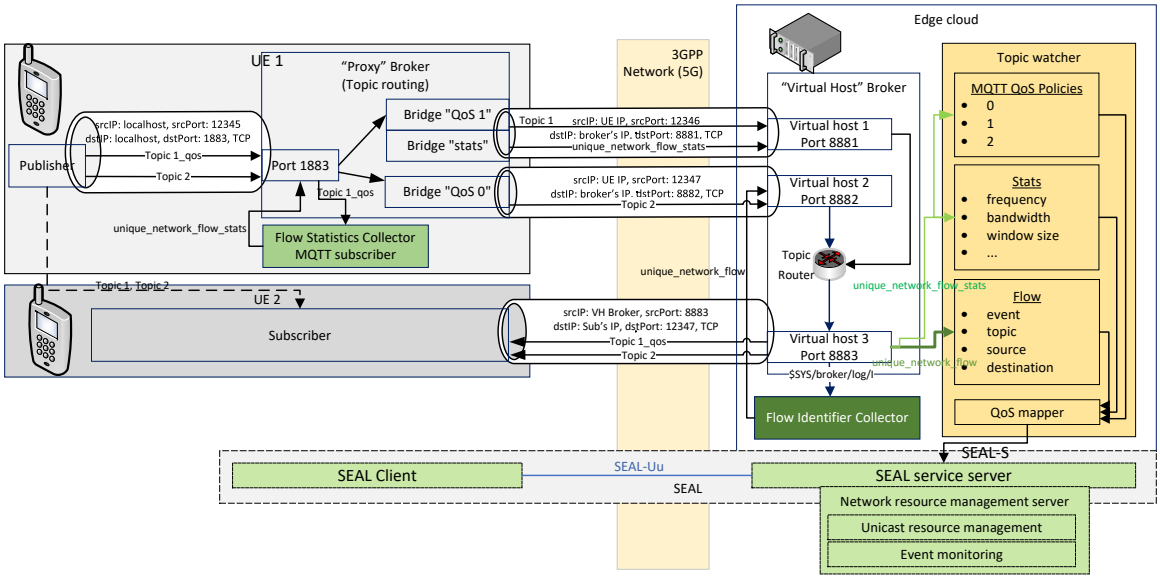


Figure 4.3: The proposed architecture for automatic handling of OPC UA QoS to SEAL mapping

specific topics, and subscribers receive messages based on their subscriptions to topics. The MQTT broker manages the routing of messages between publishers and subscribers within the established TCP/IP connection. MQTT operates over a single TCP/IP connection between the MQTT client and the MQTT broker (server) [20]. This TCP/IP connection is established when the client connects to the broker, and it remains open as long as the client is connected. Within this single TCP/IP connection, multiple MQTT topics can be subscribed to, and multiple messages can be published and received. MQTT messages are encapsulated within TCP packets and transmitted over the established connection. The MQTT broker handles the routing of messages to the subscribed clients based on the topics they have expressed interest in.

This operation mode prevents communication nodes from explicitly differentiating network QoS (not the MQTT QoS) for publishers and subscriptions on IP flow level. Thus, only the same network QoS can be assigned to publishers and subscriptions in communicating nodes.

Proxy Broker. The MQTT standard [20] is known for its lightweight nature, with most of the advanced functionalities being determined by individual implementations. An important requirement I seek from the MQTT API resembles the functionality introduced in ROS2 Galactic, termed "Support for unique network flows" [65]. This becomes particularly relevant in scenarios where QoS specifications for IP streams are necessary, especially in network

setups like 5G networks. Modern applications might necessitate the capability for specific topics to initiate new IP flows, distinct from the existing ones, rather than being combined within them. It is worth noting that these essential features are currently absent in MQTT implementations.

An alternative approach is to incorporate this capability into their functionality. To achieve the desired outcome, I utilized a feature known as "topic routing". In my setup, the publisher of the UE establishes a connection with a local MQTT proxy broker, typically accessible through the default Internet Assigned Numbers Authority (IANA) port 1883 designated for MQTT [20]. The proxy broker executes topic routing using a predefined naming convention. Specifically, topics requiring QoS management are identified by appending "_qos" to their names.

The regular topic that does not require special QoS handling is going via "Bridge QoS 0" to "Virtual host 2". Also, the "unique_network_flow_stats" topic that provides the network statistics also uses the same default bridge without the special QoS handling.

Flow statistics collector. The "Flow statistics collector" establishes a connection with the "Proxy Broker" on the UE side by subscribing to the "_qos" tagged topic. The subscriber receives all the traffic from the source before transmitted over the wireless link. It collects the packet IAT and packet sizes and it is capable of calculating basic network traffic statistics per topic basis. The flow statistics are encoded into JSON format and transmitted to the "Virtual Host broker" via the "Bridge Stats".

Virtual Host Broker. The broker in the edge cloud is extended with the "Virtual Host" capabilities. This feature enables the broker instance to listen on multiple ports. Note that every subscriber has the capability to connect to any port and receive the data that has been published on any of those ports.

The idea here is to create as many virtual hosts as many QCIs I intend to handle in the system. The QCI handling is defined in 3GPP TS 23.203 [70]. For each QCI class a separate virtual host is defined. As Figure 4.3 shows, the bridges from the proxy broker and the virtual host brokers define unique flows for topics that need special QoS handling on the wireless channel. These IP flows have no multiplexed topics and the 5-tuple identifier of the flow becomes unique for each MQTT topic.

Flow Identifier Collector. The "Flow Identifier Collector" subscribes to a special topic published by the "Virtual Host Broker" in the edge cloud. The broker is aware of the 5-tuple flow identifiers of all the publishers and subscribers. It is up to the broker implementation

where this information can be extracted. The collected flow identifiers are published back to the "Virtual host 3" on the "unique_network_flow" topic.

OPC UA publisher. The applied the OPC UA implementation in my experiments is based on [85]. I extended the code of the OPC UA MQTT PubSub example [94]. The main modifications is to add a second topic beside the existing one and set the broker address according to my proposed architecture. The publisher connects to the proxy broker, which is the local host usually. The OPC UA QoS can be set in the OPC UA code, but any further QoS settings i.e., the handling of the bridges with 3GPP QoS needs to be set up in the configuration of the proxy broker.

OPC UA subscriber. On the subscriber side I added a new data reader for the "topic1_qos". Also, I set up the broker address to be able to connect to the virtual host according to my proposed architecture.

4.2.2.4 The method to define QoS out of the connection monitoring data

Until this point I managed to identify the flow identifier of the topic which needs QoS handling. In the following steps I discuss how the QoS handling is achieved.

3GPP TS 23.203 [70] Table 6.1.7 enumerates the standardized QCI characteristics (see Section 3.2.3.2). The table provides the association of a QCI with specific input parameter sets. This mapping process is denoted as the "QoS mapper" in Figure 4.3. The method closely follows the approach described in Section 3.2.3.2; therefore, it is omitted here for brevity. For a detailed discussion of the differences, please refer to Section 3.2.4 of [30].

4.3 Proposed architecture to trigger NRM requests in VAL

3GPP TS 23.434 [11] provides NRM functionality which is implemented on the radio channel. While the NRM affects the channel, the traffic of the VAL application transferred over it is not altered in any way. If the 3GPP NRM request fails to trigger adaptive changes in the VAL traffic and the VAL traffic itself does not possess self-adaptive capabilities, there is a risk of experiencing considerable performance degradation in the VAL traffic.

In my proposal, I connect the NRM management with the execution of the NRM on the VAL and fit the VAL traffic into the requested channel. The proposed solution can be executed in the UE's VAL layer, in the Operating System (OS), or on the VAL server and the transport layer of the 5G/6G network as well. The proposed method adds new elements

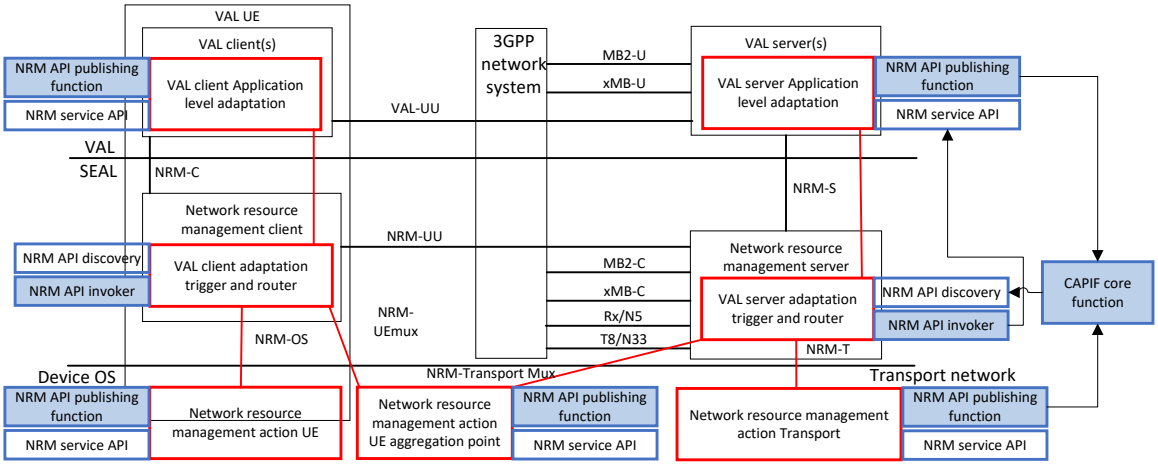


Figure 4.4: Extension of on-network functional model for NRM (extension of Figure 14.2.2.1-1 in [11])

and interfaces to the functional model of the 3GPP TS 23.434 [11]. The proposed method extends the existing NRM request API with new information elements.

4.3.1 SEAL deployment

In this section I discuss how the application agnostic solution looks like from the SEAL perspective.

4.3.1.1 On-network functional model

Figure 4.4 (which is the extension of Figure 14.2.2-1 in [11] with the red boxes) illustrates the generic on-network functional model for NRM extended with the following functional entities:

- VAL client Application-level adaptation: This functional entity is responsible to execute the NRM action in the VAL domain's client side. E.g., the Open Network Video Interface Forum (ONVIF) [95] API
- VAL server Application-level adaptation: This functional entity is responsible to execute the NRM action in the VAL domain's server side. E.g., the DASH [96]
- NRM action UE: The NRM on the UE, but below the VAL layer can be performed in the driver, OS, or on the network interface of the UE.
- NRM action Transport: The NRM on the server side can be hosted on the transport network or on the interface between the core and radio. One example is to use Software

Defined Networking (SDN) for QoS in the mobile networks like in [97].

- NRM action UE aggregation point: This function is responsible to execute the NRM action outside of the 3GPP, the VAL and UE domains. The aggregation point collects the traffic of a set of UEs connected via radio to the 3GPP network system. As an example see [98] in which in-network computing devices can be utilized within the network to perform application-level packet processing on the fly on the aggregated user traffic.
- VAL client adaptation trigger and router: This functional entity is responsible to trigger and route of a 3GPP NRM request to other layers on the client side.
- VAL server adaptation trigger and router: This functional entity is responsible to trigger and route of a 3GPP NRM request to other layers on the server side.

The following reference points are added to the standard defined list:

- NRM-OS: Interface between the NRM client and the NRM action UE.
- NRM-T: Interface between the NRM client and the NRM action Transport node.
- NRM-UE Mux: Interface between the NRM client and the NRM action UE aggregation point.
- NRM-Transport Mux: Interface between the NRM client and the NRM action UE aggregation point.

The interface between the "VAL client adaptation trigger and router" and the "VAL client Application-level adaptation" can utilize the existing NRM-C interface and similarly the server side can utilize the NRM-S interface.

4.3.1.2 Information flows

In this section I provide the extensions of the information flows compared to the standard.

Network resource adaptation action request. 3GPP TS 23.434 [11] provides the description of the information flows between the functional entities. My proposal results in the extension of the information flow tables. The extension of Table 14.3.2.1-1 in [11] describes the information flow network resource adaptation request from the VAL server to the NRM server. The added information elements are highlighted with italic text. The extension should include a fully qualified identifier of the communication channel of the VAL

application on a certain network layer i.e., it requires topic name for the OPC UA layer, or IP flow identifier in the OS layer. The resource adaptation requirement field should contain an application specific QoS related information e.g., control frequency of an actuator, or video resolution for video. Currently, it is defined as a string in [12], thus that field can be utilized without any modification on the standard (see Table 4.1).

Information element	Status	Description
Requester Identity	M	The identity of the VAL server performing the request.
List of VAL UE IDs	O	List consisting of one or more VAL UE IDs for whom the network resource adaptation occurs.
VAL group ID	O	The VAL group ID for whom the network resource adaptation occurs.
Resource identifier	M	The VAL application communication channel identifier
Resource adaptation requirement	M	The resource adaptation requirement corresponds to the VAL service QoS requirements as applied for a UE or group of UEs (E.g., bandwidth, resource, *control frequency, FPS, resolution*).

Table 4.1: Client-side Network resource adaptation action request (Extension of Table 14.3.2.1-1 [11]) ; The extensions are highlighted with italic fonts; Status can be Optional (O) or Mandatory (M)

Client-side Network resource adaptation action response. The extension of Table 14.3.2.2-1 in [11] describes the information flow of the network resource adaptation response from the NRM server to the VAL server. The result can provide information on the failure regarding the capability limitation of a certain network element. E.g., in case the "NRM action UE aggregation point" cannot handle the NRM request, then the "VAL client adaptation trigger and router" can route the request to the "NRM action UE".

NRM-UE Mux request. The information elements on this interface can have less information than in "Network resource adaptation action request" as we skip out of the 3GPP defined VAL domain. The action requires the resource identifiers and the resource adaptation requirement, and there is no need for the 3GPP-related information elements. The format of the message, and the transport protocol is preferred to be in byte format e.g., protobuf over UDP than e.g., verbose JSON format, as the network nodes have limited string parsing capabilities.

4.3.1.3 VAL client or server adaptation trigger and router

The NRM client or NRM server should choose a route in which the NRM request is forwarded. According to the state-of-the-art, the manually configured option with direct request is a

viable way. I argue that it can be automatized with the help of 3GPP capabilities. Based on the capability of the VAL client or the aggregation point, the NRM client or server can forward the request to the designated component. The method of choosing the routing is the following priority list (with decreasing priority).

The capabilities of the devices are collected. The "VAL client adaptation trigger and router" or "VAL server adaptation trigger and router" checks whether the "VAL client Application-level adaptation", the "VAL server Application-level adaptation", "NRM action UE", "NRM action Transport" or the "NRM action UE aggregation point" function exist in the deployment.

The capabilities of the functional entities should be collected in a standardized way that supports service discovery across the VAL and 3GPP domains. My proposal is to utilize the 3GPP TS 23.222 [99] "Common API Framework for 3GPP Northbound APIs" standard. The proposed solution in relation to service discovery is discussed in Section 4.3.2 in detail.

Handle the request. If there is a capable functional entity, then the preferred operation is to handle the request in it.

Error handling. In case of failure the “NRM client” or "NRM server" should forward the request to another NRM capable functional entity.

The selection of the NRM handling functional entity can be based on various KPIs. Practical KPIs consider the QoE of the VAL application, radio-time, or power consumption of the devices. If there is any other KPI altering this priority, it can be used as well.

4.3.2 CAPIF

In this section I discuss how the cross domain NRM service discovery can be achieved with a solution in-line with 3GPP. The specification of the CAPIF is covered in 3GPP TS 23.222 [99] "Common API Framework (CAPIF) for 3GPP Northbound APIs" since Rel. 15. The definition of CAPIF APIs essential for implementing CAPIF functionality is encompassed within 3GPP TS 29.222 [100].

4.3.3 Example sequence diagrams on the procedures

Figure 4.5 shows the sequence diagram of an example run of the CAPIF services, how an API provider publishes its services through CAPIF, and how an invoker discovers and invokes those services. In the NRM API provider publishing phase, the "VAL client Application-level

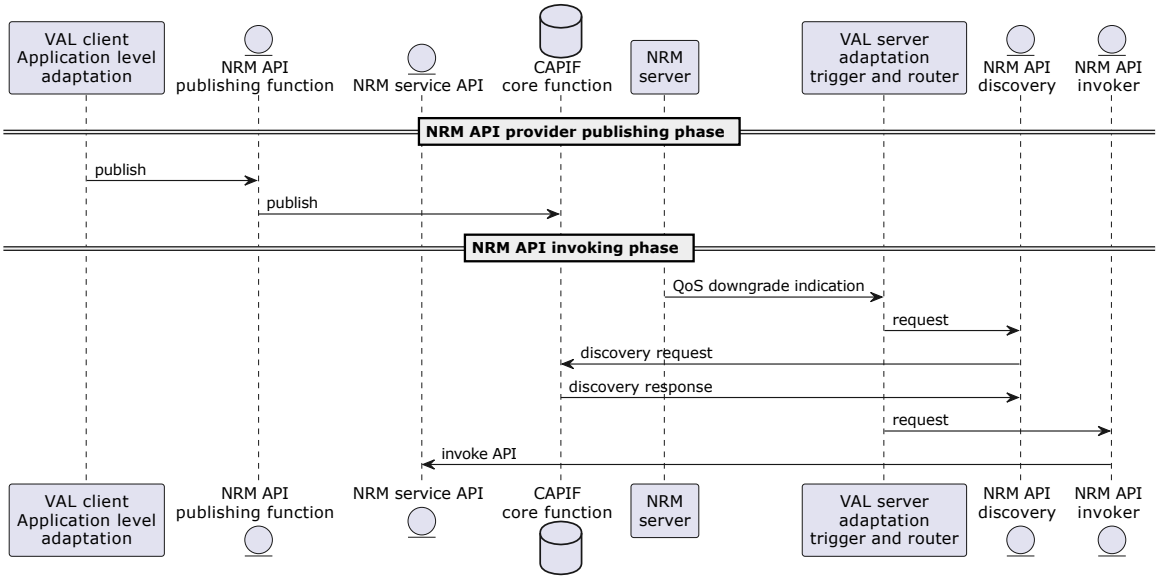


Figure 4.5: The sequence diagram of an example run of the CAPIF services

adaptation" or other functional entities (my extensions with the red nodes from Figure 4.4) calls their NRM API publishing function to register their services with CAPIF. They provide detailed information about their service, such as endpoint Uniform Resource Locators (URLs), supported methods, authentication requirements, and any other relevant metadata. CAPIF stores this service information in the API registry, making it accessible to potential invokers.

In the NRM API invoking phase when the SEAL NRM server sends an "Application QoS change notification" type of message defined in 3GPP TS 23.434 [11] §14.3.2.16 to the NRM client and the "VAL server adaptation trigger and router" (this is an extension of the procedure of TS 23.434). The "VAL server adaptation trigger and router" looks for suitable APIs to fulfill the NRM downgrade call. It is the NRM API discovery process in which it queries the API registry through CAPIF to discover available services. This query can be based on various criteria, like NRM service provided in the highest possible network layer or reaching out for the highest number of devices and searching for an aggregation point. The "VAL server adaptation trigger and router" starts the NRM API invoker process and invokes the NRM service API that best matches its requirements. CAPIF is responsible for providing the invoker with the necessary authentication details from the API registry.

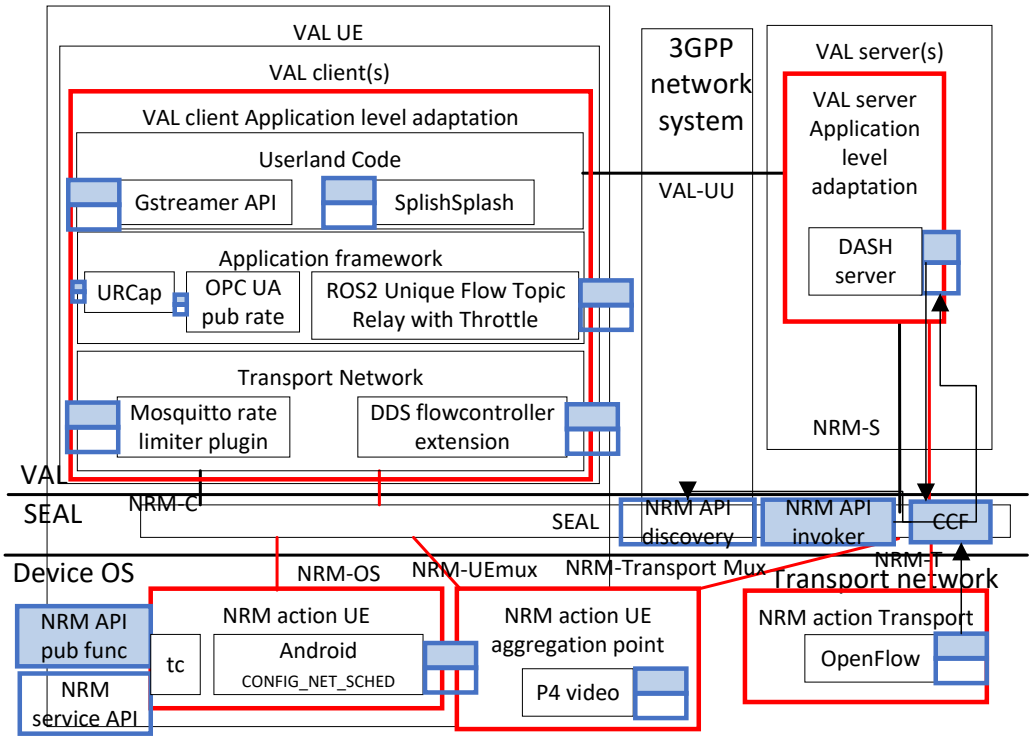


Figure 4.6: Examples of NRM in various network layers

4.3.4 Example applications

The important part of the NRM is to reduce the unnecessary traffic on the radio channel. This means that till the very last point when the packet is handed over to the radio interface, it is worth to reduce the traffic if possible. In the following section I show examples on various network layers that can implement the NRM service. Some of them are collected as a candidate, and for some of them I provide open-sourced code as well. Figure 4.6 illustrates the location of the examples within the network layer.

4.3.4.1 Application framework

This layer is universally applicable to all applications within a specific framework, such as OPC UA or ROS2 applications. One significant advantage is its inherent operating system independence. Any API call accessible within this layer is universally available to all applications utilizing the framework. Typically, this encompasses a subset of the APIs offered by the transport network layer, as it requires the interception of the diverse features present in various transport network implementations. Furthermore, within this layer, the content of packets is comprehensively interpreted. This ensures that the application does not

encounter issues with packet fragmentation or decoding if a packet is dropped at this layer.

OPC UA publishing rate. [85] provides an example for real-time pubsub communication. In the example the `pubCallback()` function is for creating event-based server publish methods. I implemented my own `addWriterGroup` extension to expose the OPC UA publishing rate of the topic. The original periodic publishing rate is emulated by calling pubCallback periodically from an event loop. The periodicity is exposed on the CAPIF API publishing function. The NRM service API can receive the periodicity ratio of the default operation and reconfigure the periodicity of the event loop to the received value. I provide the extension of the [94] with the described CAPIF functionality code in [81].

For further examples see [30]. The detailed evaluation is in Section 5.3.

Chapter 5

Feedback Mechanisms for Observability in Industrial Processes

5.1 Pick and Place use case

Network management aims to ensure satisfactory QoE for users by appropriately configuring QoS parameters. However, achieving this goal involves a balance between extensive over-provisioning, which incurs significant costs, and meeting users' perceived quality expectations.

In MBB, applications are developed in collaboration with the telecommunications community. KPIs such as web page load times and video startup times are influenced by the design of browsers, protocols, and data compression algorithms tailored for radio communication. There is a recognized relationship between the QoS delivered by a wireless link and the anticipated QoE for user applications.

Adaptability is paramount in Industry 4.0, and 5G technology offers precisely that. Widely regarded as pivotal for future factory infrastructure, 5G is valued for its ability to support real-time communication and high reliability. This chapter focuses on leveraging the low-latency capabilities of 5G for remote operation of industrial robot gantries. While MBB has historically defined MOS for voice or video communication and streaming, meeting the stringent communication demands of manufacturing processes typically revolves around binary outcomes: acceptable or unacceptable final product quality. Establishing a translation from QoS to QoE is challenging in manufacturing contexts. Industrial devices impose strict network requirements due to legacy and safety considerations, yet translating

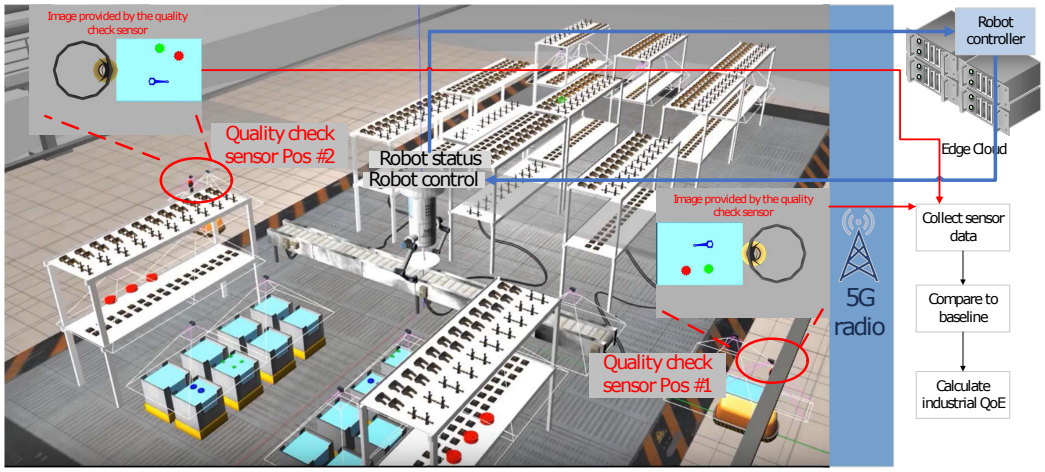


Figure 5.1: The design of the measurement arrangement

these requirements into operational performance indicators for particular scenarios remains challenging.

Establishing this mapping is crucial for enhancing network agility and making it more attuned to manufacturing processes. While current 3GPP SA6 standardization efforts empower manufacturers with necessary information and control, extending similar capabilities to industry verticals could enable feedback on manufacturing process QoE and control opportunities. Currently, network operators lack insights into whether, for instance, a packet drop resulting in a 30 ms delay in car assembly is merely inconvenient or potentially life-threatening. Research related to ARIAC [51] demonstrates that evaluating agile production cells with pick-and-place tasks for on-demand orders can yield non-binary performance metrics, enabling fine-grained scoring to rank competitors effectively.

In this chapter, my aim is to adopt the established approach used in MBB for assessing user-perceived service quality and extend it to the realm of industrial and manufacturing operations. Specifically, I investigate the previously unexplored relationship between network QoS and the quality of industrial pick-and-place scenarios. This methodology holds potential applicability across various industrial domains characterized by precision and speed requirements, such as sanding, painting, etc.

To achieve this goal, I undertake the following steps (also outlined in Figure 5.1): 1) Establish a simulation environment encompassing the remote control of a robot gantry via 5G radio from the edge cloud; 2) Investigate the execution of order fulfillment under various network conditions; 3) Introduce KPIs for assessing pick-and-place quality; 4) Compare the quality of pick-and-place executions under different network configurations; 5) Propose a quantitative method for deriving MOS.

In this chapter, my aim is to demonstrate that industrial procedures can be individually analyzed according to their network performance needs. I propose evaluating their efficiency through rapid expert assessment, bypassing the laborious task of measuring process-specific KPIs at a granular level.

5.1.1 Related work

Pick and place task is a wide range of industrial tasks e.g., stacking, truck loading, palletizing, etc. Each of these processes have their own KPIs ranging from palletizing speed to more complex KPIs defined for successful pick of various sized and shaped objects, to plug them into certain areas. An example of this is surface-mount technology, a technique used to manufacture electronic circuits by directly mounting or placing components onto printed circuit boards' surfaces in [101]. There are certain effects e.g., cleaning periodicity of the nozzles that influence the quality of the surface-mount, such as the number of component defects in the printed circuit boards.

These pick and place scenarios do not belong to the CPPS use cases as the execution is not affected by the transport channel quality. The effect of a remote controlled pick and place scenario in which the robot is controlled with velocity commands streamed from the edge cloud over an imperfect channel and the robot status messages vice versa with various network latency setups is analysed in [13] in both simulation environment and real robot hardware connected with real radio channels. It was proved that the simulation and real world are close to each other for the small latency setups. The above paper used the ARIAC 2017 scenario for evaluation of the pick and place scenarios and the productivity KPI was the ARIAC scoring [51].

5.1.2 Test environment

In the following section I discuss the test environment in details.

5.1.2.1 Baseline simulation environment

My primary objective is to assess the performance of my proposed system within a robotic cell and examine the impact of radio channel conditions on productivity KPIs in a dynamic setting. To accomplish this, I utilize the ARIAC 2020 environment [51] and extend it with custom data collection functionalities to suit my specific requirements.

ARIAC is specifically designed to foster agility in industrial robotic processes. This simulated environment is implemented in Gazebo [102]. Each order specifies the parts to be

placed in shipments and their precise positions within shipping boxes. Fulfilling an order involves collecting the correct product parts from shelves and placing them onto trays carried by Automated Guided Vehicless (AGVs). I adopt the solution presented in [103], the winner of the ARIAC competition in 2020, as my baseline.

5.1.2.2 Simulation accuracy

To evaluate the expected accuracy of my simulation setup in comparison to a fully implemented system, I can consider the following points. I acknowledge that the realism of a simulation is inherently limited when compared to an actual implementation. However, I contend that the simulation elements I employ provide near-real-life accuracy. Gazebo [102], utilized for simulating the pick-and-place process, is widely regarded as a highly accurate simulator. The network latency plugin underwent evaluation in a hardware-in-the-loop setup [13], demonstrating that up to a 20 ms latency, the behaviors observed in both real and simulated environments align, with similar outcomes observed at higher latencies. Gazebo emulates cameras by rendering their perceived images without distortion or visual artifacts, resulting in ideal representations. Although real-world camera images may exhibit imperfections, the proposed method remains valid, albeit with potentially slightly increased Earth Mover's Distance (EMD) values.

5.1.2.3 Incorporating network effects

The network resource management, requested by either the vertical application client or server, results in a specific radio link setup characterized by latency effects. However, Gazebo lacks the capability to simulate control latency effects, which are essential to function as a fully-fledged Cyber Physical System (CPS) simulator. To address this limitation, we have proposed and implemented a Gazebo plugin [13] that enables the measurement of production level KPIs by introducing control and status message delays within Gazebo. It is worth noting that CPS control traffic typically does not impose heavy bandwidth requirements. Other network KPIs such as jitter and packet loss is effectively converted into latency using simple hold and forward mechanisms. Therefore, study on latency effects emerges as an significant task in my study.

5.1.3 Evaluation

My objective is to assess the performance of the setup under various network QoS configurations. An expected outcome of network latency is the potential deviation of part positions

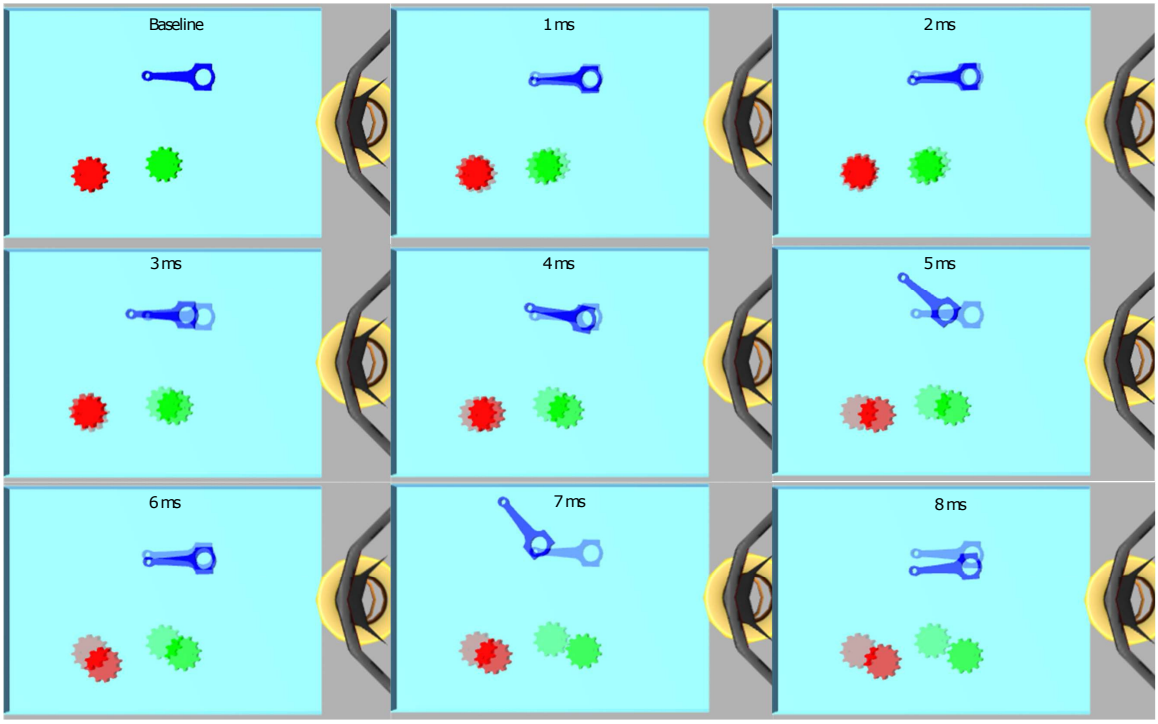


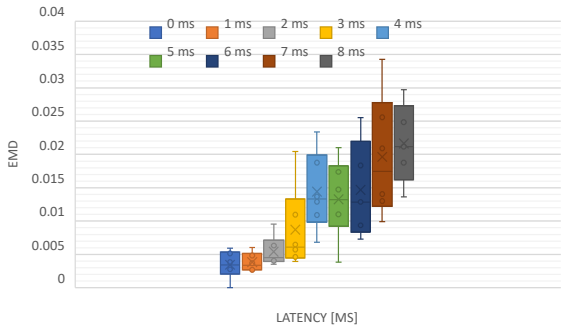
Figure 5.2: The final position of the parts on the first AGV's tray compared to the baseline case (the transparent objects)

on AGV trays from the baseline.

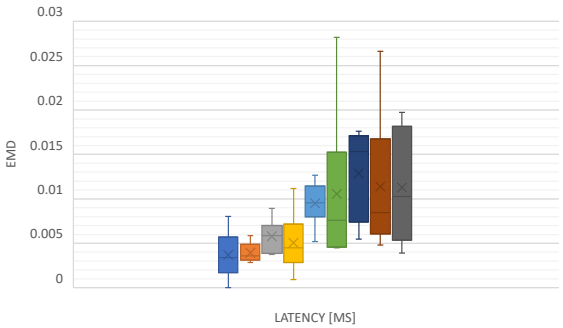
To conduct my evaluation, I executed the default ARIAC challenges, which include the presence of faulty products as an agility challenge. These faulty parts cannot be shipped and are detected by quality-checking sensors mounted on the AGVs. These sensors offer the advantage of fixed positioning, providing precise feedback regarding the final shipment. I utilized the images captured by these sensors as input for the EMD calculation.

5.1.3.1 Control strategy

In the ARIAC environment, the default control strategy is position control. However, with the expected increase of low-cost machines and robots the controller hardware is expected to transition to software executed on personal computers or in the cloud. This shift introduces velocity or effort control over the network, placing considerable strain on the network infrastructure. I have conducted an analysis of this setup.



(a) EMD as affected by network latency for Position #1



(b) EMD as affected by network latency Position #2

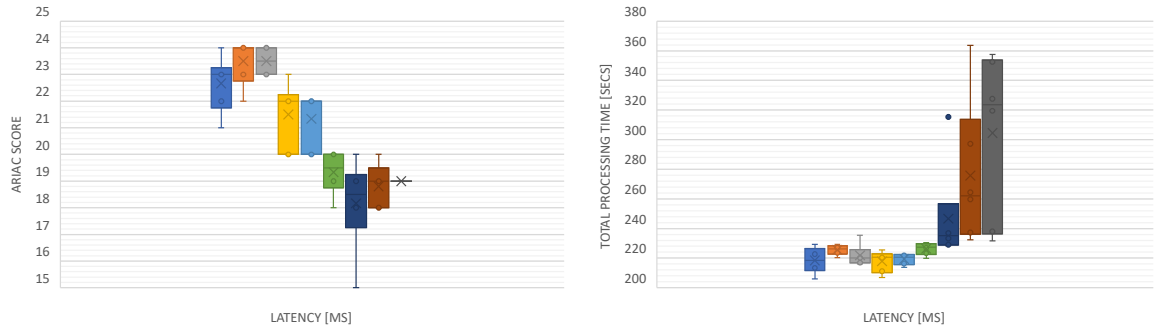
5.1.3.2 Comparison of output of quality check sensors

We need to quantitatively compare the resulting images from the quality check sensors based on network latency. Figure 5.2 shows the final position of the parts on the first AGV's tray compared to the baseline case.

To measure the variances, I opted to utilize EMD, which has proven effective in previous studies involving image databases [104]. The EMD is a statistical measure that quantifies the disparity between two probability distributions across a specified region. In pattern recognition, the EMD is utilized to assess generic data summaries known as signatures. Typically, a signature comprises pairs $((x_1, m_1), ...(x_n, m_n))$, where each x_i represents a "feature," and m_i denotes the "mass" or frequency of occurrence of that feature within the record. To compare two signatures using the EMD, I establish a distance metric between features. This distance reflects the expense of converting one unit of mass from one feature to another. Consequently, the EMD between two signatures represents the minimal cost required to transform one feature distribution into the other. I utilized [105] the EMD calculation included in OpenCV.

I performed 6 measurements for each latency setups. The EMD for a certain network latency case is compared to the no latency case. Figure 5.3a and Figure 5.3b show the 'Box and Whisker plot' of the measurements which encompasses the minimum value, first quartile, median, third quartile and maximum value of a data set. The EMD values demonstrate a consistent rise with the increase in network latency. This indicates that higher network latency leads to a decrease in the accuracy of part positioning onto the AGVs' trays. The positioning quality soon starts to drop after 4 ms latency.

Figure 5.4a depicts the relationship between network latency and the ARIAC score. This score, as defined by ARIAC [51], is automatically computed based on various components observed in each trial. These components combine cost and performance metrics, with



(a) ARIAC score in the function of introduced network latency

(b) Total Processing Time (TPT) in the function of introduced network latency

performance metrics encompassing both completion and efficiency. Completion metrics assess the quality of submitted shipments, while efficiency metrics evaluate the responsiveness in fulfilling orders.

Figure 5.4b displays the relationship between TPT and network latency. TPT represents the duration from the issuance of the first order to the end of the trial. Notably, higher latency leads to a significant increase in TPT values due to control inaccuracies introduced by network latency. Latency within the CPS affects both the pickup and placement of objects, with errors accumulating during both actions. In some instances, these errors exceed the tolerance limits set for task execution, prompting the strategic controller to initiate a rerun. This phenomenon illustrates how TPT escalates beyond control in setups with higher latency.

5.1.4 Estimated Mean Opinion Score

Telecommunications introduced a MOS [106] to gauge the overall quality of human-judged events or experiences. In typical telecommunication services, MOS serves as a rating for the quality of voice and video sessions, often assessed on a scale of 1 (poor) to 5 (excellent). Derived initially from surveys of expert observers, MOS has evolved to include objective measurement methods, such as Estimated Mean Opinion Score (EMOS), which approximate human rankings.

The implementation of fine-grained MOS has empowered operators to optimize network management efficiently to ensure acceptable QoE. Manufacturing processes could experience comparable advantages with the establishment of a detailed QoE metric. Currently, the KPIs of various manufacturing processes are diverse and challenging for network management systems to handle, necessitating a high-level, manufacturing-process-agnostic metric to streamline management tasks.

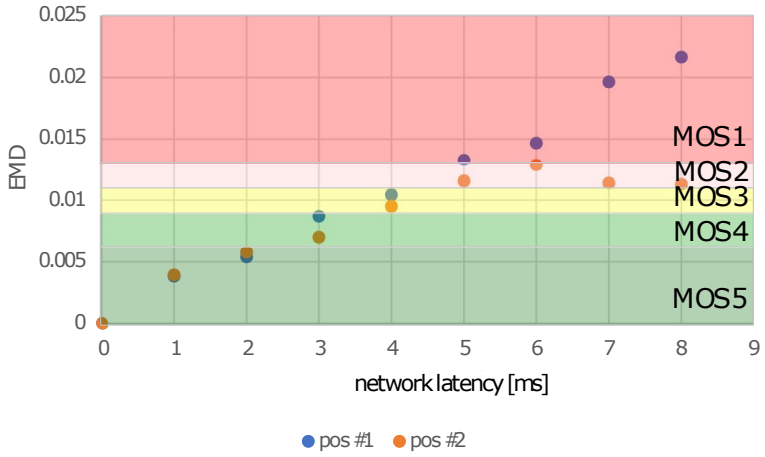


Figure 5.5: EMOS in the function of network latency

I propose a method to estimate MOS scores from EMD scores. My approach involves dividing the range of EMD scores between 0 and their maximum into five bins, with the MOS score determined by the bin index in which the EMD score falls.

To define the bin ranges, I consider both the ARIAC metric and TPT. Figure 5.4a illustrates that the ARIAC score peaks at a latency of 1 and 2 ms values, while Figure 5.4b indicates that TPT spikes significantly after a latency of 6 ms. The remaining latency cases between 3 ms and 5 ms are evenly divided among the corresponding MOS values.

The classification of EMD scores in Figure 5.5 into MOS scores is represented by the color code. Notably, the MOS score remains at its maximum until a latency of 3 ms and remains within an acceptable range until 4 ms latency.

In our previous work [107], we explored various approaches for initiating NRM by different stakeholders in the value chain and providing feedback to the network provider. The responsibility of the network operator is to configure and oversee cloud services to ensure that the delivered service quality (QoS_{nw}) meets the requirements. To accomplish this, it is essential to have insights into how the provided service is perceived by the robot operator. Different strategies can be employed depending on the availability of feedback information.

In [107], we proposed the 'detailed feedback from robot operator' framework, wherein the network receives QoS parameters and their utility function from the robot control system. With this information in hand, the network operator can compute and implement the necessary QoS parameters tailored to the specific use case. For my scenario, the definition of QoS_{nw} is as follows:

$$QoS_{nw} = g_{nw}(u_{robot}(), QoS_{robot}) = \left\{ \begin{array}{ll} MOS = 5, & \text{if } 0 < nw_{latency} \leq 0.006 \\ MOS = 4, & \text{if } 0.006 < nw_{latency} \leq 0.009 \\ MOS = 3, & \text{if } 0.009 < nw_{latency} \leq 0.011 \\ MOS = 2, & \text{if } 0.011 < nw_{latency} \leq 0.014 \\ MOS = 1, & \text{if } 0.014 < nw_{latency} \end{array} \right\} \quad (5.1)$$

5.2 Robot sanding use case

My intention in this section is to pursue the track to evaluate the perceived quality of a service of the user, extend this to the industrial and manufacturing area, and analyze a yet unexplored case how network QoS affects the quality of robotic sanding. This principle can be applied in other industrial areas where precision and process-speed requirements allow a broader range. These fields could be for example, painting, spraying, enameling, coating, iron casting, bonding, and sealing, etc. To achieve this, I performed and discuss the following steps in this section (also shown in Figure 5.6): 1) I set up a simulation environment including the calculation of the trajectories of the robotic arm with a sanding tool, 2) examine the execution of the trajectory with various introduced network effects, 3) model the sanding process on a certain surface, 4) introduce KPIs to measure sanding quality, 5) compare the sanding quality of various network setups, and 6) introduce a quantitative and an AI-supported method to deduce MOS.

A demonstration video is available at [108].

5.2.1 Related work

Authors of [109] explored the correlation between the sanding efficiency and the surface quality of wooden materials, providing a method for judging the end of belt life in the production. They provide certain KPIs like surface removal rate or surface roughness, which are good QoE KPIs but do not give a final verdict on the MOS.

Authors of [110] analyse parameters of sanding with an abrasive sanding machine to reduce energy consumption and to improve processing efficiency and quality. The influences of grit size, feed speed, sanding speed, and sanding thickness on the sanding force, normal force, arithmetic mean deviation of profile, power consumption, and power efficiency were analyzed by the orthogonal method in this study. A higher-level MOS score is preferred that is easy to comprehend and make actions in the network accordingly.

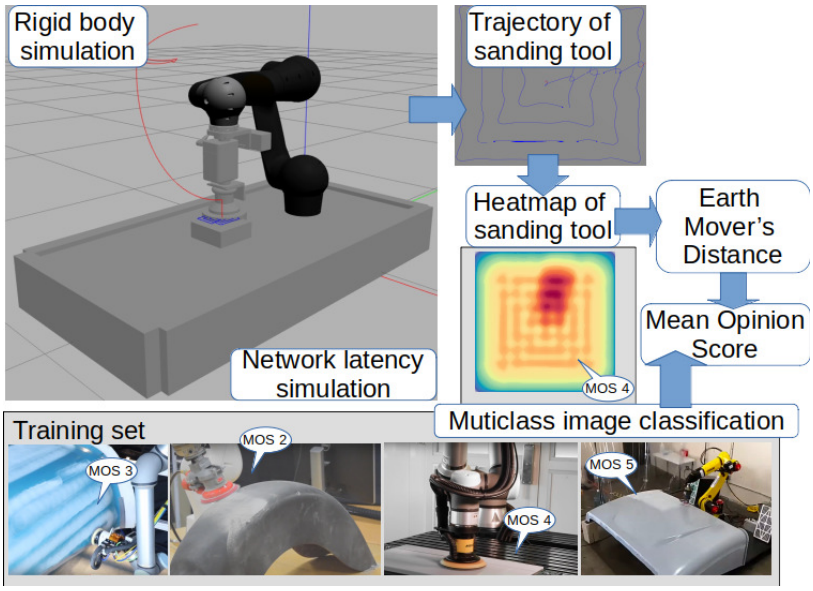


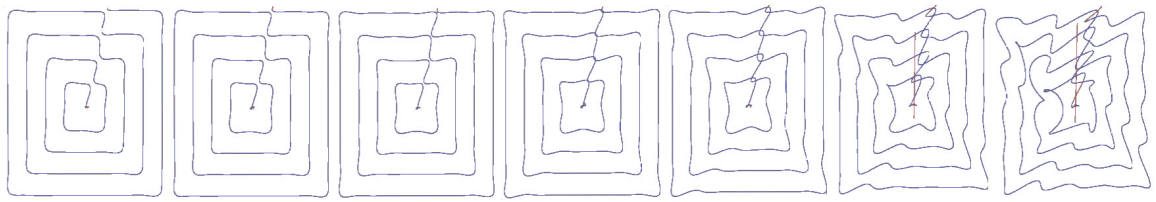
Figure 5.6: The architecture of the measurement setup

5.2.2 Test environment

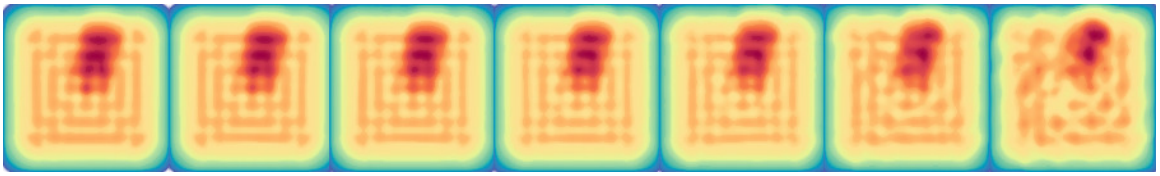
5.2.2.1 Baseline simulation environment

The simulated production cell environment consists of a robot arm equipped with a laser scanner and a sanding tool, that is controlled remotely with velocity commands. The output code of the project Scan and Polish [111] was utilized. Traditional robot programming requires an expert-in-the-loop. This limits responsiveness to changes in the process. Scan-N-Plan approach utilizes 3D scanning techniques to generate the part geometry and location on-the-fly, in real-time, eliminating the human in the loop and enabling agility. The part geometry is used as a direct input for the trajectory planner that creates custom trajectories for the specific scanned objects.

After the setup of the ROS packages, the robot can be triggered to scan the surface of the object with its laser scanner. Based on the scanned result, a trajectory for the sanding is planned in the following manner: 1) It creates a series of polygons to represent the paths for blending; 2) polygons are ordered so as to begin at most inner; 3) the path spirals out to the most outer; 4) then jumps to next incomplete inner; 5) finally spirals out to largest incomplete outer. The planned trajectory is executed by the robot arm.



(a) Trajectories.



(b) Heatmaps for tool size with radius 35 mm.

Figure 5.7: Trajectories and the resulting heatmaps for network latencies: no latency, 10, 20, 30, 40, 50, 60 ms from left to right, respectively.

5.2.2.2 Sanding plugin

I implemented another Gazebo plugin to have input for modelling the sanding. The plugin saves the path of the sanding tool by saving the x and y coordinates of the sanding tool if the z coordinate is below the surface of the sanded object. The resolution of this data is in the magnitude of 0.5 mm between two stored points.

5.2.3 Evaluation

My goal is to evaluate the setup in various network QoS settings. The expected effect of network latency is that though the planned trajectories remain the same, the execution and the final path of the sanding tool alter in the various network setup cases (see Figure 5.7a).

5.2.3.1 Control strategy

The default control strategy in Scan-N-Sand [111] is position control. In line with the explanation provided in Section 5.1.3.1, a velocity control strategy was further analyzed. The effects of latency were incorporated in a manner similar to that described in Section 5.1.2.3.

The sanding tool radius is 35 mm. The generated trajectories are independent of this size, but my sanding modelling takes this value into account.

5.2.3.2 Heatmaps

To make a quantitative comparison on the quality of sanding I started to look for quantitative KPI candidates. The sanding tool center is assumed to move along the saved output coordinate

list of the sanding plugin (see Section 5.2.2.2). Coordinates are saved in every simulation period, thus the list incorporates the speed of the sanding tool. I modelled the sanding as removing particles from a surface according to a two-dimensional normally distributed random variable $\{X, Y\}$ with zero mean and standard deviation set by the tool-size resulting in different sanding imprint for the various sanding strategies.

The imprint of the sanding tool is moved along each saved trajectory coordinate. The number of hits on a certain coordinate of the imprint is cumulatively added to the specific coordinate forming a heatmap. The resulting heatmaps can be seen in Figure 5.7b, for various network QoS settings.

5.2.3.3 Earth Mover's Distance as KPI

The resulting heatmaps need to be compared to each other as a function of network latency in a quantitative way. I decided to apply EMD as it has been done successfully in earlier studies for image databases [104]. I define the cost matrix as the distance between a pixel and all other pixels of the heatmap. Two neighboring pixels are considered to be one pixel away.

Figure 5.8 shows that the EMD values increase steadily as network latency grows. The EMD for a certain network latency case is compared to the specific sanding tool radius with no latency case. This means that increasing the network latency results in lower quality sanding. Until about 10 ms latency the sanding quality is not too sensitive on network latency.

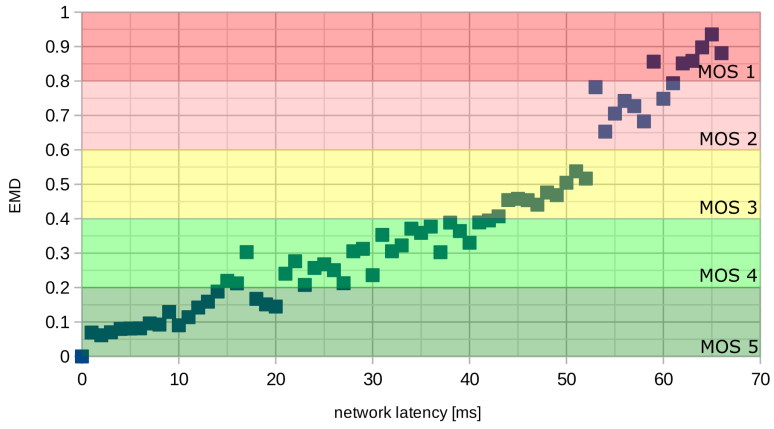


Figure 5.8: EMD in the function of network latency

5.2.4 EMOS

Fine-grained MOS gave the opportunity to operators to make room for network management to provide acceptable QoE in a cost-efficient way. The same would be beneficial for the manufacturing processes. I argue that several of these could be supported from the network management perspective in a more cost-efficient way if a fine-grained QoE metric were defined. The KPIs of various manufacturing processes are scattered among use cases – the very same robot can be used for healthcare and in a grocery store with different KPIs on the outcome of the operation – and overwhelming for a network management system. A high-level manufacturing-process-agnostic metric would make the network management tasks more plannable.

5.2.4.1 MOS estimation from heatmap distribution

In this section I discuss two simple methods to make an estimation on MOS score from the EMD scores.

In the first approach I divide the EMD scores between 0 and their maximum into five bins. The MOS score is simply the bin index in which the EMD score falls. The classification of the EMD scores of Figure 5.8 into MOS scores is shown by the color code. It this grouping, the size of the bins is equal. It is shown that the MOS score of the process is still at maximum even till around 20 ms latency and is well in the acceptable range till 40 ms latency.

The MOS score definitions for the sanding use case are the following:

$$QoS_{nw} = g_{nw}(u_{robot}(), QoS_{robot}) = \left\{ \begin{array}{ll} MOS = 5, & \text{if } 0 < nw_{latency} \leq 0.015 \\ MOS = 4, & \text{if } 0.015 < nw_{latency} \leq 0.043 \\ MOS = 3, & \text{if } 0.043 < nw_{latency} \leq 0.052 \\ MOS = 2, & \text{if } 0.052 < nw_{latency} \leq 0.061 \\ MOS = 1, & \text{if } 0.061 < nw_{latency} \end{array} \right\} \tag{5.2}$$

5.2.4.2 MOS estimation with multiclass image classification

My intention is to utilize the generated heatmaps for the sanding that shows the correlation of the quality and the network effect and add a more sophisticated automated method to provide the MOS.

I applied a multiclass image classifier that realizes a transfer learning for the VGG-16 image classifier model implemented in Keras. The code available on [112] was a good

candidate to handle this task. I need to provide it with training data. That requires a manual classification of the heatmaps into directories for the specific MOS score. Beside the generated heatmaps, I collected screenshots of automatic sanding processes available online [113]. The image classifier can learn the training data and recognize the estimated MOS scores accurately. It is not a challenging task for the algorithm. The human expert who does the manual classification of the screenshots and the heatmaps can introduce the most amount of uncertainty into the system. For this reason, in the case of MOS measurements, the human expert is trained first to have a mutual understanding on the scoring. This part needs to be elaborated in the future to make this process complete and reliable.

5.3 The 3D printing use case – OPC UA Evaluation

In this section, I delve into my proposals of Chapter 4 and assess their viability from various aspects.

5.3.1 Use case–3D printing

Section 3.2 describes a basic use case with a simple publisher and subscriber. The use case on which I intend to test the proposed method in the evaluation section is a more complex Industry 4.0 scenario, specifically 3D printing with robot arms that are remotely controlled from the edge cloud by velocity control via 5G. This is a viable architecture, inspired by a similar Industry 4.0 architecture discussed in 5G ACIA [26] on Figure 3 in which the robot is in the factory premise and controlled remotely via 5G from the edge cloud. This complex use case incorporates all the proposed elements discussed in Chapter 4.

In the following subsection, I describe the architectural elements of the evaluated system. The 3D printing use case is built by integrating several ROS packages, my proposed OPC UA to SEAL mapper, the NRM emulator server and a selection of the example applications from the proposed architecture to trigger NRM requests in VAL.

5.3.1.1 ROS Additive Manufacturing

ROS Additive Manufacturing (RAM) [114] offers a comprehensive collection of tools, comprising algorithms and GUIs, designed to assist users in creating intricate trajectories for additive manufacturing. The primary focus of this project lies in the printing of metallic components using industrial robots. RAM facilitates the generation of continuous trajectories, spanning from the initial stage of the manufacturing process to the completion of the part. To enable the seamless functioning of the entire application, the authors have developed a

range of ROS messages, services, actions, and nodes. The RAM software incorporates an RViz GUI that empowers users to choose from a range of trajectory generation algorithms. These algorithms enable the creation of trajectories based on a provided mesh or YAML file. Within the software, users have the flexibility to modify various printing parameters for each layer of the print, including blend radius, print speed, laser power, and material feed rate. The software allows for the customization of entrance and exit trajectories, granting users the ability to adjust parameters such as print speed, print angle, print length, and approach type to suit their specific requirements.

Authors claim that the objective of this project is to generate a robot program tailored to the specific brand of robot being utilized. The generated program can then be uploaded and executed on the robot. To implement this, a post-processor is required to convert the RAM trajectory into a program specific to the user's robot.

Another option involves using a ROS driver to establish a connection with the robot and utilizing MoveIt [115] for robot movement. In this approach, the RAM trajectory is read and then provided as input to MoveIt. Nonetheless, this approach is not currently being used by the project. My intention was to work within a simulated environment, primarily due to the unavailability of a welding tool and my limited expertise in setting one up. I chose the second option to connect the package with another solution that simulates 3D printing in the ROS environment with Gazebo [116].

5.3.1.2 3D Printing Robot Simulation with Viscoelastic Fluids

Authors of [117] describe the development of a simulation framework for realistic mobile 3D printing robot systems. They claim there is a need for a simulation platform that can handle the interactions between deformable 3D printing materials and other rigid bodies in a physics-based manner. To address this, the authors have created a simulation framework that combines particle-based viscoelastic fluid simulation [118] and particle-to-mesh conversion within the Gazebo [116] robotics simulator. This approach overcomes the limitations of traditional additive manufacturing simulation methods. The framework enables the simulation of robot arms or mobile manipulators in conjunction with viscoelastic fluids. The authors have conducted tests using different material properties and multiple collaborating robots to demonstrate the framework's capability to plan and control printhead trajectories, while also visually representing the printed fluid materials as a free-form mesh. Additionally, the scalability of the simulation in relation to the number of available material particles has been studied.

The provided implementation M3DP-sim [119] requires a trajectory to execute and that

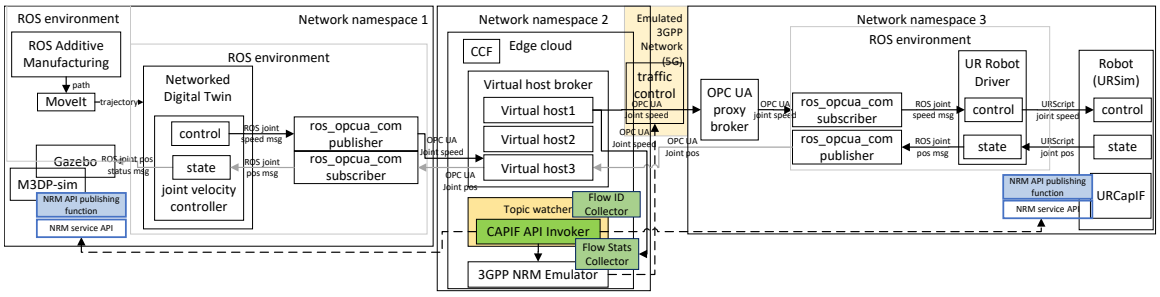


Figure 5.9: The architecture of the 3D printing use case

is the exact interface provided by RAM.

5.3.1.3 Networked Digital Twin

My goal was to evaluate my proposed architecture to trigger NRM requests in VAL. Out of the many examples considered in Section 4.3.4, my intention is to select and integrate a manufacturing related one that can effectively demonstrate both the seamless integration into the existing Industry 4.0 platform and the efficacy of my proposal. The selected application is the URCapIF (see [30] Section 5.4.2) tool and the SplishSplash extension (see [30] Section 5.4.1), which will be integrated into the 3D printing scenario. To provide an execution platform for URCapIF, I needed a real Universal Robot (UR) robot or its official simulator URSim [120]. Note that M3DP-sim simulates the 3D printing with interconnecting a fluid simulator (SplishSplash) in Gazebo (see Section 5.3.1.2). The requirement to integrate both Gazebo and URSim led us to include the Networked Digital Twin (NDT) [13], which is designed to show the network effects of a remotely controlled real robot in a simulated manufacturing cell, further enriching the evaluation process.

5.3.1.4 The proposed architecture for validation

The resulting architecture of the whole 3D printing system is shown in Figure 5.9. The following nodes and functions can be seen in the architecture diagram grouped according to the required namespace setup for the proper operation of the NRM emulation server.

- Network Namespace 1:
  - ROS Additive Manufacturing [114]
  - MoveIt [115]
  - Networked Digital Twin [13]
  - ros opcua communication publisher [121]

  - ros opcua communication subscriber [121]
  - Gazebo [116] with M3DP-sim [119] extended with CAPIF (see [30] Section 5.4.1)

- Network Namespace 2:
  - Virtual host broker (Section 4.2.2.3)
  - Topic watcher (Section 4.2.2.4)
  - Flow ID Collector (Section 4.2.2.3)
  - Flow Statistics Collector (Section 4.2.2.3)
  - Topic watcher (Section 4.2.2.4) extended with CAPIF API invoker (see [30] Appendix B.2.1)
  - 3GPP NRM Emulator (see [30] Section 4.1)

- Network Namespace 3:
  - OPC UA "proxy" broker (Section 4.2.2.3)
  - traffic control (see [30] Section 4.1)
  - ros opcua communication subscriber [121]
  - UR Robot Driver [122]
  - Robot emulated with URSim [120]
  - URCapIF (see [30] Section 5.4.2))
  - ros opcua communication publisher [121]

Architecture-wise the system is similar to [13]. It is built around the NDT, remote controlling the robot with velocity commands sent out with a 125 Hz periodicity. The RAM package creates paths for the robot that is passed towards the MoveIt package. The MoveIt package creates trajectories and provides them to the NDT to execute them with the joint velocity controller. The control loop here starts from the control box, transmitting the speed commands in ROS and OPC UA formats all the way through the black arrows until the robot. The robot provides joint states information which can be followed back through the gray arrows to the joint velocity controller and Gazebo. In [13] the communication link was deployed between the UR Robot Driver and the Robot, which sends out robot specific commands in URScript format [123]. To utilize OPC UA capabilities, I translate the ROS joint speed messages between the NDT and the UR Robot Driver into OPC UA messages with the ROS-OPC UA Communication package [121]. This tool creates a ROS

node subscribing to the specified ROS topic and publishes the received data to the OPC UA broker specified and vice versa.

Regarding the proposed OPC UA architecture in Section 3.2, I assume that a robot controller is deployed in the edge cloud within the robot's domain ("Network namespace 1"). It is connected to the 3GPP nodes and the network domain to "Network namepsace 2", where the "Virtual host broker" is deployed. This namespace is where the effect of NRM is emulated (discussed in [30] Section 4.1 in detail). The "Virtual host broker" connects to the OPC UA "proxy broker" (discussed in detail in Section 4.2.2.3) to make the QoS handling possible ("Network namespace 3"). The "OPC UA-ROS subscriber" can receive the joint speed in OPC UA format which is transformed into ROS topic format and the UR Robot driver [122] can control the robot transforming it into URScript [123] format. The reverse direction follows a similar approach. However, to ensure simplicity and clarity in explanation, I refrain from assuming QoS handling. From an architectural perspective, it means that there is no need for proxy broker in this direction.

In the last hop of the NDT, the received ROS joint position messages are forwarded to the Digital Twin in Gazebo that behaves exactly as the robot in the real word, getting the same joint positions in the simulated environment. On this simulated robot the 3D printing simulation [119] is applied, and I can make experiments with the automatic OPC UA based NRM requester (Section 4.2.2.4) handled by the SEAL NRM simulator (see [30] Section 4.1) and triggering the NRM service API invokers in both the fluid simulator (see [30] Section 5.4.1) and the URCapIF (see [30] Section 5.4.2).

5.3.2 The measurement

The measured scenario involves 3D printing a star-shaped object using a remote-controlled robot arm across five layers. Figure 5.10 illustrates the packet sending and receiving speed of robot commands over the emulated link as a function of time during specific events and settings. Each measurement of the entire five-layered star configuration takes several minutes. To maintain focus and facilitate explanation, I have extracted the relevant segments from the complete run and presented them in the figure. It is important to note that the measurements are aggregated into one-second bins. As a result, artifacts of TCP flow control do not appear in the figure. My objective was not to assess these intricate network-level details; rather, I aim to concentrate on the overall functionality. The utilized tool, tc [124], is well-known within the networking community, and its behavior is thoroughly understood [125].

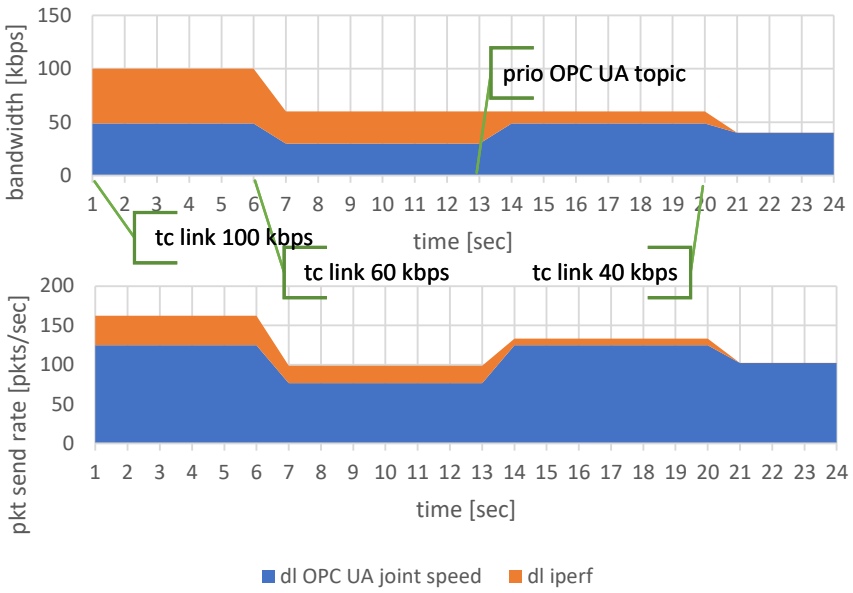


Figure 5.10: The bandwidth and packet sending rate of the robot commands and iPerf on the emulated link in the function of time during certain events and settings (stacked diagram)

The system is initialized. The measurement point is located at the "Emulated 3GPP Network" yellow box in Figure 5.9. I configured the default link size on the outgoing interface to 100 kbps. This specific value was chosen for the experiments to distinctly showcase the effects of the proposed system elements, thereby facilitating a clear and concise discussion of the results. It is worth noting that higher bandwidths are accessible in a production environment, especially if the network is over-provisioned for a high number of devices. However, the 100 kbps link capacity proves sufficient to accommodate control packets at a rate of 125 Hz in the downlink (indicated by the dark blue area). The message size is roughly 400 bytes of the control packets resulting in a 50 kbps downlink rate. The rest of the link capacity is filled in with an iPerf [126] flow indicated by the orange area that operates with MTU sized packets resulting in 50 kbps downlink rate with 37 packets/sec.

With this network setup I perform a simulated 3D printing considered later as the baseline case. The subfigure in the first column, first row of Figure 5.11 shows the planned path calculated by RAM with the executed path. The first column, second row shows the finished 3D printed object in Gazebo from above, and the first column, 3rd row shows the 3D printed object from side. The first column, 4th row shows the executed path from above depicted with hector trajectory server [127]. While I acknowledge that the resulting 3D print may not perfectly replicate a real-world object, it is important to note that both

Gazebo and SplishSplash are widely regarded as accurate simulators for rigid bodies [102] and incompressible, viscous fluids [128]. My integration of these simulators into my system has not compromised their accuracy in any manner. As a result, I treat the resulting object as a baseline representation for the ideal network use case. My intention is to subsequently compare the forthcoming measurements against this baseline.

Congested network scenario. In the second phase of the experiment, my objective is to induce packet loss and cause disruptions in the OPC UA topic. There are several methods to emulate this scenario, and I will explain my rationale for selecting a particular approach. One approach to achieve this goal is by inducing packet loss using the tc tool [124]. This method is the simplest and most straightforward. However, a drawback arises in that the designated drop rate needs to be set to an unreasonably high value to achieve a significant impact. Otherwise, the observed effects remain minimal and are swiftly mitigated by the TCP protocol. Another strategy to achieve the desired effect involves reducing the emulated link capacity using tc. Since TCP dynamically allocates bandwidth on a per-connection basis, I must set the bandwidth limit lower than the double of the required bandwidth of the control packets in order to produce the desired outcome.

In my experiment I chose to set the bandwidth limit of the emulated link to 60 kbps. As expected, Figure 5.10 shows that the two connections share the bandwidth fairly. In terms of packet number, the small sized control packets have about 75 packets/sec, while the MTU sized iPerf sends 22 packets/sec. The decrease of packet sending rate results in about 5 ms latency in packet arrival due to the TCP retransmission attempts, queuing in the brokers, etc.

In this scenario the second column of Figure 5.11 is relevant with the approximately 5 ms experienced latency. The latency in the emulated network is the dead time of the control process of the robot, and that is why the executed path alternates from the planned path more in the function of delay. The first row depicts the path followed by the fluid emitter during execution, while the second row illustrates the completed 3D printed object in Gazebo as viewed from above. The third row presents the same object from a side view, and finally, the fourth row showcases a comparison between the finished 3D printed object and the baseline printed object using CloudCompare [129]. CloudCompare is a versatile software tailored for the editing and processing of 3D point clouds and triangular meshes. Its main purpose lies in facilitating seamless comparisons between dense 3D point clouds. It can map two 3D objects onto each other, search for differences, and visualise them by highlighting the differences with a color grade as a 3D histogram. Green shows that the

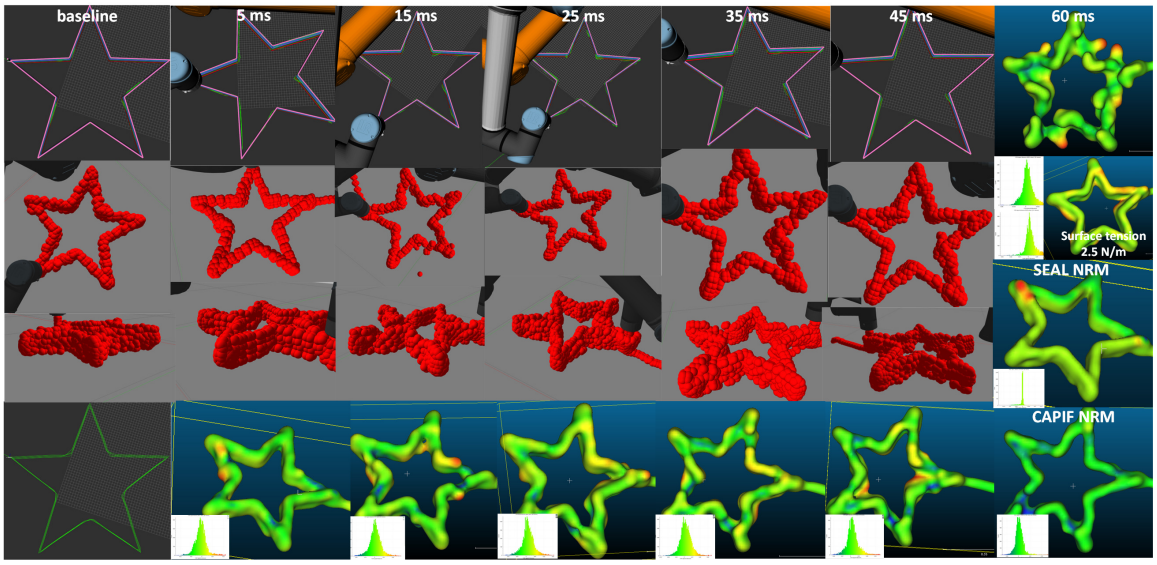


Figure 5.11: First row: The executed path by the fluid emitter; Second row: The finished 3D printed object in Gazebo from above view; Third row: The finished 3D printed object in Gazebo from side view; Fourth row: The finished 3D printed object compared to the the baseline printed object in CloudCompare [129]; The columns represent the various experienced latency values while the measurement is performed.

two objects are nearly identical at a certain location, while yellow and red means bigger deviation. The same color space is used to represent the histogram of object deviations in 2D shown in the lower left of the fourth row in every column.

Remarks on measurements with various latency setups. I started to evaluate industrial processes how sensitive they are in the function of network latency in [32]. I pursue further this topic by performing measurements with various latency values in my experimental setup to observe the effect of latency on 3D printing. The columns in Figure 5.11 represents the various experienced latency values while a certain measurement scenario is performed. Figure 5.12 shows the mean distance and standard deviation comparing the baseline object and the 3D printed one with a given latency calculated by CloudCompare [129]. We can see a trend that the latency has a direct effect on the 3D printed objects, and with higher latency, the difference is growing compared to the baseline case.

The results of these experiments highlight the sensitivity of 3D printing compared to the robust nature of sanding, as demonstrated in [32], especially for larger latency values. Before conducting the experiments, my expectation was that the fluid would compensate for errors during printing, resulting in objects similar to the baseline regardless of network performance. However, what I observed is that while this effect does occur, the errors are so

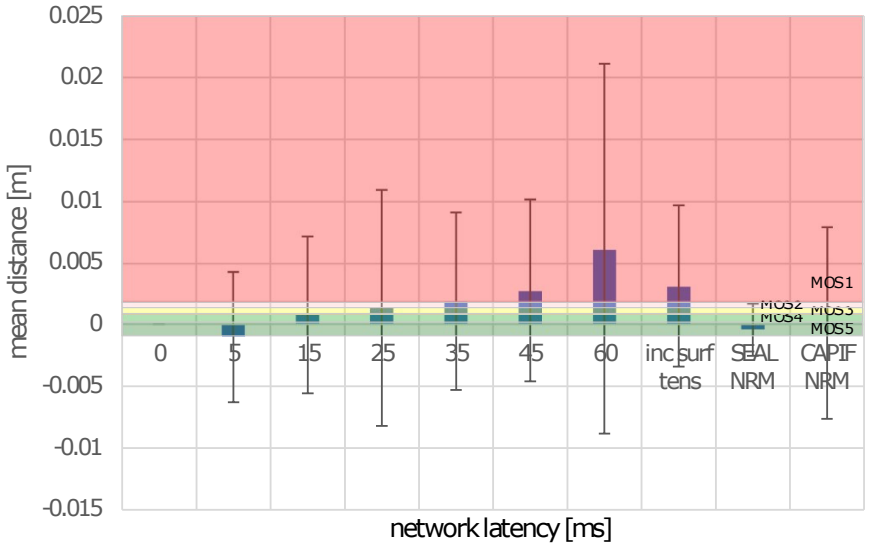


Figure 5.12: The mean distance and standard deviation comparing the baseline object and the 3D printed one with a given latency calculated by CloudCompare [129]

consistent in each layer that they systematically degrade the overall layer layout, leading to distortion in the final printed object.

Figure 5.12 indicates that the most significant deviations occur within the range of 3-5 cm. Such deviations are rare for lower latency values but become more frequent as latency increases. However, it is important to note that QoE is heavily dependent on the specific use case. It is worth emphasizing that the objects being 3D printed are of considerable size. Therefore, these scenarios should be approached for instance in the context of sustainable 3D printing for constructing houses with robotic systems using concrete, as discussed in [130].

The MOS score definitions for the 3D printing use case are the following:

$$QoS_{nw} = g_{nw}(u_{robot}(), QoS_{robot}) = \left\{ \begin{array}{ll} MOS = 5, & \text{if } 0 < nw_{latency} \leq 0.005 \\ MOS = 4, & \text{if } 0.005 < nw_{latency} \leq 0.015 \\ MOS = 3, & \text{if } 0.025 < nw_{latency} \leq 0.035 \\ MOS = 2, & \text{if } 0.035 < nw_{latency} \leq 0.045 \\ MOS = 1, & \text{if } 0.045 < nw_{latency} \end{array} \right\} \tag{5.3}$$

Based on the experiments in Figure 6 of [117], I assumed that increasing the viscosity and

decreasing surface tension makes the 3D printed objects more defined and rigid, rendering them more sensitive to network effects. For applications like precise 3D welding, as in [114], latencies exceeding 5 ms are likely unacceptable. Conversely, a 3D construction scenario might tolerate network latencies above 25 ms. These diverse fluid parameters enable the simulation of various use cases.

Starting the proposed OPC UA to SEAL mapper. When the proposed OPC UA to SEAL mapper method is started, the "Flow ID collector" and "Flow statistic collector" nodes provide the measured data on the OPC UA topic to the "Topic watcher" to trigger the SEAL NRM request. The "3GPP NRM Emulator" receives the NRM requests, and it sets the Linux traffic control [124] of the OPC UA flow according to the request. To make the setup work, the "local proxy broker" needs to be configured for the subscriber side similarly as it is shown for the publisher case in Figure 4.3. Due to the priority set for the robot control messages, the TCP fairness is not valid anymore and the control messages can get the required bandwidth for the 125 Hz control traffic, while the iPerf traffic can work on remaining roughly 11 kbps bandwidth (see Figure 5.10).

So far, this setup proves that the proposed system can be easily integrated into existing deployments as the ROS environment can be easily turned into the OPC UA-aware communication. The OPC UA to SEAL mapper architecture and the nodes work properly while the emulated SEAL NRM server operates as well.

In Figure 5.11, the subfigure labeled "SEAL NRM" in the 3rd row and 7th column depicts the resulting 3D printed object alongside the baseline case, along with a histogram showcasing distances. It is evident that the two objects are remarkably similar. This similarity highlights the rationale behind assembling the evaluation system to demonstrate the efficiency of the proposed system within a complex and authentic Industry 4.0 scenario, and to quantitatively assess its capabilities in relation to the KPIs of the examined industrial process.

As part of another experiment, I conducted a test run with increased surface tension. The outcome labeled as "Surface tension 2.5 N/m", when compared to the baseline 3D printed object, is visible in the second row, 7th column of Figure 5.11. The accompanying histogram, displaying the distances between corresponding points, is positioned beneath it (located in the bottom histogram of the second row, 7th column). In Figure 5.12, labeled "inc surf tens", it is evident that the object with less distinct definition significantly diverges from the baseline case, even with prioritized traffic.

The effect of the proposed architecture to trigger NRM requests in VAL. In this phase I set the emulated network link capacity to 40 kbps, which is lower than the required bandwidth for the control packets sent with 125 Hz frequency. The "Flow stats collector" collects the required bandwidth on the sending side being aware of the required 50 kbps bandwidth, making the correct request for the SEAL emulated NRM server.

The received NRM request is above the link capacity and the NRM emulation server sends out the "Application QoS change notification" type of message defined in 3GPP TS 23.434 [11] §14.3.2.16 to the "VAL server adaptation trigger and router" which is the "Topic watcher" in my setup. The "Topic watcher" starts the CAPIF process looking for suitable APIss to fulfill the NRM downgrade call. The two suitable APIs it finds are the URCapIF (see [30] Section 5.4.2) and the Gazebo with M3DP-sim [119] extended with CAPIF (see [30] Section 5.4.1). Its calculations show that the available bandwidth is about 80% of the requested bandwidth, thus making NRM service API calls for the two published NRM services to reduce the process speed to 80%. This method can be considered as a dynamic NRM case, for which any new QoS change notification, the system can react.

The subfigure in the 4th row, 7th column of Figure 5.11 labeled as "CAPIF NRM" shows the resulting 3D printed object compared to the baseline case. Looking at the histogram and on Figure 5.12 "CAPIF NRM" column we can see that the two resulting objects are nearly identical. This visual correspondence highlights the closely aligned characteristics, allowing us to perceive the measurable benefits provided by the VAL NRM API services.

Remarks on the implementation. If URCapIF were unable to modify the robot's joint state publishing rate, the robot's joint state messages would continue to impose the same network load. However, any packet drops would no longer lead to control errors during trajectory execution. In such a scenario, it would be advantageous to activate my Mosquitto rate limiter plugin (refer to [30] Section 5.4.3) on an additional "proxy broker" within "Network namespace 3". This proxy broker would be positioned between the 'ros_opcua_com publisher' and the Virtual host broker.

Architecture-wise, the M3DP-sim [119] should be located within "Network namespace 2", emulating the remote control of the printing emitter. The current implementation facilitates local in-memory communication between SplishSplash [118] and Gazebo [116] through the [119] library. It would require additional effort to separate the two and enable communication through a network that allows for the isolation of SplishSplash within "Network namespace 2", as it was operated remotely.

To connect the ROS environments and the related nodes – UR Robot Driver [122],

ros_opcua_com [121] – in "Network namespace 1 and 3", the ROS_MASTER_URI has to be set up that tells the nodes where they can locate the ROS master.

The ros_opcua_com [121] package utilizes event-based publishing of the OPC UA topics. It subscribes to all selected ROS topic, action, service, or server channels and when a message occurs on any of these it is translated to an OPC UA topic and published. Note that I needed to update and exchange the OPC UA layer of the [121] to provide pubsub communication as the original implementation supports only client-server-based communication.

The required lines of code in the proposed method is similar to the ROS2 case, please refer to 3.5.1.

Chapter 6

Summary of The Scientific Results

Theses 1,. ***Utilizing NRM exposure for sensors in an agile production cell*** *[28] (Chapter 2) I propose a method for replacing wired communication within an operational agile production cell with wireless communication, using a minimal number of access points. This solution benefits from the recent standardized NRM functionalities of 5G, which are invoked on UE connected to cameras. This proposal combines the efficiency of 5G radio utilization with precise modeling and high-resolution simulations, significantly improving the learning speed in the process.*

Thesis 1.1,. *[28] (Section 2.4) I propose a novel DT environment that serves as a learning environment for RL algorithms. The proposed DT operates with near real-time speed—or even faster—while simulating reality with high accuracy. It achieves this by utilizing solid-state physics simulation and hardware-accelerated ray tracing for radio wave propagation simulation.*

Thesis 1.2,. *[28] (Section 2.4) I propose a new method that optimizes the arrangement of radio access points in a manufacturing cell and the quality of video streams provided by cameras within the cell, which are essential for the cell's operation. The approach is based on introducing AI-based agents using RL. These agents are assigned to the access points to optimize their placement and work collaboratively with agents assigned to the camera systems. This collaboration aims to optimize the quality of the video streams by dynamically and demand-based allocating the network resources requested by the cameras, taking into account the characteristics of radio coverage and the operation of the manufacturing cell.*

In RL (see Figure 2.2), the observation space refers to the information an agent receives from the environment, which helps it make decisions. For the radio dot, this includes positions of other dots, gantry position, signal losses, remaining steps, and camera QoSensing. Loss is

calculated across the network using ray tracing, and rewards encourage learning based on actions, influencing learning rate and convergence. The camera's observation space focuses on the path with minimal signal loss and gantry position. These spaces are universal and don't depend on the production cell layout.

The action space defines the set of actions available to agents. Both the radio dot and camera have discrete action spaces. The radio dot's actions include movement in four directions (forward, backward, left, right) and height adjustments to simplify learning. The camera chooses between high or low QoS connectivity, impacting resolution and task accuracy. The action and observation spaces are linked when agents communicate, with connectivity based on pathloss thresholds that ensure adequate data transmission.

Thesis 1.3,. *[28] (Section 2.5) I propose a novel curriculum learning-based method to account for the dynamic nature of a production cell. The creation of customized curricula is based on grouping task domains. The discrete task domain encompasses the static and operational production cell, while the continuous task domain characterizes tasks related to reconfiguring ray tracing resolution, signal strength, and antenna properties. The reward functions are tailored to the curricula, ensuring the efficiency and goal-oriented nature of the learning process.*

I establish two operation modes for radio dots and the production cell (see Figure 2.3): static and dynamic. In the static dot scenario, dots remain fixed without movement or the ability to seek improved coverage. The static cell scenario means the production cell remains non-operational. In the dynamic dot scenario, radio dots can move freely to optimize coverage but remain static during the inference phase. In the dynamic production cell scenario, the production cell is in operation with robots actively controlled and in motion. Based on these four options, I introduce four curricula in the subsequent paragraphs.

Theses 2,. ***The vertical application is the consumer of network exposure, automatic VAL application to 3GPP SEAL mapping*** *[30, 29] (Chapter 3,4) I propose an architecture and method that can deduce information from the VAL applications and make request towards 3GPP SEAL automatically.*

Publish-subscribe protocols operate over a single IP connection between the publisher and subscriber. Within this single IP connection, multiple topics can be subscribed to, and multiple messages can be published and received. Messages are encapsulated within IP packets and transmitted over the established connection. This operation mode prevents communication nodes from explicitly differentiating network QoS for publishers and subscriptions on IP flow level. Thus, only the same network QoS can be assigned to publishers

and subscriptions in communicating nodes. The proposed solutions overcome this issue.

Thesis 2.1,. *[29] (Section 3.2) I propose an architecture and custom nodes for ROS2 to collect information on the behavior and requirements of the VAL application and make request towards 3GPP SEAL automatically. I propose a proxy node that collects the information and a node performing the mapping from ROS2 settings to 3GPP SEAL requests.*

I provided a solution to map the ROS2 application layer information elements automatically to these requests. The ROS2 application developer's only job is to tag the ROS2 topics that need special 3GPP QoS handling. I prove that my proposed solution is feasible and lightweight. I cover three SEAL functionalities: group management, network resource management and connection monitoring.

Thesis 2.2,. *[29] (Section 3.4) I propose a tunnel mode deployment of Thesis 2.1 1) to provide network monitoring functions even if 3GPP SEAL network monitoring functions are not provided by the 3GPP exposure interface and 2) to provide enhanced dynamic QoS handling for the ROS2 applications if dynamic QoS switching is not supported by the 3GPP core or radio.*

I connect the UFTRs' of the ROS2 application instances then the UFTRs publish the relayed topics to another UFTR instance which subscribes to it creating a tunnel between the two (see Figure 3.9). The publisher is initiated with a custom message type including a timestamp, a message counter, and a binary field. The timestamp is filled with the clock, the message counter is an ever-increasing integer, and the binary data is the whole message received on the general subscriber topic. Note that there is no parsing happening in this case, thus it is message type agnostic, an encapsulation occurred only.

The application publishes a topic tagged with `_qos_out`, and the UFTR subscribes using a general subscriber. Multiple publishers are then initiated as tunnel start-points – the DEMUX mode of operation –, encapsulating messages in new headers and transmitting them via QoS handling. Topics are tagged with a priority level or QCI value "_QCI_8". The UFTR directs incoming messages to specific outgoing topics with unique flow IDs and QCI tagging, ensuring QoS through NRM requests for known 5-tuples (see Figure 3.9). In the second mode of operation (MUX), the UFTR subscribes to all topics tagged with "_QCI", parses headers, and forwards the binary data stream via a general publisher. Topics tagged with `_qos_in` indicate they are decapsulated and ready for forwarding.

Thesis 2.3,. *[30] (Section 4.2) I propose a novel architecture and custom nodes for OPC UA to collect information on the behavior and requirements of the VAL application and*

make request towards 3GPP SEAL automatically. The architecture consists of a proxy broker and a virtual host broker which provides the necessary information and a node performing the mapping from OPC UA settings to 3GPP SEAL requests. I cover the network resource management and network monitoring functions of SEAL.

To achieve the desired outcome, I utilized a feature known as "topic routing". The broker in the edge cloud is extended with the "Virtual Host" capabilities. This feature enables the broker instance to listen on multiple ports. Note that every subscriber has the capability to connect to any port and receive the data that has been published on any of those ports. The idea here is to create as many virtual hosts as many QCI I intend to handle in the system. For each QCI class a separate virtual host is defined. As Figure 4.3 shows, the bridges from the proxy broker and the virtual host brokers define unique flows for topics that need special QoS handling on the wireless channel. These IP flows have no multiplexed topics and the 5-tuple identifier of the flow becomes unique for each MQTT topic.

Theses 3,. ***The network as the consumer of application exposure*** *[30, 29] (Chapter 3,4) I propose a novel architecture that puts the network into the role of the consumer of the application exposure.*

Thesis 3.1,. *[29] (Section 3.3) I propose a novel architecture layer beside the VAL layer, which I call VASEAL. The proposed architecture provides the necessary interfaces to make the VAL application observable and controllable. I propose an observability node that provides information on the execution of the VAL process both in a MOS and EMOS way. I propose a production cell resource management interface that influences the VAL process execution speed and resolution.*

I introduced the VASEAL that exposes the vertical application layer towards the network. Figure 3.7 shows the concept. Similarly, as in SEAL, VASEAL has four main components: VASEAL Service Server and Client and their respective network side nodes, the Network Layer Service Server and Client. VASEAL Service Server runs next to the Vertical Application Server hosting the Resource Management Server and Event Monitoring for the production cell or other VAL application. VASEAL Service Client runs next to the Vertical Application client running the GUI for the data collection of the event monitoring. The Network Layer Service Client runs next to the SEAL Client. The Network Layer Service Server runs next to the SEAL Service Server.

Thesis 3.2,. *[29] (Section 3.3.3, 3.5) I propose a novel solution in case of 'Application QoS change notification' from the 3GPP SEAL applying RL to keep a certain QoE of*

the user while fitting into the available network characteristics. The proposal involves the custom behavior of the proxy node by filtering packages on-the-fly emulating the network characteristics to make the RL algorithm learn fast without modifying either the network or the application.

Figure 3.11 shows the architecture of the proposed system. The proposed solution is a RL based solution. Every topic that is handled by the UFTR has an associated AI-agent, a rollout worker in RLlib terms. The external *environment* is the ROS2 setup. The *observation space* is collected by subscribing to various ROS2 topics: bandwidth and jitter values of the flow statistics, MOS, latency, dropped packet and control speed. The *action space* of the agents consists of the increasing or decreasing the latency, dropped packet, control speed with one minor step in the specific value set. The decided action of the agent is published through the above ROS2 topics.

The reward function for each topic is the following:

$$reward_{topic} = w_1 * mos + latency_{topic} - drop_{topic} \tag{6.1}$$

, where

- w_1 is selected to be 200 in my measurements,
- *mos* is score is defined as the square error between the desired position and current position of the turtle scaled with the process speed,
- $latency_{topic}$ is considered as the number of times the "UFTR with Throttle" delays the packet. Its effect depends on the topic update rate.
- $drop_{topic}$ means that "UFTR with Throttle" drops every n^{th} packet of the topic. A higher value results in less drop.

Note that the reward function does not contain the VAL process speed. It is included in the MOS score provided by the custom turtle controller.

Thesis 3.3,. *[30] (Section 4.3) I propose a novel solution for integrating non-adaptive applications into the 3GPP NRM domain, including those that do not automatically adjust to the channel conditions. My solution leverages CAPIF and its extensions beyond the 3GPP domain to achieve this.*

3GPP TS 23.434 [11] provides NRM functionality which is implemented on the radio channel and the core network. While the NRM affects the channel, the traffic of the VAL

application transferred over it is not altered in any way. If the 3GPP NRM request fails to trigger adaptive changes in the VAL traffic and the VAL traffic itself does not possess self-adaptive capabilities, there is a risk of experiencing considerable performance degradation in the VAL traffic.

In my proposal, I connect the NRM management with the execution of the NRM on the VAL and fit the VAL traffic into the requested channel. The proposed solution can be executed in the UE's VAL layer, in the OS, or on the VAL server and the transport layer of the 5G/6G network as well. The proposed method adds new elements and interfaces to the functional model of the 3GPP TS 23.434 [11]. The proposed method extends the existing NRM request API with new information elements.

Theses 4,. ***Impact of NRM on the industrial processes*** *[30, 31, 32] (Chapter 5) I propose a MOS like quality assessment method for various industrial processes.*

Network management aims to balance over-provisioning costs with delivering satisfactory QoE by optimizing QoS parameters. In MBB, application performance, such as web page load and video startup times, is influenced by browser and protocol design. The relationship between QoS and QoE is well recognized, though translating this link in manufacturing settings remains challenging due to strict network requirements. With the advent of 5G, which offers real-time communication and reliability, the technology is pivotal for Industry 4.0, particularly in supporting tasks like remote industrial robot operations. However, while mobile communications have historically defined QoE in terms of user satisfaction for voice and video, manufacturing processes often demand binary outcomes regarding product quality. Extending insights from 5G's low-latency capabilities to other industrial processes could improve network agility and align performance metrics more closely with manufacturing needs.

Thesis 4.1,. *[31] (Section 5.1.4) I propose a novel method for the quality assessment of a pick and place process in the function of the network performance. The method provides an objective metric for evaluating the placement accuracy of objects positioned by the robot and, based on this, estimates a subjective MOS value. The method examines the impact of remotely controlling a robot via an edge-cloud-based robot controller, where the robot's velocity vector-based control is executed over a radio network while considering the effects of network latency.*

[106] defines the measurement setup including the environment, the training of the human expert and the evaluation criterias strictly. Similarly, I state that in the ARIAC 2020

environment the MOS for the pick an place process can be estimated in the function of network latency.

To achieve this goal, I undertake the following steps (also outlined in Figure 5.1): 1) Establish a simulation environment encompassing the remote control of a robot gantry via 5G radio from the edge cloud; 2) Investigate the execution of order fulfillment under various network conditions; 3) Introduce KPIs for assessing pick-and-place quality; 4) Compare the quality of pick-and-place executions under different network configurations; 5) Propose a quantitative method for deriving MOSs.

Thesis 4.2,. *[32] (Section 5.2.4) I propose a novel method for quality assessment of sanding processes in the function of the network performance. The method provides an objective metric for evaluating the quality of surfaces processed by the robot and uses these results to estimate a subjective (MOS) value. The approach examines the interaction between the robot and the remotely operated control system running on the edge cloud, where the robot's velocity vector-based control is executed over a radio network while considering the effects of network latency.*

The utilized environment is the Scan and Sand [131]. The MOS for the sanding process can be estimated in the function of network latency. To achieve this, I performed the following steps (also shown in Figure 5.6): 1) I set up a simulation environment including the calculation of the trajectories of the robotic arm with a sanding tool, 2) examine the execution of the trajectory with various introduced network effects, 3) model the sanding process on a certain surface, 4) introduce KPIs to measure sanding quality, 5) compare the sanding quality of various network setups, and 6) introduce a quantitative method to deduce MOS.

Thesis 4.3,. *[30] (Section 5.3) I propose a method for the quality assessment of 3D printing in the function of the network performance. The method provides an objective metric for evaluating the geometric distortions of objects printed by the robot, comparing them to prints conducted in a latency-free network environment. Based on the results, the method estimates a subjective (MOS) value. The approach examines the interaction between the robot and the remotely operated control system running on the edge cloud, where the robot's velocity vector-based control is executed over a radio network while considering the effects of network latency.*

The achieve this I propose an architecture shown in Figure 5.9. The measured scenario involves 3D printing a star-shaped object using a remote-controlled robot arm across five layers. With this measurement setup I perform a simulated 3D printing considered later as

the baseline case. In the second phase of the experiment, my objective is to induce packet loss and cause disruptions in the OPC UA topic. Figure 5.12 shows the mean distance and standard deviation comparing the baseline object and the 3D printed one with a given latency calculated by CloudCompare [129]. We can see a trend that the latency has a direct effect on the 3D printed objects, and with higher latency, the difference is growing compared to the baseline case.

Bibliography

[1] K. Abdulkadir, D. Ognjen, S. Dirk, S. Gergely, N. Ala, P. Hubert, and S. Joachim, "Managing 5G Non-Public Networks from Industrial Automation Systems," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023.

[2] G. Seres, D. Schulz, O. Dobrijevic, A. Karaagac, H. Przybysz, A. Nazari, P. Chen, M. L. Mikecz, and A. D. Szabo, "Creating Programmable 5G Systems for The Industrial IoT," *Ericsson Technology Review*, vol. 2022, no. 10, pp. 2–12, 2022.

[3] G. Szabó, G. Seres, M. L. Mikecz, M. Hortobágyi, S. Rácz, J. Pető, and T. Cinkler, "Assessment of the Efficiency of 5G Network Exposure for the Industrial Internet of Things," in *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2021, pp. 52–58.

[4] "External configurability of QoS policies," Nov 2020. [Online]. Available: http://design.ros2.org/articles/qos_configurability.html

[5] "OPC Foundation UA Part 14: 5.4.5.4 QoS configuration ," Nov 2022. [Online]. Available: https://reference.opcfoundation.org/Core/Part14/v105/docs/5.4.5.4

[6] "OPC 10000-22: UA Part 22: Base Network Model," Nov 2022. [Online]. Available: https://reference.opcfoundation.org/Core/Part22/v105/docs/

[7] "FastDDS," Aug 2023. [Online]. Available: https://www.eprosima.com/index.php/products-all/eprosima-fast-dds

[8] "Mapping of OPC UA Qos to MQTT QoS in open62541," Jul 2023. [Online]. Available: https://github.com/open62541/open62541/blob/2e9959fed844e182ba3dbd2706eba6e973e44585/plugins/ua_pubsub_mqtt.c#L24

[9] "3GPP TS 23.501, System architecture for the 5G System (5GS)," June 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-h11.zip

[10] "3GPP TS 23.502, Procedures for the 5G System (5GS)," June 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/23502-h10.zip

[11] "3GPP TS 23.434, Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows," June 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.434/23434-h20.zip

[12] "3GPP TS 29.549, 29.549 Service Enabler Architecture Layer for Verticals (SEAL); Application Programming Interface (API) specification; Stage 3," Sept 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.549/29549-h60.zip

[13] G. Szabó, S. Rácz, N. Reider, H. A. Munz, and J. Pető, "Digital Twin: Network Provisioning of Mission Critical Communication in Cyber Physical Production Systems," in *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 2019, pp. 37–43.

[14] G. Szabó, J. Pető, L. Németh, and A. Vidács, "Information Gain Regulation In Reinforcement Learning With The Digital Twins' Level of Realism," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–7.

[15] G. Szabó, L. Vajda, J. Pető, and A. Vidács, "Radio Resource-and Quality of Control-aware Planning For Self-reconfiguring Factory Cells," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6.

[16] "rclpy/rclpy/rclpy/qos_overriding_options.py," Nov 2022. [Online]. Available: https://github.com/ros2/rclpy/blob/rolling/rclpy/rclpy/qos_overriding_options.py

[17] "ROS2, Recording and playing back data," May 2022. [Online]. Available: https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Recording-And-Playing-Back-Data/Recording-And-Playing-Back-Data.html

[18] "ROS2Executors," Jan 2023. [Online]. Available: https://docs.ros.org/en/rolling/Concepts/About-Executors.html

[19] "FastDDS 3.1.2.1. Standard QoS Policies." [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest/fastdds/dds_layer/core/policy/standardQosPolicies.html

[20] "MQTT specification," May 2023. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Toc3901103

[21] "Camaraproject/QualityOnDemand," Jan 2023. [Online]. Available: https://github.com/camaraproject/QualityOnDemand/blob/release-0.8.1/code/API_definitions/qod-api.yaml

[22] "The 3rd Generation Partnership Project," 2021. [Online]. Available: https://www.3gpp.org/

[23] "3GPP TS 29.522, 5G System; Network Exposure Function Northbound APIs; Stage 3," June 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.522/29522-h20.zip

[24] "Network Configuration Protocol (NETCONF)," June 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6241

[25] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses in The Wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[26] 5G-ACIA, "Exposure of 5G Capabilities for Connected Industries and Automation Applications," *5G-ACIA white paper*, 2021. [Online]. Available: https://5g-acia.org/whitepapers/exposure-of-5g-capabilities-for-connected-industries-and-automation-applications/

[27] G. Szabó, S. Rácz, N. Reider, J. Pető, and R. Roque Aschoff, "Quality of Control-Aware Resource Allocation in 5G Wireless Access Networks," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 1–6.

[28] G. Szabó and J. Pető, "Intelligent Wireless Resource Management in Industrial Camera Systems: Reinforcement Learning-based AI-extension for Efficient Network Utilization," *Computer Communications*, vol. 216, pp. 68–85, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366423004607

[29] G. Szabó, "Toward the Automatic Network Resource Management of Robot Operating System in Programmable Mobile Networks," *IEEE Access*, vol. 11, pp. 65 934–65 955, 2023.

[30] ——, "Towards The Automatic Network Resource Management of OPC UA in 5G Private Networks," *Computer Networks*, vol. 250, p. 110581, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624004134

[31] G. Szabó, M. Balogh, and A. Vidács, "Impact of Network Resource Management on the Quality of Pick and Place Processes," in *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2024, pp. 1–5.

[32] G. Szabó, Z. Trombitás, J. Pető, T. Lohmar, I. Komlósi, T. Pepó, A. Vidács, and M. Andó, "Impact of Network Resource Management On Quality of Industrial Processes," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–4.

[33] J. W. Webb and R. A. Reis, *Programmable Logic Controllers: Principles and Applications*. Prentice Hall PTR, 2002.

[34] A. Pekar, J. Mocnej, W. K. G. Seah, and I. Zolotova, "Application Domain-Based Overview of IoT Network Traffic Characteristics," *ACM Comput. Surv.*, vol. 53, no. 4, jul 2020. [Online]. Available: https://doi.org/10.1145/3399669

[35] E. F. P. Meléndez, L. O. P. Asqui, L. S. Córdova, T. G. B. Cabrera, and M. A. F. Bayas, "Analyzing And Improving Quality Of Sensing In Wireless Sensor Network," in *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 2019, pp. 1–7.

[36] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[37] "Demo video," Jan 2023. [Online]. Available: https://youtu.be/ZzzCjnnDRxY

[38] C. Czwick and R. Anderl, "Cyber-physical Twins - Definition, Conception and Benefit," *Procedia CIRP*, vol. 90, pp. 584–588, 2020, 27th CIRP Life Cycle Engineering Conference (LCE2020) Advancing Life Cycle Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827120301554

[39] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: https://doi.org/10.1145/3469029

[40] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, 05 2000.

[41] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[42] R. Felbecker, L. Raschkowski, W. Keusgen, and M. Peter, "Electromagnetic wave propagation in the millimeter wave band using the NVIDIA OptiX GPU ray tracing engine," in *2012 6th European Conference on Antennas and Propagation (EUCAP)*, 2012, pp. 488–492.

[43] D. He, B. Ai, K. Guan, L. Wang, Z. Zhong, and T. Kürner, "The Design and Applications of High-Performance Ray-Tracing Simulation Platform for 5G and Beyond Wireless Communications: A Tutorial," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 10–27, 2019.

[44] F. Qureshi and D. Terzopoulos, "Surveillance camera scheduling: A virtual vision approach," in *Proc. Third ACM Workshop on Video Surveillance and Sensor Networks (VSSN 05)*, Singapore, November 2005, pp. 131–139.

[45] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The Design and Implementation of a Wireless Video Surveillance System," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 426–438. [Online]. Available: https://doi.org/10.1145/2789168.2790123

[46] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[47] N. E. Nwogbaga, R. Latip, L. S. Affendey, and A. R. A. Rahiman, "Investigation into the Effect of Data Reduction in Offloadable Task for Distributed IoT-Fog-Cloud Computing," *J. Cloud Comput.*, vol. 10, no. 1, jul 2021. [Online]. Available: https://doi.org/10.1186/s13677-021-00254-6

[48] E. Šlapak, J. Gazda, G. Bugár, and T. Maksymyuk, "Review of Cellular Radio Network Cell Placement Design, From Traditional to Artificial Intelligence Based Approaches," in *2019 International Symposium ELMAR*, 2019, pp. 93–96.

[49] N. P. Kuruvatti, M. A. Habibi, S. Partani, B. Han, A. Fellan, and H. D. Schotten, "Empowering 6G Communication Systems With Digital Twin Technology: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 112 158–112 186, 2022.

[50] M. Yazdi and T. Bouwmans, "New Trends on Moving Object Detection in Video Images Captured by a Moving Camera: A Survey," *Computer Science Review*, vol. 28, pp. 157–177, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013716301794

[51] A. Downs, Z. Kootbally, W. Harrison, P. Pilliptchak, B. Antonishek, M. Aksu, C. Schlenoff, and S. K. Gupta, "Assessing Industrial Robot Agility Through International Competitions," *Robotics and Computer-Integrated Manufacturing*, vol. 70, p. 102113, 2021.

[52] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2

[53] H. Nyquist, "Certain Topics in Telegraph Transmission Theory," *Proceedings of the IEEE*, vol. 90, no. 2, pp. 280–305, 2002.

[54] "Implementation of the proposed system," https://github.com/Ericsson/ros2-3gppSA6-mapper, 2023.

[55] "Demo video of the proposed system in action," https://youtu.be/JXGAHvDSU4o, 2023.

[56] "Ten Agility Aspects in a Closed Loop Control Modelling," 2021. [Online]. Available: https://ieee-sa.imeetcentral.com/p/aQAAAAAEz9vp

[57] "FastDDS Filtering data on a Topic," May 2023. [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest/fastdds/dds_layer/topic/contentFilteredTopic/createContentFilteredTopic.html

[58] "ROS2 Design, About Quality of Service settings," May 2021. [Online]. Available: https://docs.ros.org/en/rolling/Concepts/About-Quality-of-Service-Settings.html

[59] "ROS2 topic hz," May 2022. [Online]. Available: https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html#ros2-topic-hz

[60] "ROS2, About topic statistics," May 2022. [Online]. Available: https://docs.ros.org/en/humble/Concepts/About-Topic-Statistics.html

[61] "ROS2 topic tools," Oct 2022. [Online]. Available: https://github.com/ros-tooling/topic_tools

[62] "eProsima Fast DDS Statistics Backend," Nov 2022. [Online]. Available: https://www.eprosima.com/index.php/products-all/tools/eprosima-fast-dds-statistics-backend

[63] "3GPP TS 33.521, 3.521 5G Security Assurance Specification (SCAS);Network Data Analytics Function (NWDAF)," June 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/33_series/33.521/33521-h20.zip

[64] "Ananya Muddukrishna: ROS2 Design, Unique Network Flows," May 2021. [Online]. Available: https://design.ros2.org/articles/unique_network_flows.html

[65] "Support for Unique Network Flows," May 2021. [Online]. Available: https://docs.ros.org/en/galactic/Releases/Release-Galactic-Geochelone.html#support-for-unique-network-flows

[66] "FastDDS DomainParticipantListener," Nov 2022. [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest/fastdds/dds_layer/domain/domainParticipantListener/domainParticipantListener.html

[67] "rclcpp::GenericPublisher Class Reference," May 2023. [Online]. Available: https://docs.ros2.org/galactic/api/rclcpp/classrclcpp_1_1GenericPublisher.html

[68] "rclcpp::GenericSubscription Class Reference," May 2023. [Online]. Available: https://docs.ros2.org/galactic/api/rclcpp/classrclcpp_1_1GenericSubscription.html

[69] "Turtlesim package from ros_tutorials repo," May 2022. [Online]. Available: https://index.ros.org/p/turtlesim/github-ros-ros_tutorials/

[70] "3GPP TS 23.203, Policy and charging control architecture," Dec 2021. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.203/23203-h20.zip

[71] "ROS2 Quality of Service demo," Jan 2023. [Online]. Available: https://github.com/ros2/demos/blob/master/quality_of_service_demo/rclpy/quality_of_service_demo_py/liveliness.py

[72] D. Minovski, C. Åhlund, K. Mitra, and P. Johansson, "Analysis and Estimation of Video QoE in Wireless Cellular Networks using Machine Learning," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, 2019, pp. 1–6.

[73] M. Eckert, T. M. Knoll, and F. Schlegel, "Advanced MOS Calculation for Network Based QoE Estimation of TCP Streamed Video Services," in *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2013, pp. 1–9.

[74] "chronyc," Nov 2022. [Online]. Available: https://chrony.tuxfamily.org/index.html

[75] "RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification," Jun 2010. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5905

[76] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica, "Ray RLLib: A Composable and Scalable Reinforcement Learning Library," *CoRR*, vol. abs/1712.09381, 2017. [Online]. Available: http://arxiv.org/abs/1712.09381

[77] "External Agents and Applications," Dec 2022. [Online]. Available: https://docs.ray.io/en/latest/rllib/rllib-env.html#external-agents-and-applications

[78] "Specifying Rollout Workers," Dec 2022. [Online]. Available: https://docs.ray.io/en/latest/rllib/rllib-training.html

[79] "OPC Unified Architecture," Aug 2023. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[80] "List of demo videos," Sep 2023. [Online]. Available: https://www.youtube.com/playlist?list=PLl1UXvs61i-GtR29ZgM2-GY9BYSGUeM34

[81] "Open source code base related to the dissertation," June 2024. [Online]. Available: https://github.com/EricssonResearch/opcua-3gppSA6-mapper

[82] "3GPP SA6 accelerates work on new verticals," Jun 2019. [Online]. Available: https://www.3gpp.org/news-events/3gpp-news/sa6-verticals

[83] "OPC Foundation 5.12.1 MonitoredItem model ," May 2023. [Online]. Available: https://reference.opcfoundation.org/Core/Part4/v104/docs/5.12.1

[84] "OPC Foundation 7.17.2 DataChangeFilter," May 2023. [Online]. Available: https://reference.opcfoundation.org/Core/Part4/v104/docs/7.17.2

[85] "open62541, Open Source OPC UA licensed under the MPL v2.0," June 2023. [Online]. Available: https://www.open62541.org/

[86] "OPC Foundation UA Part 22: 5.5.2 PriorityMappingTableType ," May 2023. [Online]. Available: https://reference.opcfoundation.org/Core/Part22/v105/docs/5.5.2

[87] "Free OPC UA Pub Sub C example," May 2023. [Online]. Available: https://github.com/open62541/open62541/blob/master/examples/pubsub/tutorial_pubsub_mqtt_publish.c

[88] "OPC Foundation 7.8 DiagnosticInfo," May 2023. [Online]. Available: https://reference.opcfoundation.org/Core/Part4/v104/docs/7.8

[89] "Free OPC UA Object IDs," May 2023. [Online]. Available: https://github.com/FreeOpcUa/freeopcua/blob/bd13aee2455c1d137bee45f7d2aedea8dd30115c/include/opc/ua/protocol/object_ids.h#L773

[90] "Mosquitto man page," May 2023. [Online]. Available: https://mosquitto.org/man/mosquitto-8.html

[91] "Mosquitto payload modification plugin," May 2023. [Online]. Available: https://github.com/eclipse/mosquitto/blob/master/plugins/payload-modification/mosquitto_payload_modification.c

[92] "MQTT Topic Statistics," May 2023. [Online]. Available: https://github.com/gambitcomminc/mqtt-stats

[93] "OPC UA Security: 6.6 Rate limiting and flow control," May 2023. [Online]. Available: https://reference.opcfoundation.org/Core/Part2/v105/docs/6.6

[94] "open62541 PubSub MQTT example," June 2023. [Online]. Available: https://github.com/open62541/open62541/blob/master/examples/pubsub/tutorial_pubsub_mqtt_publish.c

[95] "ONVIF Media Service Specification," Dec 2022. [Online]. Available: https://www.onvif.org/specs/srv/media/ONVIF-Media-Service-Spec.pdf

[96] "Dynamic Adaptive Streaming over HTTP," Apr 2012. [Online]. Available: https://www.iso.org/standard/57623.html

[97] A. Alidadi, S. Arab, and T. Askari, "A Novel Optimized Routing Algorithm for QoS Traffic Engineering in SDN-based Mobile Networks," *ICT Express*, vol. 8, no. 1, pp. 130–134, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959521001752

[98] C. Györgyi, K. Kecskeméti, P. Vörös, S. Laki, and G. Szabó, "In-Network Quality Control of IP Camera Streams," in *Accepted for publication in 6G-PDN (6G Programmable Deterministic Networking with AI) Workshop*, Oct 2023. [Online]. Available: preprint:https://drive.google.com/file/d/1-qGoC5k6AmATHvcIGnnV3Z-mNUutIvcS/view?usp=sharing

[99] "3GPP TS 23.222: Common API Framework for 3GPP Northbound APIs," Sep 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.222/23222-h70.zip

[100] "3GPP TS 29.222: Common API Framework for 3GPP Northbound APIs," Mar 2023. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/29_series/29.222/29222-h80.zip

[101] L. Silva and et al., "A Systematic Analysis of an Industrial Pickup and Placement Production System," in *5th EAI International Conference on Management of Manufacturing Systems*. Cham: Springer International Publishing, 2022, pp. 479–491.

[102] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-source Multi-robot Simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[103] A. Vidács and G. Szabó, "Winning ARIAC 2020 by KISSing The BEAR: Keeping things simple in Best Effort Agile Robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102166, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584521000508

[104] Y. Rubner, C. Tomasi, and L. Guibas, "A Metric for Distributions With Applications to Image Databases," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 59–66.

[105] "Cv2.emd method." [Online]. Available: https://shimat.github.io/opencvsharp_docs/html/980c9599-0cb2-a1cc-8322-850700753045.htm

[106] "P.800.1 : Mean Opinion Score (MOS) Terminology." [Online]. Available: https://www.itu.int/rec/T-REC-P.800.1-200303-S/en

[107] A. Vidács, Z. Trombitás, and G. Szabó, "Network Resource Management For Cyber-Physical Production Systems Based On Quality Of Experience," in *EMS 2023 37th European Simulation and Modelling Conference*, 2023, pp. 1–7.

[108] "Our demo video." [Online]. Available: https://youtu.be/5sMFZLwDU68

[109] T. Miao and L. Li, "Study on Influencing Factors of Sanding Efficiency of Abrasive Belts in Wood Materials Sanding," *Wood research*, vol. 59, pp. 835–842, 01 2014.

[110] B. Luo, L. Li, H. Liu, M. Xu, and F. Xing, "Analysis of Sanding Parameters, Sanding Force, Normal Force, Power Consumption, and Surface Roughness in Sanding Wood-Based Panels," *BioResources*, vol. 9, 10 2014.

[111] "Scan-N-Plan." [Online]. Available: https://rosindustrial.org/scan-n-plan

[112] "Multicalss Image Classifier in Keras." [Online]. Available: https://github.com/imamun93/animal-image-classifications/blob/master/final_notebook.ipynb

[113] "List of videos on automatic sanding." [Online]. Available: https://www.youtube.com/playlist?list=PLl1UXvs61i-G7EKnt_XhtDjdlvNUoKJYV

[114] "ros_additive_manufacturing," Feb 2021. [Online]. Available: https://gitlab.com/InstitutMaupertuis/ros_additive_manufacturing/

[115] I. A. Sucan and S. Chitta, "MoveIt!" http://moveit.ros.org/, 2013.

[116] "Gazebo Robot Simulator," http://gazebosim.org, 2023.

[117] U. Berdica, Y. Fu, Y. Liu, E. Angelidis, and C. Feng, "Mobile 3D Printing Robot Simulation with Viscoelastic Fluids," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7557–7563.

[118] "SPlisHSPlasH, Open-source Library for The Physically-based Simulation of Fluids," Jul 2023. [Online]. Available: https://github.com/InteractiveComputerGraphics/SPlisHSPlasH

[119] "M3DP-sim," Jul 2021. [Online]. Available: https://github.com/ai4ce/M3DP-Sim

[120] "URSim," Jul 2023. [Online]. Available: https://www.universal-robots.com/download/software-cb-series/simulator-non-linux/offline-simulator-cb-series-non-linux-ursim-3143/

[121] "ROS OPC UA Communication," Jul 2023. [Online]. Available: https://github.com/iirob/ros_opcua_communication/tree/noetic_support

[122] "UR Robot Driver," Jul 2023. [Online]. Available: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

[123] "The URScript Programming Language," https://s3-eu-west-1.amazonaws.com/ur-support-site/22198/scriptManual.pdf, 2019.

[124] "Show / manipulate traffic control settings," http://man7.org/linux/man-pages/man8/tc.8.html, 2023.

[125] J. H. Salim, "Linux Traffic Control Classifier-action Subsystem Architecture," *Proceedings of Netdev 0.1*, 2015.

[126] "iPerf," Aug 2023. [Online]. Available: https://iperf.fr/

[127] "Hector Trajectory Server," http://wiki.ros.org/hector_trajectory_server, 2017.

[128] J. Bender and D. Koschier, "Divergence-Free SPH for Incompressible and Viscous Fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1193–1206, 2017.

[129] "CloudCompare," Aug 2023. [Online]. Available: https://www.danielgm.net/cc/

[130] M. Sakin and Y. C. Kiroglu, "3D Printing of Buildings: Construction of the Sustainable Houses of the Future by BIM," *Energy Procedia*, vol. 134, pp. 702–711, 2017, sustainability in Energy and Buildings 2017: Proceedings of the Ninth KES International Conference, Chania, Greece, 5-7 July 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610217346969

[131] "Next-generation Simulation Technology To Accelerate The 5G Journey." [Online]. Available: https://github.com/rosin-project/automatica18_scan_and_plan_demo

Acronyms

3GPP Third Generation Partnership Project. iii, 4, 7–11, 33–36, 42, 44, 45, 59, 60, 65, 67–73, 78, 93–95, 99, 100, 104–108, 129

4G fourth generation of mobile communications. 42

5G fifth generation of mobile communications. i, 1, 4, 7–10, 14, 15, 18, 33, 34, 38, 39, 41, 42, 45, 57, 60, 65, 67, 68, 77, 78, 90, 103, 108, 109, 129, 130

5GC 5G Core. 8

5GS 5G System. 38

6G sixth generation of mobile communications. 18, 68, 108

ACIA Alliance for Connected Industries and Automation. 10, 90

AGV Automated Guided Vehicles. 80–82, 130

AI Artificial Intelligence. ii, 11, 14, 15, 18–32, 53, 54, 85, 103, 107, 129

API Application Programming Interface. 6–9, 36, 38, 39, 57, 65, 66, 69, 72–75, 93, 94, 100, 108

ARIAC Agile Robotics for Industrial Automation Competition. 11, 26–28, 30, 78–84, 108

BTS Base Transceiver Station. 17, 18

CAPIF Common API Framework. iv, 8, 59, 72, 73, 75, 93, 100, 107, 130

CPPS Cyber Physical Production System. 10, 79

CPS Cyber Physical System. 80, 83

CSP Communication Service Provider. 1, 8

CSR Customer Service Request. 1, 8

DASH Dynamic Adaptive Streaming Over HTTP. 48, 69

DDS Data Distribution Service. 3, 6, 9, 35, 37–39, 45

DPI Deep Packet Inspection. 37

DRAW Data Rate Averaging Window. 42, 44

DSCP Differentiated Services Code Points. 39

DT Digital Twin. 18, 19, 23, 59, 103

EMD Earth Mover's Distance. 80–82, 84, 88, 89, 130

EMOS Estimated Mean Opinion Score. v, 83, 84, 89, 106, 130

FOV Field of View. 17

FPS Frames Per Second. 48

FTP File Transfer Protocol. 4

GBR Guaranteed Bitrate. 44

GPS Global Positioning System. 51

GPU Graphics Processing Unit. 14, 16

GUI Graphical User Interface. 46, 90, 91, 106

HTTP HyperText Transfer Protocol. 57

IANA Internet Assigned Numbers Authority. 67

IAT Inter-Arrival Time. 35, 36, 60, 67

IEEE Institute of Electrical and Electronics Engineers. 34

IIoT Industrial Internet of Things. 1, 10, 34, 60

IO Input/Output. 13

IoT Internet of Things. 2, 5, 13, 15, 18

IP Internet Protocol. 38, 39, 41, 43, 57, 58, 65–67, 71, 104, 106

IT Information Technology. 1

JSON JavaScript Object Notation. 57, 67, 71

KPI Key Performance Indicator. ii–iv, 13, 35, 37, 46–49, 62–64, 72, 77–80, 83, 85, 87–89, 99, 109

LIDAR Laser Imaging, Detection, And Ranging. 13

LOC Lines of Code. iii, 56, 57

LoS Line of Sight. 16

MBB Mobile Broadband. 4, 77, 78, 108

MDBV Maximum Data Burst Volume. 42, 44

MIMO Multiple-Input Multiple-Output. 16

ML Machine Learning. 15, 16

MOS Mean Opinion Score. iii, v, 46, 47, 49, 52–54, 56, 77, 78, 83–85, 89, 90, 106–109, 129, 130

MQTT Message Queuing Telemetry Transport. iv, 3, 4, 6, 7, 10, 59, 61–68, 106

MTC Machine Type Communication. 33

MTU Maximum Transferable Unit. 44, 95, 96

NDT Networked Digital Twin. 92–94

NEF Network Exposure Function. 4, 8, 57

NF Network Function. 8, 49

NFE Network Flow Endpoint. 39

NPN Non-Public Network. 1

NRM Network Resource Management. i, iv, 1–8, 11, 14–16, 38, 41, 44, 45, 48, 52, 57–59, 64, 65, 68–76, 84, 90, 92–94, 99, 100, 103, 105, 107, 108, 130

NSA Non-Standalone. 57

NTP Network Time Protocol. 51

NWDAF Network Data Analytics Function. 38

OM Operations and Maintenance. 8

ONVIF Open Network Video Interface Forum. 69

OPC Open Platform Communications. i, iii, v, 3, 6, 7, 9–11, 59–62, 64–66, 68, 71, 74, 75, 90, 92–94, 96, 98–102, 105, 106, 110, 130

OS Operating System. 68–71, 108

OT Operational Technology. 1, 10

PDB Packet Delay Budget. 42, 44

PELR Packet Error Loss Rate. 42, 44

PID Proportional-Integral-Derivative. 47

PL Priority Level. 42

PLC Programmable Logic Controller. 13

PPO Proximal Policy Optimization. 53, 54

QCI QoS Class Identifiers. 4, 7, 42–44, 52, 57, 67, 68, 105, 106

QoE Quality of Experience. 38, 72, 77, 78, 83, 85, 89, 98, 106, 108

QoS Quality of Service. iii, iv, 3–7, 9, 20–22, 28–32, 34, 35, 37–49, 51–53, 56–58, 61–63, 65–68, 70, 71, 73, 77, 78, 80, 84, 85, 87, 88, 94, 100, 104–106, 108, 130

QoSensing Quality of Sensing. 13

RAM ROS Additive Manufacturing. 90–93, 95

RAN Radio Access Network. 17, 18

REST Representational State Transfer. 7

RFC Request for Comments. 51

RL Reinforcement Learning. 14, 16, 19, 21–23, 27, 53, 103, 106, 107

RMW ROS Middleware. 39

ROS Robot Operating System. iii, v, 3, 5, 7, 9, 11, 33–42, 44–50, 52, 53, 57–59, 66, 74, 86, 90–94, 99–101, 105, 107, 129

RT Resource Type. 42, 44

SA System Aspect. 8, 34–36, 60, 78, 129

SBA Service-Based Architecture. 8

SBI Service-Based Interface. 8

SDN Software Defined Networking. 69

SEAL Service Enabler Architecture Layer. iii, iv, 4, 7–9, 11, 33, 38, 45, 46, 48, 49, 56, 57, 59, 65, 66, 69, 73, 90, 94, 99, 100, 104–106, 129, 130

TCP Transmission Control Protocol. 9, 35, 39, 62, 65, 66, 94, 96, 99

TPT Total Processing Time. 83, 84

TS Technical Specifications. 4, 8, 9, 38, 41, 42, 44, 45, 48, 56, 57, 65, 67–70, 72, 73, 100, 107, 108

TSG Technical Specification Group. 8

UA Unified Architecture. i, iii, v, 3, 6, 7, 9–11, 59–62, 64–66, 68, 71, 74, 75, 90, 92–94, 96, 98–102, 105, 106, 110, 130

UDP User Datagram Protocol. 9, 35, 39, 41, 71

UE User Equipment. 2, 4, 5, 14, 41, 57, 58, 65, 67–72, 103, 108

UFTR Unique Flow Topic Relay. 37, 42, 46, 47, 49–54, 57, 58, 105, 107, 129

UR Universal Robot. 92–94, 100

URL Uniform Resource Locator. 73

VAL Vertical Application Layer. iv, 2, 11, 41, 45, 46, 48, 49, 54, 57, 68–74, 76, 90, 92, 100, 104–108

VASEAL Vertical Application Service Enabler Architecture Layer. iii, 45, 46, 48, 49, 106

WG Working Group. 8

WiFi Wireless Fidelity. 17

List of Figures