Towards Edge Robotics: The progress from Cloud-based robotic systems to intelligent and context-aware robotic services

by

Milan Groshev

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

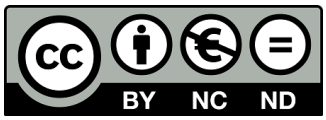
Telematic Engineering

Universidad Carlos III de Madrid

Advisor:

Dr. Antonio de la Oliva

September 2022



"You've got to just keep on pushing
Keep on pushing
Push the sky away."
— Nick Cave

"What's love got to do with what you got?
Don't let a win get to your head or a loss to your heart"
— Public Enemy

"How long should I stay dedicated?
How long 'til opportunity meet preparation?"
— Nipsey Hussle

Acknowledgements

After my years pursuing my PhD, I would like to express my sincere gratitude to all the people that were involved in the process. First of all, I would like to thank my supervisor **Antonio de la Oliva** who gave me the opportunity for doing this PhD. Thank you for the complete trust and full support during the PhD, and the experience that I gained through participation in European research projects and collaborations with top European research institutions such as NEC, Politecnico di Torino, and CTTC. A special graduate also goes to **Carlos Jesus Bernardos** for his wise advice and constructively critical comments that served for improving the quality of each of our research works. The combination of Antonio's directness and fast reaction with Carlos's patience and calmness allowed me to grow as a researcher over the last 4 years. Thank you very much for everything.

Second, I would like to thank all my PhD and post-doc colleagues from UC3M for the unlimited support over all these years. Special gratitude goes to **Luca Cominardi**, **Carlos Guimarães**, **Jorge Martín-Pérez**, and **Kiril Antevski** who helped me and supported me in every stage of my PhD. From brainstorming research ideas to hands-on assistance in the laboratory and scientific writing. Thanks to your patience, dedication, kindness and criticism i learned a lot of research tools and methods that made me a better researcher. I would also like to thank my colleagues: **Borja**, **Victor**, **Sergio**, **Winnie**, **Nuria**, **Stefano**, **Pedro**, **Miguel**, **Gines**, **Marco**, **Jesus**, **Luis**, **Dulce**, **Raul** and **Gonzalo** for the moments we shared all this 4 years and the support that they gave me during our lunch time, coffee breaks and other moments. Third, i would like to thank all my close friend that were always here for me, in good and bad, providing incredible support and allowing me to completely disconnect from the research. Thanks you for the all the good moments during this years. Without you this thesis would have been impossible. I would also like to thank all my basketball teammates throughout this 4 years that always gave me additional value, motivation and strength to keep moving forward. Thank you for the many great moments, victories and trophies.

Last but not least, i would like to dedicate this work to my family, especially my parents **Tanja** and **Gradimir**, who from day one provided me with unconditional love, understanding, advice and confidence that made this whole journey to be easier. Thank you and i love you. Special thanks to my sister **Nina** for her endless support in everything i do. She is a bright young researcher that always challenged me to do more and more.

Published Content

In compliance with the principles referring to plagiarism in Law 14/2011 and in the Code of Good Practices of the Universidad Carlos III de Madrid (UC3M) Doctoral School, I hereby report a bibliography of articles or other contributions I have (co)authored that are included as part of the thesis and that have been published.

Articles published in indexed journals and magazines:

- **M. Groshev**, G. Baldoni, L. Cominardi, A. de la Oliva, and R. Gazda, "Edge robotics: are we ready? An experimental evaluation of current vision and future," *Digital Communications and Networks*, 2022, Accepted for publication in August 2021. Waiting for polishing office to generate the proofe read.

  - This work is partially included in Chapter 2 and Chapter 3.
  - The role of the author of this thesis was to study the related work with respect to the integration of edge computing and fog computing for robotics systems, design the end-to-end edge robotics system, implement a prototype of the designed system for mobile robots, execute the experimental analysis and analyze the obtained results.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- **M. Groshev**, C. Guimarães, A. De La Oliva, and R. Gazda, "Dissecting the impact of information and communication technologies on digital twins as a service," *IEEE Access*, vol. 9, pp. 102 862–102 876, 2021. DOI: 10.1109/ACCESS.2021.3098109.

  - This work is partially included in Chapter 2 and Chapter 3.
  - The role of the author of this thesis was to analyze the role of the recent advances in ICT in industrial Digital Twin systems, design an exemplary edge robotic digital twin, implement a prototype of the edge robotic digital twin, execute the experimental study and analyze the obtained results.

  - The material from this source included in this thesis is not singled out with typographic means and references.

- **M. Groshev**, C. Guimarães, J. Martín-Pérez, and A. de la Oliva, "Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 14–20, 2021. DOI: 10.1109/MCOM.001.2001237.

  - This work is partially included in Chapter 5.
  - The role of the author of this thesis was to define the existing open challenges that can be tackled by AI in existing industrial digital twin systems, assist in the definition of the exemplary AI agents and implement the proof of concept showcasing the movement prediction.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- **M. Groshev**, J. Martín-Pérez, C. Guimarães, A. de la Oliva, and C. J. Bernardos, "FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators," *IEEE Transactions on Network and Service Management*, 2022, Submitted for publication in November 2021. Under minor revision.

  - This work is included in Chapter 5
  - The role of the author of this thesis was to participate in the study the related work, design of the solution, implementation of the testbed, collect the dataset and execute the experimental evaluation.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- J. Martín-Pérez, F. Malandrino, C. F. Chiasserini, **M. Groshev**, and C. J. Bernardos, "Kpi guarantees in network slicing," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2021. DOI: 10.1109/TNET.2021.3120318.

  - This work is partially included in Chapter 4.
  - The role of the author of this thesis was to design, implement and execute the testbed and experimental validation.
  - The material from this source included in this thesis is not singled out with typographic means and references.

Articles published in conferences and workshops:

- **M. Groshev**, J. Martín-Pérez, K. Antevski, A. de la Oliva, and C. J. Bernardos, "Cotorra: Context-aware testbed for robotic applications," in *Proceedings of the 1st Workshop on Serverless Mobile Networking for 6G Communications*, ser. MobileServerless'21, Virtual, WI, USA: Association for Computing Machinery, 2021, 7–12. DOI: 10.1145/3469263.3469857.
  - This work is partially included in Chapter 2.
  - The role of the author of this thesis has been to contribute into the design of COTTORA architecture and development, implementation and experimental evaluation of COTTORA.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- K. Antevski, **M. Groshev**, G. Baldoni, and C. J. Bernardos, "DLT federation for Edge robotics," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2020, pp. 71–76. DOI: 10.1109/NFV-SDN50289.2020.9289887.
  - This work is partially included in Chapter 4 and Chapter 1.
  - The role of the author of this thesis was to design the edge robotics service, implement the edge robotics related part of the experimental testbed and assist in the execution of the experiments.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- K. Antevski, **M. Groshev**, L. Cominardi, C. J. Bernardos, A. Mourad, and R. Gazda, "Enhancing edge robotics through the use of context information," in *Proceedings of the Workshop on Experimentation and Measurements in 5G*, ser. EM-5G'18, Heraklion, Greece: Association for Computing Machinery, 2018, 7–12. DOI: 10.1145/3286680.3286682. [Online]. Available: https://doi.org/10.1145/3286680.3286682.
  - This work is partially included in Chapter 2 and Chapter 3.
  - The role of the author of this thesis was to implement and execute the experimental analysis. In addition, the author assisted in the edge robotics system design.
  - The material from this source included in this thesis is not singled out with typographic means and references.

- L. Girletti, **M. Groshev**, C. Guimarães, C. J. Bernardos, and A. de la Oliva, "An intelligent edge-based digital twin for robotics," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6. DOI: 10.1109/GCWkshps50303.2020.9367549.
  - This work is partially included in Chapter 3.

– The role of the author of this thesis was to assist in the design the intelligent edge-based digital twin for robotics and assist in the implementation and execution of the experimental analysis.
– The material from this source included in this thesis is not singled out with typographic means and references.

Other Research Merits

This chapter first provides a list of additional publications I have (co)authored, which are related to this thesis but not included in the document. Next, it provides an overview of the various European funded projects where I was involved in during the lifetime of this thesis, as well as my role and participation. Finally, it includes open-source contributions and demonstrations that were accomplished under the scope of this thesis.

Related publications and demonstrations:

- C. Guimarães, **M. Groshev**, L. Cominardi, A. Zabala, L. M. Contreras, S. T. Talat, C. Zhang, S. Hazra, A. Mourad, and A. de la Oliva, "Deep: A vertical-oriented intelligent and automated platform for the edge and fog," *IEEE Communications Magazine*, vol. 59, no. 6, pp. 66–72, 2021. DOI: 10.1109/MCOM.001.2000986.
  - The role of the author of this thesis was to contribute in the DEEP concept overview content and to integrate the Digital Twin service for the experimental evaluation of the DEEP.
  - The material from this source is not included in this thesis.
- G. Baldoni, L. Cominardi, **M. Groshev**, A. De la Oliva, and A. Corsaro, "Managing the far-edge: Are today's centralized solutions a good fit," *IEEE Consumer Electronics Magazine*, pp. 1–1, 2021. DOI: 10.1109/MCE.2021.3082503.
  - The role of the author of this thesis was to contribute in the analysis of the infrastructure differences and management solutions of the cloud and the edge.
  - The material from this source is not included in this thesis.
- F. Conceição, C. Guimarães, L. Cominardi, S. T. Talat, M. F. Ardiansvah, C. Zhang, **M. Groshev**, T. William, G. Patra, I. Hemadeh, C. Lu, and A. Mourad, "Empowering industry 4.0 and autonomous drone scouting use cases through 5g-dive solution," in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2021, pp. 265–270. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482590.

  - The role of the author of this thesis was to contribute in identifying the R&D challenges and experimental results related to the Digital Twin use case.
  - The material from this source is not included in this thesis.

- **M. Groshev** and C. Guimarães, "Assessing the need for 5g driven edge and fog solution for digital twin systems," in *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, ser. WiNTECH'20, London, United Kingdom: Association for Computing Machinery, 2020, 126–127. DOI: 10.1145/3411276.3414697.
  - The role of the author of this thesis was to design, implement and execute the Digital Twin demonstration.
  - The material from this source is not included in this thesis.

- J. Baranda, J. Mangues-Bafalluy, E. Zeydan, C. Casetti, C. F. Chiasserini, M. Malinverno, C. Puligheddu, **M. Groshev**, C. Guimarães, K. Tomakh, D. Kucherenko, and O. Kolodiazhnyi, "Demo: Aiml-as-a-service for sla management of a digital twin virtual network service," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–2. DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484610.
  - The role of the author of this thesis was to provide the Digital Twin service and help the integration of the service in the AI/ML platform (5Gr-AIMLP).
  - The material from this source is not included in this thesis.

Participation and role in European research projects:

European H2020 5G–CORAL Project

5G-CORAL project leverages on the pervasiveness of edge and fog computing in the Radio Access Network (RAN) to create a unique opportunity for access convergence. This is envisioned by the means of an integrated and virtualised networking and computing solution where virtualised functions, context-aware services, and user and third-party applications are blended together to offer enhanced connectivity and better quality of experience. The proposed solution contemplates two major building blocks, namely *i)* the Edge and Fog computing System (EFS) subsuming all the edge and fog computing substrate offered as a shared hosting environment for virtualised functions, services, and applications; and *ii)* the Orchestration and Control System (OCS) responsible for managing and controlling the EFS, including its interworking with other (non-EFS) domains.

This project was coordinated by UC3M and the role and the activities of the author of this thesis in the project are the following:

- Actively participating in the project since July 2018 until the end of the project (August 2019).
- Collaborated in the coordination of the project.

– Mainly involved in WP2,WP3,WP4: design of the 5G-Coral architecture; desing of the fog-assisted robotics use case.
– Fog-assisted robotics use case owner: driving the design, development, integration, execution and results reporting of the use case.
– Two demonstrations of the of the fog-assisted robotics demo.
– Contribution and editorial roles in several. deliverables[1]: (D3.2, D4.2, D5.2)

European H2020 5G–DIVE Project

5G-DIVE aims at providing a higher-level of abstraction to its customers (e.g., the verticals) by providing a set of supporting strata that would enable enhanced business automation and ease the provisioning of intelligence capabilities into the vertical services. In doing so, 5G-DIVE positions itself on top of Edge Computing Infrastructures, allowing the shift from an Infrastructure-as-a-Service (IasS) service model towards an end-toend Platform-as-a-Service (PaaS) service model. The above concept is materialized in a new building block called 5G-DIVE Elastic Edge Platform (DEEP), which spans as an add-on on-top of existing Edge Computing Infrastructures while underpinning vertical industries OSS/BSS systems. The DEEP building block envisions three supporting stratas: *i)* Data Analytics Support Stratum (DASS);*ii* Intelligence Engine Support Stratum (IESS); and *iii* Business Automation Support Stratum (BASS). This strats offer unique capabilities tailored to the support of the vertical industries OSS/BSS systems with the goal of enhancing day-by-day operations and business processes.

This project was coordinated by UC3M and the role and the activities of the author of this thesis in the project are the following:

– Actively participating in the project since the start of the project (October 2019) until the end of the project (December 2021).
– Collaborated in the coordination of the project.
– Mainly involved in WP1,WP2,WP3: design and refinement of the DEEP architecture; design of the Digital Twin use case.
– Digital Twin use case owner: driving the design, development, integration, execution and results reporting of the use case
– Two demonstrations of the Digital Twin demo
– Contribution and editorial roles in several deliverables[2]: (D1.1, D1.3, D1.4, D2.1, D2.2, D2.3, D2.4, D3.1, D3.2, D3.3)

European H2020 5GROWTH Project

The 5Growth project's objective is to improve vertical industries like Industry 4.0, Transportation, and Energy with a 5G End-to-End Solution that is AI-driven, Automated, and Shareable, enabling them to

[1] http://5g-coral.eu/index.php/deliverables/ [Accessed: Jul. 14, 2022]
[2] https://5g-dive.eu/?page_id=26 [Accessed:Jul. 14, 2022]

effortlessly meet the key performance targets. Different vertical industries are supported by 5Growth via a simplistic graphical user interface (GUI) that receives verticals service requirements and translates them to deploy the respective network slices on top of 5Growth or other 5G End-to-End platforms. In addition, the 5Growth platform provides closed-loop automation and SLA control for the vertical services lifecycle management and AI driven solutions to optimize the Access, Transport, Core and Fog, Edge and Cloud resources across multiple domains.

The role and the activities of the author of this thesis in the project are the following:

- Side participation in the project since September 2020 to May 2022;
- Integration of a Digital Twin network service in the 5Growth platform with a goal of studding the Digital Twin service scalability using AI/ML.

Open-source projects:

The work performed in this thesis resulted in the creation of two open-source projects that aim to serve as network services for the networking research community, where researchers can use them to validate their innovative concepts.

Niryo One digital twin for robot manipulators
The Niryo One Digital Twin is a network service of an operational Digital Twin for robot manipulators. The network service uses the Niryo One ROS stack[3] to create simulated robot manipulator and CoppeliaSim[4] to create the virtual replica of the robot. The user of the network service can use or develop remote controllers (e.g., ROS python API, Joystick, web interface) to control the robot manipulator via the Digital Twin. The network service was created by the author as part of the 5G-DIVE project and in this thesis was used to implement the robot manipulator related NFVs used in Chapter 2, Chapter 3 and Chapter 5. As such, this networks service can be used with the real Niryo One robot manipulator or in a only-simulated environment where a simulated version of the robot manipulator is available.

- **Title:** niryo-one-digital-twin.
- **Used technologies:** Docker, ROS, CoppeliaSim, Python, HTML, Docker compose.
- **URL:** https://github.com/milangroshev/niryo-one-digital-twin.

Turtlebot2 autonomous navigation and mapping for mobile robots
The Turtlebot2 autonomous navigation is a network service for mobile robots that performs mapping and/or autonomous robot teleoperation. This network service virtualizes the Turtelbot2 drivers[5], its

[3] https://github.com/NiryoRobotics/niryo_one_ros/ [Accessed:Jul. 14, 2022]
[4] https://www.coppeliarobotics.com/ [Accessed:Jul. 14, 2022]
[5] https://github.com/turtlebot/turtlebot [Accessed: Jul. 14, 2022]

sensors[6] (e.g., lidar, camera) and 2D ROS navigation stack[7] in order to allow autonomous navigation of the robot and map creation. The user can interact with the navigation stack using a GUI or by developing a custom Python ROS script that defines the robot trajectory. The network service was created by the author of this thesis to implement the mobile robot related NFVs used in Chapter 2, Chapter 3, Chapter 4 and Chapter 5. As such, this networks service can be used with the real Turtlebot2 mobile robot or in a only-simulated environment where a simulated version of the robot drivers is available.

- **Title:** turtlebot2-autonomous-navigation.
- **Used technologies:** Docker, ROS.
- **URL:** https://github.com/milangroshev/turtlebot2-service.

Demonstrations:

Edge Robotics: Synchronized Delivery
This demonstration validates the concept of edge robotics by showcasing an edge robotic system for mobile robots that performs cooperative delivery of large items in shopping malls. The demo scenario is based on delivering goods/products from storage rooms to the shops in the shopping mall. Since large items need multiple robots to carry them to a shop, the idea of the demo is to show synchronous cooperation between two robots, carrying a single item. At start the robots are in idle state (maintaining the network connection) located in the storage room. An employee manually initiates the transportation of a large items from the storage room to a shop. Two robots form a fleet using Wi-Fi technology. The Wi-Fi would be used for infrastructure-to-robot communication assisting the robots for navigation and Wi-Fi Direct is used for robot-to-robot connection. The Wi-Fi Direct connection would be established for robot-to-robot or Device-to-Device (D2D) communication for low-latency connection and maintaining better coordination (e.g., moving in formation). Upon establishing the fleet (formation) the large item is loaded on the robot. The robots deliver the item to the destination while maintaining the formation. Once the item is delivered, the robots break the formation and navigate back to the start location. This demonstration validates the feasibility of the edge-based mobile robot system that is designed in Chapter 2 and some of the context-aware features for robotic systems that are presented in Chapter 3.

- **Location:** Taipei, Taiwan
- **Project:** 5G-Coral
- **Date:** November 2018
- **URL:** https://youtu.be/YX5UmNBkFqc
- **Author role**: Development and integration of the synchronize delivery robot application

[6] https://github.com/Slamtec/rplidar_ros [Accessed: Jul. 14, 2022]
[7] https://github.com/ros-planning/navigation [Accessed: Jul. 14, 2022]

5G-CORAL: Fog Assisted Robotics Demo

This demonstration extends the edge robotics concept for mobile robots by introducing the features of robot re-usability, D2D coordination and virtual access point offloading. The demo scenario is based on cleaning service and a product delivery service in an business center. At start of the demonstration the robots are in idle state (maintaining the network connection) located in the one of the floors of the business center. An building operation manager manually initiates the cleaning service and both robots start to clean the office hallway floor independently, each one navigated from the intelligence that resides in the cloud. During the execution of the cleaning service, an employee requests a delivery of a specific HW to his office. The two robots that execute the cleaning service receive the request and establish a formation. Upon establishing the formation, the robot intelligence is offloaded from the cloud to the fog in order to support the latency sensitive coordinated delivery. The package is loaded on top of the robots and network-assisted D2D communication channel is established between the robots in order to improve the coordinated movements. The robots start the delivery of the package with synchronized driving where the first robot is autonomously navigated from the intelligence that resides in the fog and the second robot is navigated with the information that is consumed via the D2D link. As the robots drive towards the destination, based on the network context information (Wi-Fi signal strength), the platform can detect that the robots are moving away from the AP and trigger the migration of the virtual access point. Once the package is delivered, the robots break the formation and return to the start positions navigated by the intelligence that resides in the fog. This demonstration validates the feasibility of the edge-based mobile robot system that is designed in Chapter 2, the D2D and virtual AP offloading features for robotic systems that are presented in Chapter 3.

- **Location:** Taipei, Taiwan
- **Project:** 5G-Coral
- **Date:** October 2019
- **URL:** https://youtu.be/zzjxDSLGdas
- **Author role**: Development and integration of the synchronize delivery robot application, development of the network-assisted D2D feature, development of the virtual access point offloading feature and presentation of the demonstration

I4.0: Edge-assisted Digital Twin over 5G

This demonstration showcases a 5G solution that integrates 5G connectivity and edge computing in operational Digital Twin systems. The demo scenario is based on remotely controlled robot manipulator and its digital twin. Since robot manipulators usually need to execute high precision tasks in the factory floor, the idea of the demo is to show the synchronization and real-time remote control between a robot manipulator and its digital twin, executing a pick-and-place task. A remote operator executes the pick-and-place of an box using 5G and 4G connectivity where the observant can notice the precision and speed that 5G can bring to remote control operational digital twins with respect to 4G. In addition, a replay feature is demonstrated that allows the remote operator to request replay of a given robot movement sequence from the past in a separate virtual replica in order to detect robot

misbehaviour. This demonstration validates the feasibility of the edge-based robot manipulator system that is designed in Chapter 2.

- **Location:** Madrid, Spain
- **Project:** 5G-DIVE
- **Date:** December 2020
- **URL:** https://youtu.be/b9jme58dLt8
- **Author role**: Development of the Digital Twin prototype, integration of the prototype with the 5G DIVE platform and presentation of the demonstration.

Digital Twin movement prediction

This demonstration presents movement prediction, a feature to recover lost control commands in remotely controlled operational digital twins. In the demonstration, a robotic arm is remotely controlled to perform pick-and-place task under the presence of packet losses in the wireless medium. The lost control commands result in a distorted trajectory of the robotic arm, thus, we deploy the movement prediction feature to recover lost control commands using an ML model that we train with a real-world dataset. The demonstration shows how the movement prediction recovers the lost commands, and how the robot arm operates smoothly despite the losses that are present in the wireless medium.. This demonstration validates the feasibility of the edge-based robot manipulator system that is designed in Chapter 2 and the FoReCo solution that is presented in Chapter 5.

- **Location:** Madrid, Spain
- **Project:** 5G-DIVE
- **Date:** March 2022
- **URL:** https://youtu.be/eYzXfx6uBJE
- **Author role**: Integration of the movement prediction in the Digital Twin prototype and the 5G DIVE platform, integration of influxDB GUI to visualize the control commands and presentation of the demonstration.

Abstract

Current robotic systems handle a different range of applications such as video surveillance, delivery of goods, cleaning, material handling, assembly, painting, or pick and place services. These systems have been embraced not only by the general population but also by the vertical industries to help them in performing daily activities. Traditionally, the robotic systems have been deployed in standalone robots that were exclusively dedicated to performing a specific task such as cleaning the floor in indoor environments. In recent years, cloud providers started to offer their infrastructures to robotic systems for offloading some of the robot's functions. This ultimate form of the distributed robotic system was first introduced 10 years ago as cloud robotics and nowadays a lot of robotic solutions are appearing in this form. As a result, standalone robots became software-enhanced objects with increased reconfigurability as well as decreased complexity and cost. Moreover, by offloading the heavy processing from the robot to the cloud, it is easier to share services and information from various robots or agents to achieve better cooperation and coordination.

Cloud robotics is suitable for human-scale responsive and delay-tolerant robotic functionalities (e.g., monitoring, predictive maintenance). However, there is a whole set of real-time robotic applications (e.g., remote control, motion planning, autonomous navigation) that can not be executed with cloud robotics solutions, mainly because cloud facilities traditionally reside far away from the robots. While the cloud providers can ensure certain performance in their infrastructure, very little can be ensured in the network between the robots and the cloud, especially in the last hop where wireless radio access networks are involved. Over the last years advances in edge computing, fog computing, 5G NR, network slicing, Network Function Virtualization (NFV), and network orchestration are stimulating the interest of the industrial sector to satisfy the stringent and real-time requirements of their applications. Robotic systems are a key piece in the industrial digital transformation and their benefits are very well studied in the literature. However, designing and implementing a robotic system that integrates all the emerging technologies and meets the connectivity requirements (e.g., latency, reliability) is an ambitious task.

This thesis studies the integration of modern Information and Communication Technologies (ICTs) in robotic systems and proposes some robotic enhancements that tackle the real-time constraints of robotic services. To evaluate the performance of the proposed enhancements, this thesis departs from the design and prototype implementation of an edge native robotic system that embodies the

concepts of edge computing, fog computing, orchestration, and virtualization. The proposed edge robotics system serves to represent two exemplary robotic applications. In particular, autonomous navigation of mobile robots and remote-control of robot manipulator where the end-to-end robotic system is distributed between the robots and the edge server. The open-source prototype implementation of the designed edge native robotic system resulted in the creation of two real-world testbeds that are used in this thesis as a baseline scenario for the evaluation of new innovative solutions in robotic systems.

After detailing the design and prototype implementation of the end-to-end edge native robotic system, this thesis proposes several enhancements that can be offered to robotic systems by adapting the concept of edge computing via the Multi-Access Edge Computing (MEC) framework. First, it proposes exemplary network context-aware enhancements in which the real-time information about robot connectivity and location can be used to dynamically adapt the end-to-end system behavior to the actual status of the communication (e.g., radio channel). Three different exemplary context-aware enhancements are proposed that aim to optimize the end-to-end edge native robotic system. Later, the thesis studies the capability of the edge native robotic system to offer potential savings by means of computation offloading for robot manipulators in different deployment configurations. Further, the impact of different wireless channels (e.g., 5G, 4G and Wi-Fi) to support the data exchange between a robot manipulator and its remote controller are assessed.

In the following part of the thesis, the focus is set on how orchestration solutions can support mobile robot systems to make high quality decisions. The application of OKpi as an orchestration algorithm and DLT-based federation are studied to meet the KPIs that autonomously controlled mobile robots have in order to provide uninterrupted connectivity over the radio access network. The elaborated solutions present high compatibility with the designed edge robotics system where the robot driving range is extended without any interruption of the end-to-end edge robotics service. While the DLT-based federation extends the robot driving range by deploying access point extension on top of external domain infrastructure, OKpi selects the most suitable access point and computing resource in the cloud-to-thing continuum in order to fulfill the latency requirements of autonomously controlled mobile robots.

To conclude the thesis the focus is set on how robotic systems can improve their performance by leveraging Artificial Intelligence (AI) and Machine Learning (ML) algorithms to generate smart decisions. To do so, the edge native robotic system is presented as a true embodiment of a Cyber-Physical System (CPS) in Industry 4.0, showing the mission of AI in such concept. It presents the key enabling technologies of the edge robotic system such as edge, fog, and 5G, where the physical processes are integrated with computing and network domains. The role of AI in each technology domain is identified by analyzing a set of AI agents at the application and infrastructure level. In the last part of the thesis, the movement prediction is selected to study the feasibility of applying a forecast-based recovery mechanism for real-time remote control of robotic manipulators (FoReCo) that uses ML to infer lost commands caused by interference in the wireless channel. The obtained results are showcasing the its potential in simulation and real-world experimentation.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Symbols

3GPP 3th Generation Partnership Project

4G 4^{th} Generation of Mobile Networks

5G 5^{th} Generation of Mobile Networks

5G-PPP 5G Infrastructure Public Private Partnership

6G 6^{th} Generation of Mobile Networks

A

AD Administrative Domain

AI Artificial Intelligence

AP Access Point

API Application Programming Interface

AR Augmented Reality

ARIMA Autoregressive Integrated Moving Average

AVG Autonomous Guided Vehicle

B

BN Bayesian Network

BSS Business Support System

BSSID Basic Service Set Identifier

C

CDF Cumulative Density Function

CFS Customer Faceting Service

CNN Convolutional Neural Networks

CPE Customer Premises Equipment

CPS Cyber-Physical System

CPU Central Processing Unit

CSMA/CA Carrier-sense multiple access with collision avoidance

D

D2D Device to Device

DC Data Center

DDS Data Distribution Service

DLT Distributed Ledger Technology

DNS Domain Name System

DS Distributed System

E

E2E End-to-End

EM Electromagnetic

eMBB enhanced Mobile Broadband

eMTC enhanced Machine Type Communication

ETSI European Telecommunications Standards Institute

G

GRU Gated Recurrent Unit

I

ICT Information and Communication Technologies

IEEE Institute of Electrical and Electronics Engineers

IIC Industrial Internet Consortium

IL Imitation Learning

IoT Internet of Things

IT Information Technology

ITU International Telecommunications Union

K

KPI Key Performance Indicator

KVM Kernel-based Virtual Machine

L

LR Logistic Regression

LSTM Long-Short Term Memory

LXD Linux Container Daemon

M

MA Moving Average

MAC Media Access Control

MANO Management & Orchestration

MEC Multi-access Edge Computing

MEM Memory

MEP Multi-access Edge Platform

ML Machine Learning

MLP Multi-Layer Perceptron

mMTC massive Machine Type Communications

MQTT Message Queuing Telemetry Transport

N

NB Narrowband

NFV Network Functions Virtualization

NFVI NFV Infrastructure

NFVO NFV Orchestrator

NR New Radio

NS Network Service

NSA non-standalone

O

OSS Operations Support System

OT Operational Technology

P

PCA Principal Component Analysis

PDF Probability Density Function

PID Proportional-Integral-Derivative

PoA Proof-of-Authority

PoW Proof-of-Work

Q

QoE Quality of Experience

QoS Quality of Service

R

RAN Radio Access Network

RAT Radio Access Technology

REST Representational State Transfer

RF Random Forest

RL Reinforcement Learning

RMSE Root-mean-square error

RNIS Radio Network Information Service

ROS Robot Operating System

RT Random Tree

RTX re-transmissions

S

SA standalone

SC Smart Contract

seq2seq Sequence to sequence

SLA Service Level Agreement

SoA State of the Art

SSID Service Set Identifier

SVM Support Vector Machines

T

TCN Temporal Convolutional Networks

U

UC3M Universidad Carlos III de Madrid

UE User Equipment

UPF User Plane Function

URLLC Ultra Reliable Low Latency Communications

V

VAR Vector Autoregressive

VIM Virtualized Infrastructure Manager

VL Virtual Link

VM Virtual Machine

VNE Virtual Network Embedding

VNF Virtual Network Function

VR Virtual Reality

W

WAIS WLAN Access Information Service

WLAN Wireless Local Area Network

CHAPTER 1

Introduction

Robotic systems are devices and robots with three main functions: sensing, actuation, and control. As such, autonomous robotic systems are serving the manufacturing industry for forty years with a focus on applications like spot and arc welding, painting, material handling, and pick and place of objects. These industrial robotic systems have always been controlled and monitored by the industrial workers where due to strict industrial safety policies the robot and human workspaces were always separated. However, to enable efficient and optimized collaboration between workers and robots, these barriers need to be eliminated [1].

Providing robotic services to support industrial activities through interactions with humans or machines is an emerging topic in robotics research [1]. For the last decade, many studies are investigating how to develop mission-critical robot systems that can be useful in industrial scenarios and how they can behave in natural and trustworthy ways so that the industrial workers can intuitively work with them in close proximity. One example application of collaborative robots, or cobots is a robotic arm that supports workers in handling heavy parts when loading machines or pick components out of bins. Cognitive robots for knowledge-intensive industrial work are another popular application of mission-critical robotic systems. Although the applicability of cobots on the industrial floor remains limited, their ability to perform daily industrial tasks is rapidly improving.

With such improvement, the construction of robots is becoming increasingly complex. New sensors, actuators, and algorithms are being developed such as intelligent navigation and manipulation, human detection, and speech recognition. An example of these improvements are Spot [2] and Atlas [3] agile robots of Boston Dynamics, that can navigate industrial floors with unprecedented mo-

bility or achieve human-level agility. In addition to such standalone robots, recent research has focused on the approach of **Networked Robots**, which overcomes the limitations of standalone robots by having robots, environment sensors, and humans communicate and cooperate through a network. The IEEE Robotics and Automation Society defined the network robot as *robotic device connected to a wireless or wired communications network (e.g., Internet or LAN) and based on the available protocols such as TCP, UDP, or 802.11. There are two subclasses in Networked Robots:* i) *Tele-operated, where the human operator send commands and receive feedback via the network; and* ii) *Autonomous, where robots and sensors exchange data via the network.* [4]. Since then, **Networked Robots** have became an active research field where many projects have built prove-of-concept solutions demonstrating how this new paradigm can improve the ability of standalone robotic services.

Networked robotics can be considered as an evolutionary step toward cloud robotics [5], which leverages cloud computing technologies to provide massive parallel computation and sharing of data between robots. As a result, robotic systems can become more intelligent, efficient, collaborative, re-configurable, and yet cheaper. The massive parallel computation on demand is now widely available from commercial companies such as Amazon, Google, and Microsoft. These systems provide access to powerful remote processors, data storage, and networking that enables robotics to offload computing-intensive tasks and allows access to a vast resources of data that is not possible to store in onboard memory. The cloud is a step forward in the realization of mission-critical robotic applications by creating a playground for easy sharing of data between robots. For example, robots can share initial and desired conditions, associated control policies and trajectories, and data on the resulting performance and outcomes. Collaborative robot platforms like Lightning, KIVA Systems [6] or MyRobots [7] are proposed in the literature that based on sensor information from the connected robots or the surrounding environment can coordinate the motion of mobile robots by performing parallel planning and trajectory amusements.

Even though cloud computing increased the ability of serviced robots, they remain insufficient for thoroughly supporting mission-critical robotic applications in industrial activities. While the cloud computing model can ensure certain performance and Quality-of-Service (QoS) for the shared pool of configurable cloud computing resources (e.g., networks, servers, storage, applications, and services), very little can be ensured about the factory floor and the network between the robots and the cloud. The factory floor remains proprietary and deterministic. Robotics systems are deployed with proprietary robotics hardware and software along with private access to cloud infrastructure. Let's take as an example a warehouse logistic system. The factory owner buys the warehouse logistic service from the robotic company. The robotic company will deploy the proprietary mobile robots together with the separate wireless infrastructure that will ensure full converge of the factory floor. This connectivity will enable communication with a cloud server that coordinates the robot's trajectories and share updates on detected changes in the environment. In this case, the cloud providers can provide guarantees about the performance of the robotic software that resides in the cloud but they can not provide guarantees about the performance of the wireless infrastructure or the robots. Accessing sensitive information related to the factory floor infrastructure (e.g., radio channel, robot hardware) is not allowed in the cloud due to regulatory, privacy, and security reasons. In addition, the mobile robots

use proprietary software and communication protocols to interact with the cloud making it very difficult to perform collaboration and coordination between robot services from different vendors.

As the requirements of industrial services become more complicated, it becomes difficult to develop a cloud robotics service that supports real-time remote control because such service requires a high level of transparency, reliability, and stability to fulfill the industrial safety requirements. Cloud facilities usually reside far away from the factory floor separated by one or more uncontrolled transit and radio access networks, making it hard to achieve high reliability, low latency, and bounded jitter [8]. In fact, the integration of the industrial robots and external computing resources is very difficult due to the variable and uncertain characteristics of the factory floor [9].

The complexity of the robots that can perform multiple types of tasks may exponentially increase. Similar to the warehouse logistic robots, various types of robots with a single ability are available. Cost is another concern in the wide applicability of such robots on the factory floor. As the mission-critical robots are more responsive, they have a higher control frequency and with that, the production and maintenance costs are very high. We need a means to combine and share different types of nodes (e.g., robots, access points, edge of the network resources) with limited abilities in the cloud-to-thing continuum to perform a sequence of robotic services that are useful to support our industrial activities.

Given the above consideration, in this thesis, we investigate the feasibility of extending the cloud robotics concept towards the edge of the network by using the new ICTs. The edge networked robotics includes all equipment and devices in the cloud-to-robot continuum where robotic functionalities can be distributed anywhere between the cloud and the robots. This cloud-to-robot continuum is not only composed of fixed (stationary), centralized resources (e.g., cloud or edge data center) but also any node or device that is on the factory floor (e.g., robots, access points, industrial computers). The robots interact with the infrastructure to realize an integrated end-to-end system that provides seamless support in daily industrial activities using the available resources on demand. Beyond networking functions, which have been the focus of the edge of the network so far, an opportunity arises also for robotic functions to execute closer to the factory floor benefiting from inherent low latency. Being in close proximity to the access, the edge becomes an attractive natural place for hosting robotic functions, which could also be virtualized. Virtualized robotic functions may therefore execute dynamically anywhere on the edge of the network, but also in distant clouds. Robots could also offer their own networking and computing capabilities to nearby devices and things, thus becoming a living component of the factory floor in what is a new paradigm of D2D communications. A rich set of context information from the RAN, as well as from other network domains, could also be offered as services through the edge for robotic applications to consume and hence optimize their behavior or KPIs.

Based on these opportunities for robotic systems, in this thesis, we design a prototype system with which we aim to experimentally evaluate our concept. We introduce two exemplary industrial use cases: *i)* autonomously navigated mobile robots; and *ii)* remote-controlled robot manipulator. Next, we describe our prototype infrastructure system and we implement two experimental testbeds

that are going to be used in this thesis for field tests.

We present exemplary enhancements that the edge of the network can offer to industrial robotic applications. Context-aware enhancements are examined through example algorithms that target mobile robotic platforms. In addition, resource consumption and savings potentials of robot manipulator functionalities are studied together with the impact that different RATs that are available on the factory floor.

We exploit the concept of orchestration to implement and integrate two innovative orchestration solutions that work on top of shared infrastructure in the cloud-to-robot continuum. These orchestration solutions have a general view of the complete robotic system and can manage the computing and networking resources on demand. Based on our experimental analysis, we list the key benefits of applying orchestration algorithms for industrial robotic services.

We investigate a forecast-based recovery mechanism for real-time remote control of robotic manipulators (FoReCo) that uses Machine Learning (ML) to infer lost commands caused by interference in the wireless channel. Based on our study, we list key challenges for realizing "predictive" remote control.

1.1. Enabling Technologies

In this section, we present the relevant enabling technologies that are used to develop this thesis. In Section 1.1.1 we give a short overview of the evolution of 5G and how it is envisioned to fulfill the industrial requirements. Section 1.1.2 and Section 1.1.3 introduce the concept of edge computing and fog computing respectively, and Section 1.1.4 describes the ETSI NFV framework that works on top of them. The orchestration concept is presented in Section 1.1.5 and finally, Section 1.1.6 presents on federation as a paradigm used to share services or resources between different infrastructure owners.

1.1.1. 5G networks

Early expectations and industrial requirements

One of the main differences between 5G and previous generations of cellular networks lies in 5G's strong focus on machine-type communication. The capabilities of 5G to extend far beyond mobile broadband with ever increasing data rates. In particular, 5G supports communication with unprecedented reliability and very low latencies, and also massive device connectivity. This paves the way for numerous new use cases and applications in many different vertical domains, including the automotive, healthcare, agriculture, energy, and manufacturing sectors. In manufacturing, in particular, 5G may have a disruptive impact as related building blocks, such as wireless connectivity, edge computing, or network slicing, find their way into future smart factories.

Motivated by the new opportunities that 5G can bring to industrial vertical domains, different

standardization bodies and alliances, such as ETSI, 3GPP, 5G Alliance for Connected Industries and Automation (5G ACIA) and Next Generation Mobile Networks Alliance (NGMN) defined new industrial use cases with distinct connectivity requirements that connect people, robots, processes and systems [10][11][12][13]. For example, in the 3GPP report [11], a special section is dedicated to the Factories of the Future where a different set of use cases are defined for full factory automation, process automation, and closed-loop control, human-machine interfaces and remote access and maintenance of the factory floor. Summary of the target KPIs of 5G that are defined by the International Telecommunications Union (ITU) to support the anticipated 5G industrial use cases are presented in Figure 1.1.

In order to support this diverse KPIs by a common cellular infrastructure, 5G-PPP envisioned the use of different new concepts, such as Multi-access Edge Computing (MEC), Network Function Virtualization (NFV), Software Defined Networking (SDN) and network slicing. Additionally, new radio technologies should be introduced which significantly would increase the bandwidth through Multiple Radio Access Technology (multi-RAT) and efficient inter-working with Long Term Evolution (LTE). Frequencies above 6 GHz, in particular, millimeter Wave (mmWave) frequencies (e.g., N257 Band 26.50 - 29.50 GHz) are intended for the extension of 5G spectrum. Despite the limited propagation and increased 2 loss in signal penetration, the focus to mitigate the challenges was set on the use of various radio methods: massive Multiple-Input Multiple-Output (MIMO), beam steering, beam tracking, small-cells, and self-backhauling.

Key characteristics of 5G

Based on the wide variety of use cases related to different vertical sectors and with a goal of defining a set of common regiments for the use cases, the ITU defined the recommendations in three main 5G classes. 3rd Generation Partnership Project (3GPP) used these categorizations in the development of the releases. The three main 5G classes are elaborated in the following.

enhanced Mobile Broadband (eMBB): aims at enhancing the Quality-of-Experience (QoE) of mobile broadband services. 4G provided mobile broadband services where people have truly connected anytime anywhere. The diversity of these types of services such as video and music streaming, mobile payments, mobile gaming, and mobile navigation has resulted in an exponential growth of mobile broadband traffic. In comparison with the mobile broadband services provided by 4G, eMBB is faster, cheaper, and energy efficient. Aiming at these services, industrial verticals should have enhanced access to Augmented/Virtual Reality (AR/VR) applications and high definition video streams from the factory floor. That being said, eMBB aims at the same market segments (mass consumer markets) and will likely follow the similar business model as in mobile broadband with price and tariff plan changes. 3GPP specified the eMBB class for 5G in the Release 15 specification.

massive Machine Type Communications (mMTC): is the fundamental 5G class to fulfill the potential of truly connected world in 5G Internet of Things (IoT). While the mMTC class has been initially introduced in Release 14 as part of 4G, in 5G this class aims at enabling the use of Narrowband IoT

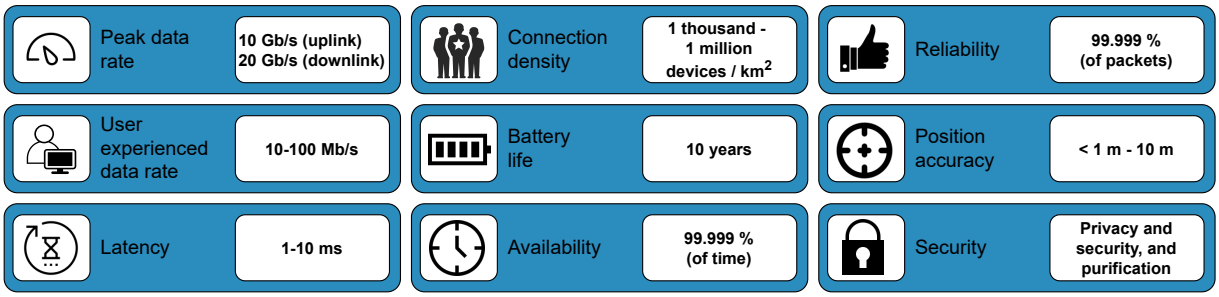


Figure 1.1. *Selected target KPIs of 5G according to ITU-R [14]*

(NB-IoT) for massive number of low-cost devices and extended coverage. Machine-to-machine communication is an important component of IoT, enabling devices to talk to each other through wired or wireless networks. MTC devices are interconnected with each other in different network topologies to provide support for water management, smart grid, environment measurements, patent monitoring, traffic management, smart agriculture, or remote diagnostics systems.

Ultra Reliable Low Latency Communications (URLLC): is the 5G class that makes 5G totally different with respect to its predecessors. Introduced in 3GPP release 15 to address the requirements of ITU, URLLC is one of the key pillars that can be considered as the primary enabler for several unique mission-critical use cases in the areas of industrial automation, real-time control, augmented reality/virtual reality (AR/VR)-based applications, as well as consumer-oriented services such as gaming.

The vertical industries have a good understanding of the technical and business aspect of eMBB and mMTC use cases since they have been introduced with 4G. URLLC use cases are new and a lot of studies and proof of concept projects have been developed to understand these use cases and what are the requirements for the communications systems.

The role of 5G in Industry 4.0

The fourth phase of the industrial revolution, also named Industry 4.0, is the next era in industrial manufacturing, through which industrial verticals are aiming to improve the flexibility, versatility, usability, and efficiency of the factory floor. For the first time, Industry 4.0 tries to brights together the IT and OT domains in order to deliver seamless vertical and horizontal integration down the entire value chain and across all layers of the automation pyramid. Connectivity plays an important role in Industry 4.0, where 5G networks will provide critical communications between people, machines, and devices in real-time. This means that reliable and secure applications such as remote control of a robot arm or a manufacturing production line can be implemented through wireless connectivity with 5G. One example of the use of 5G in industrial environments is the operational mobile panels that allow industrial workers to connect remotely to various systems and facilities and support different types of applications. Historically, every industrial machine or robot has one of these devices attached statically. With 5G not just the costs can be decreased by reducing the number of these devices but also the working conditions of the industrial worker can be improved by allowing them to

quickly and safely access the machines.

5G Roadmap

Based on the ITU recommendations for 5G classes, the 3GPP developed a roadmap for 5G technology roll-out. There are currently 4 phases, that are covered by Release 15, Release 16, Release 17, and Release 18. In the following is described which technologies and features have been rolled out in each of the releases.

3GPP Release 15: it was released in 2017. The main focus of this standard is enabling eMBB and was composed of 3 separate sub-releases. The first sub-release was primarily on mobile broadband for non-standalone (NSA) 5G architecture. NSA architecture means 5G would require 4G as a base, with 5G on top to boost capacity or network speeds. 5G NR technology was also introduced in the first sub-release together with the hardware specifications for development of 5G NR . From an industrial point of view, the 5G NR specifications were the most important element of this sub-release because they allowed verticals to have an early look into the development difficulties that come with 5G NR. The second sub-release of Release 15 was in 2018 and it introduced the 5G SA architecture that allows the existence of 5G without a 4G base. Together with the 5G SA architecture, an entire set of 5G core functionalities were released which makes the SA architecture possible. The third and last sub-release was released in 2019 and concluded with Release 15. In this sub-release, the 5G system is extended with a description on mission-critical communications and use cases such as Vehicle-to-everything (V2X) communications and features related to Mobile Communication Systems for Railways, Virtual Reality (VR), multimedia, Operation Administration and Maintenance (OAM) are defined.

3GPP Release 16: continued the previous work on eMBB from Release 15 but also in introduced the support for URLLC 5G Class as one of the most significant features. The main focus of Release 16 was the industrial verticals rather than end customers. New features were introduced in the 5G system such as network slicing, 5G private networks, and edge computing that enables enterprises to change aspects of their networks over and pave the way for complete automation and safety. From a business angle Release 16 introduced features such as location and positioning Services, 5G in unlicensed spectrum, Intelligent transportation systems (ITS), vehicle-to-anything (V2X) communications, and Integrated Access and Backhaul (IAB).

3GPP Release 17: one of the major objectives of this release is to extend the current 5G waveform up to 71 GHz. Another major feature of Release 17 is to introduce cost-effective NR NB-IoT/eMTC devices with capabilities that lie between the full-featured NR and NB-IoT. This release also envision studies about the evaluation and improvements that the 5G system needs to make in order to be better suited for AR/VR/MR and Non-Terrestrial Network support for 5G and NB-IoT/eMTC. These studies include an evaluation of distributed architectures that fully utilize the complete cloud-to-thing continuum to optimize latency, processing, and power. In addition to the study items, since Release 16 was a short release, Release 17 aims at perfecting the new concepts such as Integrated Access and Backhaul, Positioning, Sidelink, and URLLC use cases that were introduced in Release 16. Finally, Re-

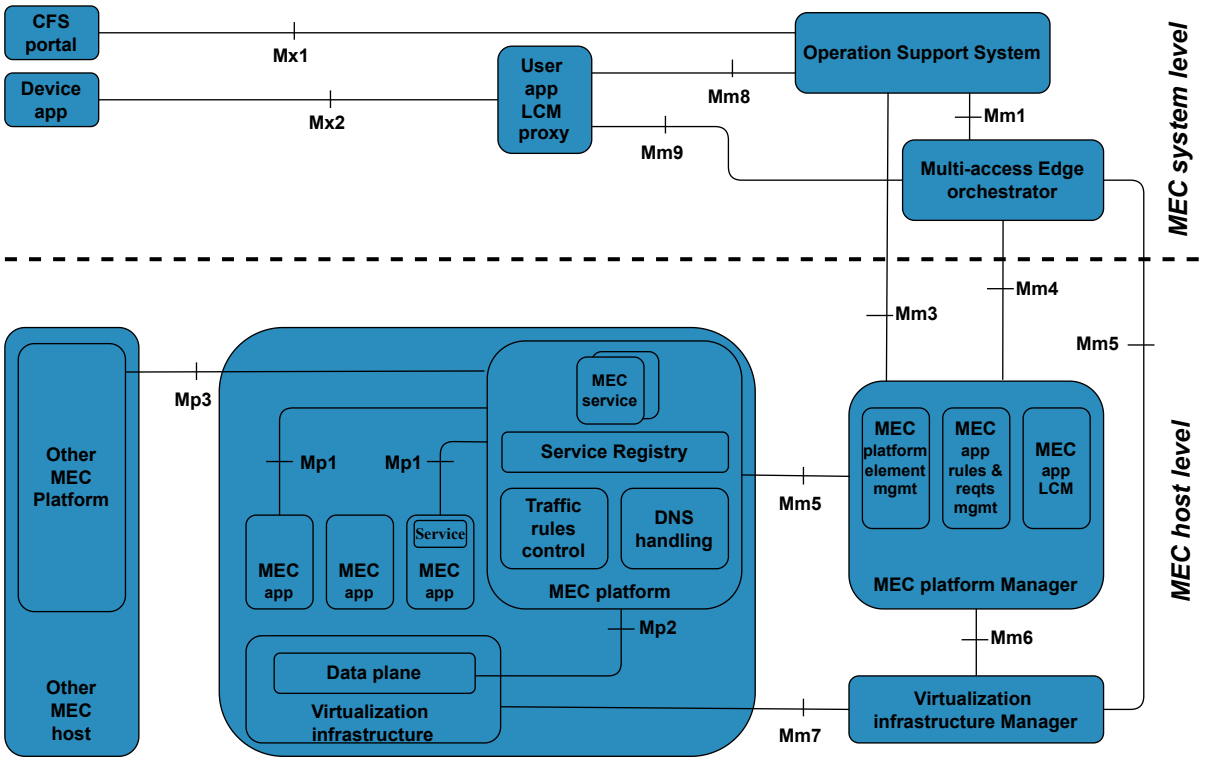


Figure 1.2. *ETSI MEC reference architecture [15]*

lease 17 will also try to optimize the eMBB features defined in Release 15 based on the findings from the early 5G deployments.

3GPP Release 18: will further enhance 5G performance, introduce features to enable more flexible and efficient spectrum use, advance the support of diverse NR devices such as reduced capacity UEs, evolve network topology to facilitate different deployments, and provide data-driven, intelligent network solutions by introducing AI/ML-based solutions for RAN. Release 18 is still in the very early stage since its approval has been released in December 2021. It can be expected that the innovative works conducted in this Release such as AI and ML technologies, will set the strong foundation for 6G design and create an important impact on future wireless systems.

1.1.2. Multi-access Edge Computing

With the rise of the URLLC service class, 5G systems aimed to satisfy several technical requirements in order to cope with various emerging applications [16]. Specific applications such as Augmented Reality (AR), connected vehicles, and robotics require reliable communications with very low end-to-end latency to deliver high quality services [17]. Fulfilling such requirements is extremely challenging for centralized network architecture and requires the gradual shifting of networking, computing, and storage capabilities closer to the end users to eliminate the delay caused by data transfer to distant cloud servers. This concept is also known as the edge of the network and it integrates and extends the

edge computing approach.

Motivated by these needs, the European Telecommunications Standards Institute (ETSI) has been the first to address this need by providing the framework of Mobile Edge Computing (MEC) [15], which is supported by the concept of Network Functions Virtualization (NFV) that is explained later in this section. ETSI has further re-branded MEC as *Multi-Access Edge Computing* to remark its goal of achieving multi-RAT coordination via the edge. In the ETSI MEC approach, the edge virtualization substrate has been largely assumed to be fixed (stationary) and centralized in a pre-defined location (e.g., edge data center), and exclusively owned by one stakeholder, that is the operator. In this thesis we use this ETSI MEC approach as a definition of edge computing.

The MEC framework is shown in Figure 1.2 and it identifies and groups two high-level functional domains:

- MEC host level where the MEC host is placed together with the MEC platform and the virtualization infrastructure that runs the applications.
- MEC system level management which provides orchestration and management capabilities of the MEC system. This entity has a global view of the whole MEC system with the MEC orchestrator as its most important component.

The domains are composed of functional components that can be deployed as virtual functions or in any form (e.g., native applications, bare metal proprietary software). While main MEC component at host level is the MEC host, which can be seen as an edge data center, the main MEC component at system level is the MEC orchestrator and it operates across multiple MEC host levels.

The **virtual infrastructure** component of the MEC host provides compute, storage and networking resources for the MEC application. The virtualization infrastructure includes a data plane element that enforces the forwarding rules received by the MEC platform and routes the traffic between the applications, services, and networks.

The **MEC application** uses the virtual infrastructure provided by the MEC host and runs as virtual component (e.g., VM, container, etc.). This component interacts with the MEC platform using the Mp1 interface to consume and/or provide MEC services in the MEC host. The MEC application can indicate the service and resource requirements (e.g., latency constrain) so when instantiating the application, the system level management can select the target MEC host as well as perform load balancing at a MEC host level.

The **MEC platform** enables MEC applications to discover, consume and provide MEC services. It also includes essential functionalities that are needed to successfully execute MEC applications on the MEC host such as traffic steering or storage. In addition, the MEC platform can act as a local DNS server, in order to forward user traffic to the MEC applications. The communication between different MEC platforms is enabled through the Mp3 reference point.

The **MEC platform manager** is the bridge between the MEC host level and MEC system level domains. While in the MEC host level domain it is responsible for the life-cycle management (instan-

tiation, termination and reallocation) of the MEC application, in the MEC system level domain it interacts with the MEC orchestrator to provide him with indications on some MEC application related events.

The **Virtualized Infrastructure Manager (VIM)** is responsible for the management of the virtualized compute, storage, and network resources provided by the virtualization infrastructure for the MEC application. It interacts directly with the virtualization infrastructure via the Mm7 interface. Examples of commercial VIM solutions are OpenStack and Kubernetes.

The **MEC orchestrator** has general view of all the resources and capabilities of the entire MEC system. It is responsible for the instantiation, termination, healing, and coordination of the MEC applications and related procedures such as onboarding, authentication, and integrity checks. The orchestrator fulfills the pre-defined application requirements (e.g., latency, user throughput) by selecting the appropriate target MEC host, and if needed it can also trigger the application relocation.

The operator's **Operations Support System (OSS)** is the highest management entity of the MEC system that is responsible for running the MEC applications in the desired location of the network. The OSS receives requests directly from the users through the Customer Faceting Service (CFS) portal and interacts with the orchestrator to fulfill them.

The **CFS** is the user entry point in the MEC system and it is envisioned to be similar to a cloud platform portal. The user can use the CFS to fill in information related to onboarding, instantiation, and termination of the application.

Finally, the MEC system is designed to offer enhancements to the MEC application through **MEC services**. The MEC services can be provided or consumed by either the MEC platform or the MEC application. A MEC application subscribes to the set of available services in the system over the Mp1 reference interface. ETSI provide exemplary MEC services such as the Radio Network Information Service (RNIS) [18] and Location Service. The RNIS enhances the MEC applications by allowing them to use up-to-date radio network information regarding radio network conditions when adjusting the operations for particular user. While the original RNIS service was designed for cellular networks, a new service is being defined to cover also Wi-Fi networks (WAIS) [19]. The Location Service [20] allows the MEC applications to use information related to the specific location of the UEs currently serving by the radio node(s) that are associate with the MEC host. Finally, the Bandwidth Management Service allows prioritization of certain traffic.

MEC in 5G networks

As described in the 5G networks section, the 3GPP 5G system specifications and its Service Based Architecture (SBA) are ideal target setup for MEC deployments. The specifications of 3GPP 5G system and MEC leverage on same networking paradigms such as virtualization, Software Defined Networking (SDN), network functions, and service based interactions. Additionally, the 3GPP 5G system specifications outline the concept of edge computing, where the MEC system and a 5G system can

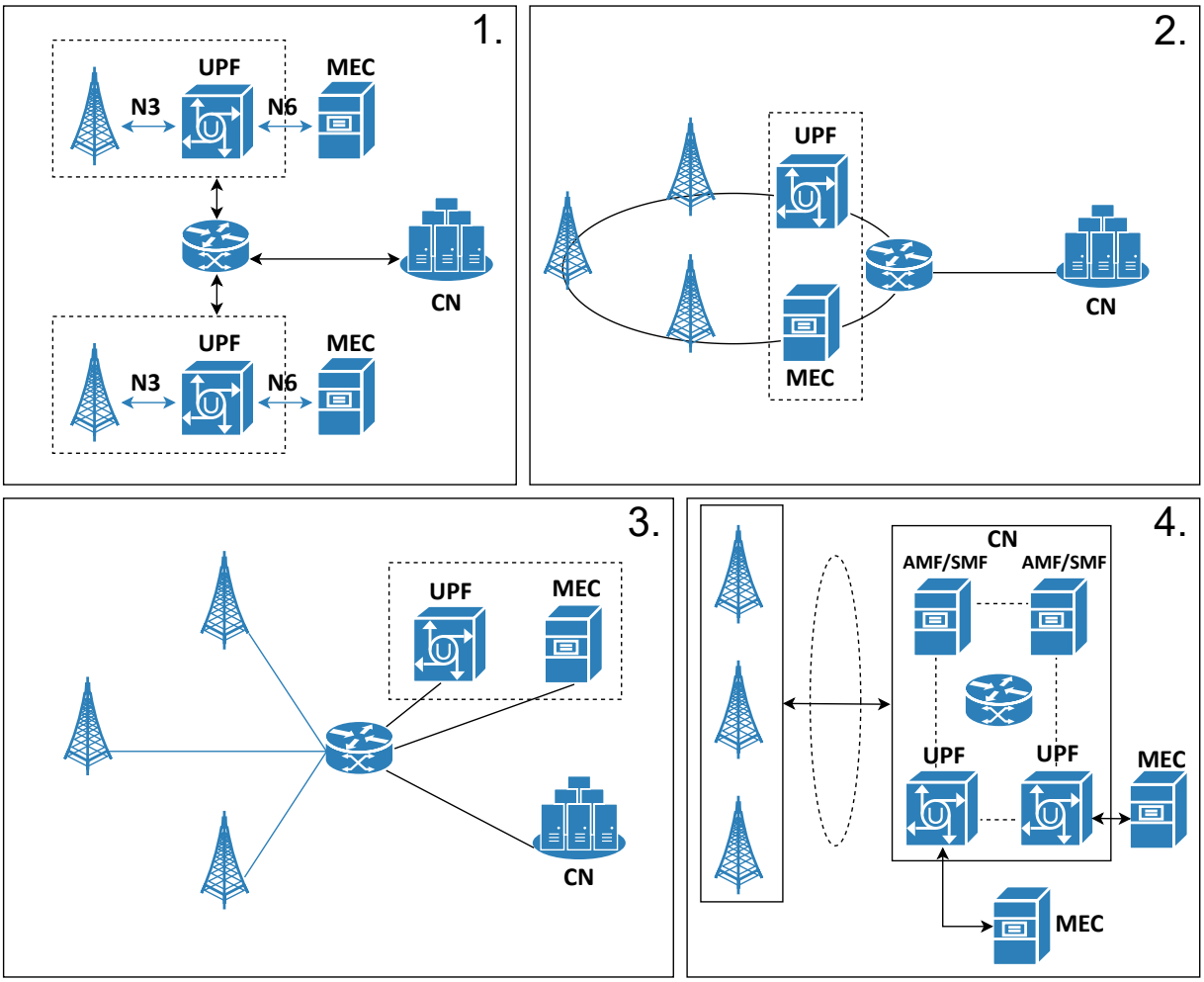


Figure 1.3. *Example scenarios of phisical deployment of MEC in 5G [21]*

work together on tasks like traffic routing and policy control. MEC framework in combination with the 5G system can create a powerful environment for edge computing. In the following section, we describe examples of the physical deployment of MEC in the 5G network that is presented in ETSI white paper[21], with a goal of specifically defining the term of edge computing in the scope of this thesis.

Logically MEC hosts are deployed in the edge or central data network where the User Plane Function (UPF) is responsible for directing the user plane traffic to the targeted MEC application. The network operator decides the location of the data network (e.g., edge or central), UPF and computing resources based on technical and business observations such as available physical space, type of applications that need to be supported, network traffic load, etc. Different MEC hosts and applications are orchestrated by the MEC management system that dynamically makes decisions about where to place the MEC applications. Based on different operational, performance, or security-related requirements, there are numerous solutions available for the physical deployment of MEC hosts. Figure 1.3 presents some of the practical options for the physical location of MEC:

1) MEC and the UPF located with the Base Station.

2) MEC located with a transmission node, possibly with a UPF.

3) MEC and the UPF located in the aggregation ring

4) MEC located in the same data center with the Core Network functions

The scenarios presented in Figure 1.3 show the flexibility of MEC to be deployed in different locations, spanning from the Base Station (BS) to the central data network. In the work presented in this thesis the location of the edge computing is defined by referring to scenario *3)*, where the MEC and the UPF are located in the aggregation ring. The UPF is used to perform the local breakout in the aggregation ring by steering the traffic towards the MEC applications and/or towards the central data network.

1.1.3. Fog computing

In the previous section of this chapter, we defined the concept of edge computing throughout the definition of the MEC framework. However, the edge of the network is not only composed of computing resources in the form of servers, indeed the edge is usually divided into *i)* far edge; and *ii)* near the edge. While the MEC paradigm covers the far edge concept where the static computation resources are pushed deeper in the network infrastructure, the near edge concept integrates any node or device that is in near proximity of the end-user. All these devices that are at the very last hops of the network infrastructure are known as fog devices. A fog device provides computing, storage, and/or networking capabilities on resource limited nodes. This device may be volatile, mobile, battery-constrained, or have other constraints, which limit its availability, capability, and design considerations such as the dominance of wireless connectivity. In general, commercial fog devices are based on stand-alone hardware entities combined with a software platform that allows main fog functionality, such as remote, automatic software deployment, connectivity with end devices (sensors – e.g. cameras, actuators, industrial robots), and to some extent interconnection with other fog devices (creating a specific topology) via wired and/or wireless media. Fog device capability (computation, storage) varies and is dependent on the applications of interest, but it is often significantly lower than that of data centers.

Consortium (IIC) [22] put has put effort to define fog computing, as well as the use cases and the reference architecture that explains the benefits and functions that the fog concept can bring. Whilst by the IIC definition the fog computing includes edge computing, and all aggregations of the edge towards the cloud, the most appealing value of fog has been in complementing the edge by extending it further down to the very distributed computing substrate of volatile, mobile and constrained devices. Because of these most significant added-values of the fog, in this thesis, we scope the fog around the volatile and constrained substrate, complemented by the edge next, and further on by the cloud as shown in Figure 1.4. It is worth mentioning that ETSI also has a working group with the aim to include distributed and constrain resources as part of the MEC vision.

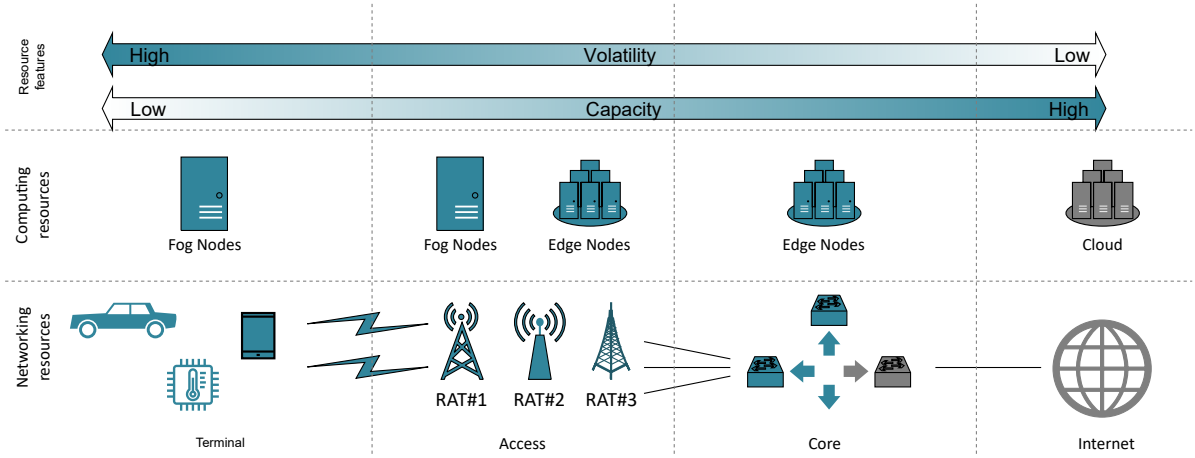


Figure 1.4. *Cloud, edge and fog resources and characteristics*

As presented in Figure 1.4, the overall pool of resources at the edge of the network is divided into three groups:

- Edge Data Centres (Edge DCs) are a first core group, that spans the upper tiers of network aggregation layers. Essentially, this core group is pushed deeper into the network infrastructure and is capable of handling heavy computing tasks as well as storing a large amount of data. This may happen, for example, in a company's headquarters.

- The second access group is around the Base Stations and Access Points and consists of mini-edge DC and fog devices. Local DCs or servers connected to the APs may be included in this access group. As it can be noticed this group includes both edge and fog computing.

- Fog devices of limited resources make up the third access group, which is often mapped into terminal networking resources. This might include onboard computing technology in cars or trains, PCs, smartphones, robotics, and drones, among other things. Some lightweight but latency-sensitive computational operations can be handled by these Fog devices.

1.1.4. Network Function Virtualization

The edge and fog computing go together with virtualization technologies where ETSI introduced the concept of Network Function Virtualization (NFV) [24]. NFV was born with the idea of running network nodes and services (e.g., switches, gateways, load balancers) on general-purpose hardware by the use of virtualization. The network services are traditionally hosted with proprietary vendor hardware that is sold to telco operators. With the introduction of NFV, the services are not split from the hardware where the use of VMs or containers allows telco operators to run their network services on commodity servers rather than on vendor-specific equipment.

The NFV combined with edge and fog computing has the potential to lower the operating expenses of telco operators by reducing the maintenance, hardware costs, power consumption, phys-

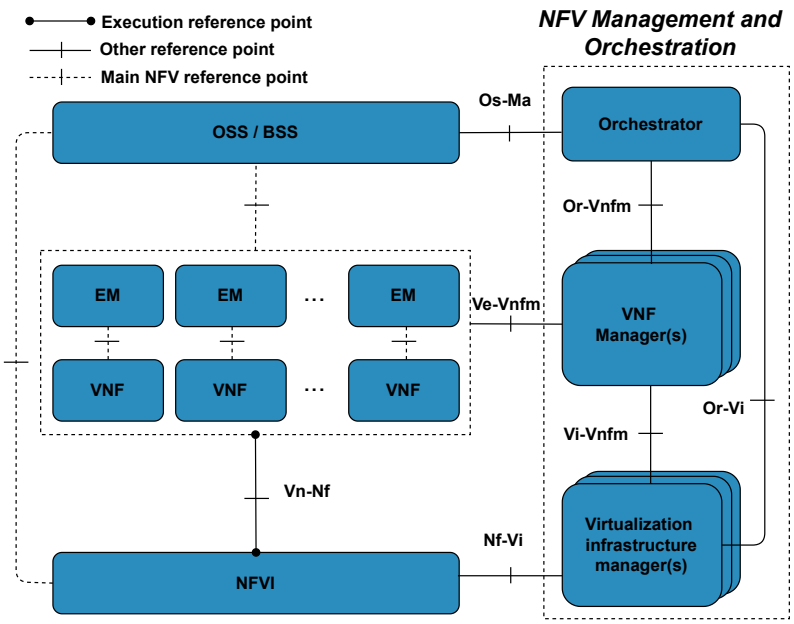


Figure 1.5. *ETSI MANO reference architecture [23]*

ical space, and network upgrades. Additionally, NFV makes the service provisioning easier and the networks more flexible. Operators are better prepared to deal with difficulties such as adapting to customer-changing demands since the network functions are now aggregated in software and they can be relatively easily scaled or moved to different network locations. Finally, NFV enables telco providers to offer customers isolated and dedicated network resources on the fly. This concept is known as network slicing and allows telco providers to create isolated networking resources over shared networking infrastructure (e.g., computing, radio access) that work within strictly defined rules in terms of latency, bandwidth, etc. Based on the vertical industries' specific requirements, the operator can create a network slice that will guarantee the end-to-end QoS.

The main architectural framework [23] for NFV that is widely adopted by the industry is presented in Figure 1.5. This framework is composed of three domains:

- Virtual Network Function (VNF) running on top of the NFV Infrastructure (NFVI).
- NFV Infrastructure (NFVI), that supports the execution endowment of the VNFs.
- NFV Management And Orchestration (MANO) that provides the orchestration and life-cycle management of the NFVI and also the VNFs.

The **NFVI** is composed of hardware resources together with the virtualization layer. It uses virtualization software that simulates the hardware functionality to create the environment in which VNFs are deployed. The NFVI virtualizes physical computing, storage, and networking and places them into resource pools.

The **Virtualized Infrastructure Manager(s) (VIM)** has a similar functionality as the VIM in ETSI MEC and its the functional block that controls and manages the NFVI compute storage and network-

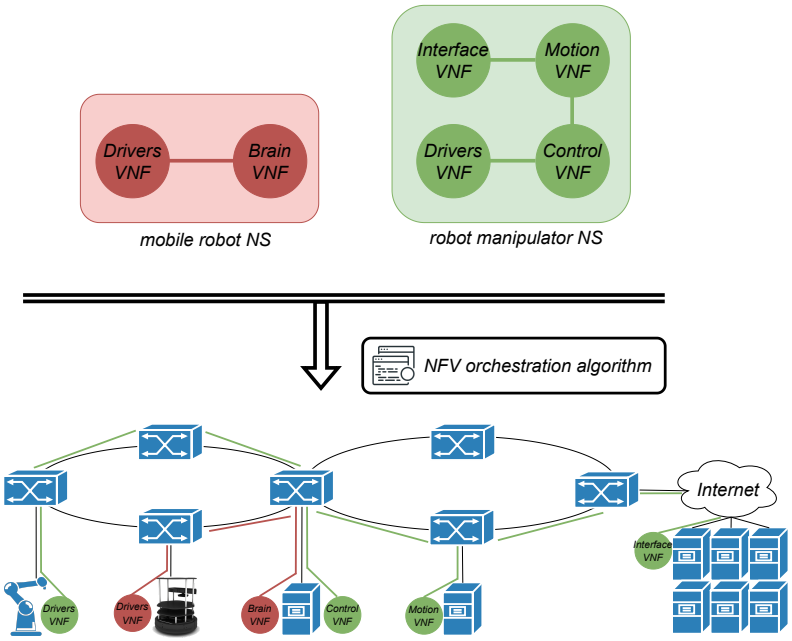


Figure 1.6. *Example of NFV orchestration*

ing resources. It is responsible for controlling and managing the VNF virtualization, in addition to their interactions with computing, storage and network.

The **NFV Orchestrator (NFVO)** has two main roles that it needs to fulfill: *i)* the Resource Orchestration (RO) role where it provides orchestration of NFVI resources across multiple VIMs; and *ii* the Network Service Orchestration Role (NSO) where it provides life-cycle management of the Network Service (NS). The NFVO interacts with the OSS/BSS to provide installation and life-cycle management of new NS and individual VNF packages. It has a general view of the whole system and is responsible for the global resource management and validation and authorization of NFVI resource requests.

The **VNF Manager(s)** is a functional block with the main responsibility for the life-cycle management (e.g., instantiation, update, query, scaling, termination) of VNF instances.

The **Operations and Business Support Systems (OSS/BSS)** provide the operational and business support systems implemented by the VNF service provider.

It is worth mentioning that in order to deploy a NS through the NFV MANO ETSI envisioned the use of templates/descriptors that contain specifications regarding the individual VNFs, VNF Forwarding Graphs, service-related information and NFV infrastructure information models.

1.1.5. Orchestration in NFV

With the advancement of NFV, edge computing and fog computing in the 5G networks, the concept of *service orchestration* has been introduced. Vertical industries, such as automotive, e-health, and smart factories, can define services through a set of VNFs that can be deployed in the cloud-to-thing

continuum. Upon the deployment of a network service, three main decisions have to be taken: *i)* the allocation of resources for VNFs in the servers; *ii)* the traffic steering of the Virtual Link (VL)s across the network links, and *iii)* the allocation of bandwidth resources on the links that steer the Network Service (NS)s traffic. These decisions are taken by a NFV orchestration algorithm, i.e., an algorithm that maps the VLs and VNFs in the underlying physical infrastructure, or even in an abstraction of the physical infrastructure. It is important to remark the difference between an orchestrator, which assesses the management and assignment of services to resources, and a NFV orchestration algorithm, which solves the Virtual Network Embedding (VNE) problem. That is, to embed a virtualized service in a substrate network. Figure 1.6 shows an exemplary NFV orchestration algorithm that performs traffic steering of the VLs and the mapping of VNFs to servers.

The goal of a NFV orchestration algorithm is to deploy a NS in the cloud-to-thing continuum, taking into account the volatile and resource-limited resources of the fog and route the traffic in between VNFs. The decision-making process of an NFV orchestration algorithm when solving the VNE problem is usually composed of two problems: *i)* where to deploy the VNF and how much resources should be allocated, and *ii)* what links should be used to route the traffic between VNFs. Orchestration algorithms to solve the VNE problem were used even before 5G networks. Early solutions focused on energy consumption, minimization of resource waste, or even temperature within a data center infrastructure. With the arrival of the NFV paradigm and the functional split of the NSs into multiple VNFs that can run in different servers, the problem complexity increased. The VNFs that compose the NS can be spread across the complete network infrastructure and the flexibility of the deployment options increased. This distribution of the VNFs in the infrastructure also resulted in increasing the complexity of the traffic steering between VNFs, which in the past were all running on the same server. In the following, we present some exemplary NFV orchestration algorithms from the existing literature. In [25] the author proposes an algorithm that discovers deployments that decrease the costs by utilizing matrices and eigen-values. [26] proposes an orchestration algorithm that uses VNFs replication to minimize the link utilization in the network used to deploy the NSs. In [27] an orchestration algorithm is proposed that determines the number of VNFs that each server should host to process completely or partially the new incoming flow demand. Another orchestration solution that decides how many VNF instances should be deployed in a server in order to process the client requests is presented in [28].

All this algorithm that are proposed in the literature solve the VNF problem but they did not consider the strict communication constrains of 5G services. As we elaborated in Section 1.1.1.2, 3GPP and ITU-R defined the URLLC class of use cases so the networking community started to tackle the VNF problem in 5G scenarios where the (VNE) problem complexity increased again. While some of the algorithms [29] targeted deployments of process control function in industrial use cases, other algorithms [30] solve the mapping between the VNFs and VMs and the possibility of sharing VNFs. For more detailed analysis on existing NFV orchestration algorithms in 5G scenarios, [31] elaborates on the current State-of-the-Art.

1.1.6. Federation in NFV

In the ETSI MEC platform, the edge virtualization infrastructure is developed to complement the cloud in the form of edge data centers that are typically located outside big cities and owned by telco operators or service providers (e.g., Amazon Green Grass, Google distributed Cloud). The business model is very similar to the one of cloud computing where compute, storage, and network resources are offered in a pay-as-you-go fashion. With fog computing the cloud was even further extended and a cloud-to-thing continuum was created where different providers can be resource owners in the shared infrastructure. The VNFs that compose the network service can be placed in the resources that are owned by shopping malls, business centers, hospitals, or factories. Having in mind that fog resources usually have limited availability due to the volatile, mobile, and resource and battery-constrained nature, the dimension and management of the cloud-to-thing continuum to satisfy the service requirements are very challenging. In the case of fog node failure (e.g., empty battery or lost connectivity) or an increase in the number of users for a given service, service providers will not be able to react in time in order to keep the service operational.

In these scenarios, thanks to NFV service providers are enabled to use services and/or resources owned by other stakeholders/providers. This concept, known as *federation*, enables service providers to use services/resources from external Administrative Domains (AD). In a NFV environment, the federation concept is used by administrative domains (ADs) to deploy network services or allocate resources over the infrastructure of an external domain. The federation procedure is triggered by a consumer domain as a need of extending a specific service (e.g., at a specific geo-location) or the lack of constituent resources. Under the umbrella of the 5G-Public-Private-Partnership (5G-PPP) and the European Research Council (ERC), the European networking community has been researching the concept of federation and how it can support Industry 4.0. 5G-DIVE [32] and 5Growth [33], are two examples of research platforms that were actively contributing to the development of the federation concept. Moreover, there are recently existing works [34–37] that also address the open challenges in the federation process by proposing different federation mechanisms.

The Federation requires all collaborative administrative domains to have mutual cooperation agreements. Typically, these agreements are signed by business executives. Upon agreement, the agreed ADs trust each other and establish peer-to-peer connectivity among themselves or define a trusted centralized entity to manage their interactions. Therefore, federation interactions can be executed in a centralized, decentralized, or distributed manner. In a centralized solution, all involved ADs have to sign mutual agreements and trust a centralized entity located in a neutral location. The centralized entity is managed by the joint community of involved ADs and oversees the federation interaction, acting as a neutral "middle-man". The positive side is that it is a highly trusty and scalable solution, however, the disadvantage is the set-up time, joint effort, continuous maintenance, and that it is a single point of failure system. A decentralized solution is the simplest to employ, at the cost of the lowest scalability. Each AD establishes peer-to-peer connectivity with each external AD. This implies that each newly created (federation-ready) connection between ADs is followed by a business meeting and inter-domain connectivity set-up (e.g., 50 connections between different ADs would take at least

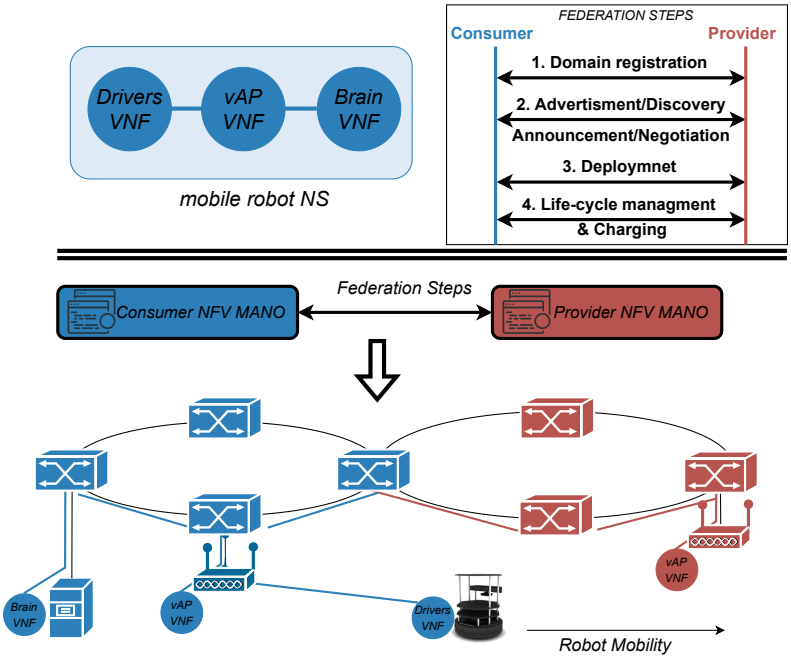


Figure 1.7. *Service Federation in NFV*

50 or more days). Several approaches rely on this type of federation [38] [39]. A distributed solution is a hybrid approach, more similar to the centralized solution, where the central entity is distributed in each AD.

In this thesis when we refer to the concept of federation we focus on the service federation rather than the resource federation. An example of a service federation is presented in Figure 1.7. The consumer domain throughout the NFV MANO requests an extension of a service (or part of a service) to be deployed over a provider domain. The provider domain oversees the complete deployment process of the service extension. While in resource federation the provider domain only provides available resources (e.g., computing or networking) to the consumer domain, and the deployment of the service extension is executed by the consumer domain. In order to successfully complete a service federation [34] [40], there are several federation procedures that are executed in sequence:

- **Registration** - initial procedure through which the administrative domains establish their peer-to-peer inter-connectivity or register to a central entity. The registration procedure characterizes the type of federation, which can be relatively open or strictly closed. As an open federation can be considered when external new domains can more easily register to the peer-to-peer or centralized interaction. The closed federation includes pre-defined participants with strict policies and rules, manually set and defined by the ADs.

- **Discovery** - in this procedure the participating ADs periodically broadcast or exchange among themselves information on their capabilities to provide services or (computing/networking) resources. Each AD creates and continuously updates a global view of the available service or

resources at the external ADs.

- **Announcement** - this procedure is triggered by the consumer domain, once it has been decided the need to federate part of a service in an external peering domain. An announcement is broadcast to all potential provider ADs. The announcement conveys the requirements for a given service or set of resources. In the centralized case, the central entity is used as a proxy.

- **Negotiation** - the potential provider ADs receive the announced offer, analyze if they can satisfy the requirements and send back a positive or negative answer. The positive answer includes the pricing of the service.

- **Acceptance & deployment** - the consumer AD analyzes all collected answers and chooses an offer of a single provider domain. The selection process is entirely left to the consumer AD's internal policies and preferences. The consumer domain sends an acceptance reply to the chosen provider AD. The provider AD starts the deployment of the requested *federated* service.

- **Usage & Charging** - once the provider AD deploys the federated service, it notifies the consumer AD and sends all necessary information for the consumer AD to include the federated service as part of the end-to-end service deployment. From that on, the provider AD starts charging for the federated service during its life-cycle, until it is terminated.

Please note that the security/privacy and trust among the participating ADs is vital in all the aforementioned procedures. Actually, due to competitive reasons, any AD (e.g., mobile operators, cloud providers, etc.,) would not reveal much information regarding the underlying internal infrastructure or the full capabilities for service deployments.

1.2. Research Challenges

In context with the previously outlined considerations, seven general open challenges were considered in the realization of this thesis. Each of these challenges is addressed in one of the subsequent chapters of this thesis (see Section 1.3).

CHALLENGE C1: HOW TO EFFICIENTLY INTEGRATE ROBOTIC SYSTEMS WITH NETWORK AND COMPUTING INFRASTRUCTURES HAVING IN MIND THE VARIABLE AND UNCERTAIN CHARACTERISTICS OF THE FACTORY FLOOR?

The adaptation of industrial processes towards a concept based on networked robots is facing a major challenge with respect to the unification and exchange of information by different applications [41]. To make use of the networked robots, applications must have access to information from different systems, making abstractions a very important topic to handle the underlying heterogeneity. Moreover, manufacturing service encapsulation, composition, and publication are identified as key technologies that need to be studied [42]. Cloud robotics enables networked robot systems to be

deployed at the Cloud. Cloud-based robotic systems are suitable for delay tolerant, long-term, high computational applications (e.g., monitoring or simulation systems) crucial for preventive maintenance and business-decision support. However, proper execution of a robotic system in manufacturing needs real-time access to production data, making real-time connection and synchronization a major challenge.

CHALLENGE C2: HOW TO USE THE REAL-TIME NETWORK CONTEXTUAL INFORMATION TO OPTIMIZE THE END-TO-END ROBOTIC SYSTEM?

Edge computing and fog computing can be seen as cloud robotics extensions to the network edge since they exploit similar computation offloading and storage management schemes. A unique opportunity arises for robotic services to use context-aware techniques, such as real-time information for the radio network condition that is available only at the network edge to optimize the end-to-end performance of the robotic system. Moreover, these context-aware techniques can be used to enable robots to communicate directly in a Device-to-Device (D2D) fashion through a network-assisted approach [43]. All these concepts are very well defined in the existing literature by proposing different communication architectures. However, very few studies show the exemplary usage and benefits that the robot applications can experience from the usage of network context information.

CHALLENGE C3: WHICH FUNCTIONS FROM THE ROBOT COULD BE ADVANTAGEOUS TO VIRTUALIZE IN THE EDGE AND FOG? WHERE IN THE EDGE AND FOG ARE THEY BEST SUITED TO EXECUTE?

The manufacturing sector, as the largest world energy consumer (55% of the world's energy [44]) is in a constant search for energy efficient optimization strategies that will reduce the energy consumption. Cloud robotics started to contribute toward energy optimization in the manufacturing sector by offloading computation from the factory floor to the cloud. However, as we mentioned before this computation offloading was only suitable for delay tolerant, long-term, and high computation robotic applications. Additional reduction of the energy consumption can be achieved by offloading robot computation modules in an edge server. One of the main challenges is how to reduce the on-device energy consumption while fulfilling the strict safety and reliability factory policies. This reduction of energy consumption on the robots not only contributes toward the global energy optimization of the factory floor but also contributes to the reduction of the cost of the industrial robots.

CHALLENGE C4: WHAT ARE THE BENEFITS THAT 5G CONNECTIVITY CAN BRING TO REMOTE CONTROLLED ROBOTS?

Wireless technologies provide several benefits for manufacturing, such as greater flexibility for connecting physical devices, reductions in installation and maintenance costs, support for mobility, and improved safety for the employees [45]. As of today, wired networks are widely used in industrial automation since wireless technologies fall short of meeting the stringent requirements of real-time applications. Consequently, the physical devices are connected through wired technologies such as

Fieldbus [46] and industrial Ethernet [47], which are costly and inflexible. With the emergence of 5G, Industry 4.0 recognized the opportunity for a unified communication interface that can support the requirements of any type of robots through three main service profiles: *(i) eMBB*; *(ii) mMTC*; and *(iii) URLLC*. 5G provides guaranteed QoS for time-critical applications as a built-in feature. In addition, 5G also defines support for handling mobility in industrial environments that opens the possibility to apply the networked robotics concept for movable platforms (e.g., mobile robots, automated guided vehicles). Despite all the expectations, industrial verticals remain skeptical about the maturity of 5G and the benefits that it can bring especially to remotely controlled robot manipulators whereas we know wireless connectivity is not required. There is a lack of experimental studies that deal with the integration of 5G connectivity with current network deployments, aiming to give insights into performance improvements for industrial verticals willing to invest on 5G.

CHALLENGE C5: HOW TO OPTIMALLY ALLOCATE COMPUTING AND NETWORKING RESOURCES FOR ROBOTIC APPLICATIONS IN THE CLOUD-TO-ROBOT CONTINUUM, TO SATISFY KEY PERFORMANCE INDICATORS (KPIS) AS LATENCY

The NFV paradigm has been currently consolidated as one of the key enabling technologies towards the implementation of Industry 4.0. This technology aims at alleviating the hardware dependencies in the provision of robotic functions and services, using virtualization techniques that allow those functionalities to be executed in commodity equipment over the cloud-to-robot continuum. As mentioned before, the cloud-to-robot continuum is composed of large scale public/operator-owned data centers, small scale computing infrastructure deployed at the edge, and a heterogeneous set of resources with limited computing capabilities like network nodes, end-user devices, or robots. This heterogeneous and distributed computing infrastructure needs to fulfill the strict robotic functions KPIs with respect to latency (0.5-20 ms), jitter ($\pm$ 0.5), and reliability (99.9999%) [10–13]. In this context, an open challenge is how to optimally allocate the computing and networking resources for robotic applications in this cloud-to-robot continuum while satisfying the KPIs.

CHALLENGE C6: HOW CAN A ROBOTIC SERVICE EXTEND ITS SERVICE FOOTPRINT IN A FAST AND EFFICIENT WAY OVER INFRASTRUCTURES THAT ARE OWNED BY DIFFERENT STAKEHOLDERS?

Another key enabling technology for the implementation of Industry 4.0 is edge computing. By placing computing, networking, and storage resource near the edge, robotic systems can execute robotic applications closer to the robots resulting in more predictable communication and overall better system performance. For the market, such robotics applications are an opportunity for mobile robots to be employed in accomplishing a range of manual tasks (e.g., security/surveillance, cleaning, delivery of products, warehouse logistics, etc.). The key element of such mobile robotic system is the uninterrupted robot connectivity over the access network and the available real-time information about the connectivity. The high mobility of robots demands a change in the point of access in the access network which is currently feasible within a single administrative domain. Often enough, an edge robotics service requires fast and short-lasting expansion of the service footprint over infrastructures

that are owned by different stakeholders. This concept in the literature is known as federation and it enables the orchestration of resources and services across multiple administrative domains. Still, an open challenge for robotic services is how to perform the federation of services in a secure and trusted way while not interrupting the operation of the mobile robots.

CHALLENGE C7: HOW CAN AI/ML ASSIST THE NETWORKED ROBOTIC SYSTEMS IN ITS INTEGRATION WITH THE NETWORK AND COMPUTING INFRASTRUCTURES TO SATISFY THE EXPECTED REAL-TIME PERFORMANCE?

The cyber-physical integration in industrial environments mirrored through the networked robots arises a perfect opportunity for the development of AI agents to devise smarter and more accurate robots. ML is a strong candidate to implement such agents, as an alternative to heuristic or decision-tree based solutions, among others. Robots have sensors(e.g., odometry, camera, lidar) that transfer the information about the robot state and its surrounding into raw data in the cyber space, which can be later used to train and cross-validate different ML algorithms and AI agents. These agents not only learn about the details of the specific industrial tasks but also extend and optimize them beyond the human capability due to the volume of data they can handle to make decisions. However, an open challenge in the existing literature is how to build AI-agents that benefit from ML algorithms to exploit the excising data sources with context information at both the application and infrastructure level. There is a lack of exemplary AI-agents that can contribute towards the efficient integration of networked robot systems with existing network and computing infrastructures.

CHALLENGE C8: HOW TO ACHIEVE RELIABILITY, TRANSPARENCY, AND STABILITY FOR REAL-TIME REMOTE CONTROL OF ROBOT MANIPULATORS OVER IEEE 802.11 CHANNEL

Nowadays, industrial verticals implement IEEE 802.11 technologies for factory automation through commercial solutions such as Industrial WLAN (IWLAN) developed by Siemens. The low cost, good performance (e.g., low latency, high throughput), and extensive implementation in commercial equipment make IEEE 802.11 a suitable candidate to fulfill the tight timing constraints of industrial automation. However, achieving the reliability, transparency, and stability for real-time remote control required in many applications remains a critical challenge in IEEE 802.11. due to the highly unpredictable, unreliable, and interference-prone wireless channel, which introduces delays, packet loss, jitter, throughput bottlenecks, and even loss of connectivity [48]. The presence of packet collisions and electromagnetic (EM) interference in the shared medium results in delayed or even lost control commands. While the delayed delivery of control commands to the robot breaks the transparency of the remote control system (e.g., lag between the executed remote control commands and the robot movements), the lost commands directly influence the stability resulting in a deviation from the desired trajectory.

1.3. Thesis Overview

This thesis proposes robot enhancements by virtualizing and distributing robotic services at the edge of the network and it is organized into four main chapters. Chapter 2 presents an end-to-end edge robotics system design and a real-world testbed implementation that is used in the rest of the chapters to evaluate the performance of the proposed enhancements. Chapter 3 proposes enhancements that can be offered to robotic systems through the use of MEC and Chapter 4 elaborates on how the concept of orchestration can offer enhancements for mobile robot applications. Chapter 5 discusses the role of AI in cyber-physical systems, highlighting specific aspects related to networked robots that can be enhanced with AI capabilities. Finally, in the last chapter, the thesis the conclusions, and future work are presented.

The outline of the organization of this thesis is presented as follows:

- ***Chapter 1. Introduction:*** this chapter describes the context in which this thesis is established, presenting the enabling technologies, as well as the open challenges that are going to be addressed.
- ***Chapter 2. An end-to-end edge robotics system :*** defines the design of an end-to-end edge-based robotics system. The design uses the ETSI MEC and NFV definition of applications and context-aware services as a set of autonomous and virtualized microservices that can be distributed and orchestrated in the cloud-to-robot continuum. In addition, this chapter includes detailed information about the real-world implementation of two experimental testbeds that integrate the designed edge-based robotics system using existing open source technologies. This chapter addresses Challenge C1 that is defined in the scope of this thesis. Moreover, the work from this chapter is used in the rest of the chapters as a baseline scenario for addressing the rest of the Open Challenges.
- ***Chapter 3. Enhancing robotics systems through the use of MEC:*** addresses Challenge C2, Challenge C3 and Challenges C4. In particular, this chapter is divided in two parts. The first part that addresses Challenge C2 by proposing four exemplary context-aware enhancements to optimize the end-to-end robotic system. The second part addresses Challenge C3 and Challenges C4 by providing an analysis on the potential on-device savings by means of computation offloading and showing the benefits that 5G connectivity can bring to robot manipulators when they exchange data with the edge infrastructure.
- ***Chapter 4. Orchestration of Robotic services:*** this chapter studies the feasibility of applying orchestration solutions to existing robotic services. Specifically, this chapter uses the prototype developed in Chapter 2 to apply a novel orchestration algorithm (OKpi) and address Challenge C5 that is defined in the scope of this thesis. The same prototype is used to address Challenge C6 by proposing a DLT-based federation concept for edge robotics.
- ***Chapter 5. AI-assisted control for edge robotics:*** this chapter studies the specific aspects related to networked robots that can be enhanced with AI capabilities. In particular, the chapter

starts addresses Challenge C7 of this thesis by proposing a variety of AI-agents at the application and infrastructure level that have the potential to enhance the cyber-physical systems. In order to scope the work from general to specific this chapter continues by studying one of the proposed AI-agents, namely, the Movement Prediction. As a result, the rest of the chapter proposes and evaluates FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators to address the last Challenge C8 of this thesis.

- ***Chapter 6. Conclusions & Future Work:*** presents the conclusions derived from the lessons learned with the realization of this thesis, and the future research lines to be explored.

CHAPTER 2

End-to-end edge robotics systems architecture and implementation

As the roll-out of the new ICTs has gained pace, service-oriented robotics shifts the implementation of the robotic processes, offering them as a service. A complex robotics application can be realized by a set of collaborative interoperable and platform independent services. As a consequence, capabilities like smart orchestration, dynamic scaling, and computation offloading become available. In order to implement a service-oriented robot, advances on ICTs, such as NFV and MEC defined by ETSI are playing a paramount role in satisfying the required KPIs in terms of latency, reliability, bandwidth, and scalability.

In this context, and with the aim of addressing the challenges defined within the scope of this thesis, this chapter focuses on the design and prototype implementation of an end-to-end edge-based robotics system. The design and prototype implementation of the robotics systems from this chapter are later on used in this thesis as a baseline scenario for experimental evaluation of new innovative solutions in robotic systems that have the potential to improve performance and production cost. Our design uses the ETSI MEC and NFV definition of applications and context-aware services as a set of autonomous and virtualized microservices that can be distributed and orchestrated in the system. For this, Section 2.1 motivates the design and prototype implementation of an end-to-end edge-based robotics system focusing on the benefits that current robotic systems can have by adopting the emerging ICTs. Next, we propose the exemplary designs of end-to-end edge-based robotics systems in Section 2.2). In Section 2.3 we present the prototype implementation of the above mentioned robotic systems by creating two separate testbeds. The UC3M testbed for mobile robots in Section 2.3.1 and

the 5TONIC testbed for robot manipulators in Section 2.3.2. Finally, Section 2.4 closes this chapter by summarizing the presented work.

2.1. Introduction

Industry 4.0, or the factory of the future, is offering a new way of understanding and organizing distinct manufacturing processes, pushed by the increasing interest of industrial verticals to accomplish a digital transformation of their industries. It focuses on effective interaction between humans, machinery, and products to build an efficient, agile, and smart manufacturing environment. Intelligent production lines and flexible factory are the two of its key features that are contributing to meeting the increased demands from customers [49], [50]. Some countries (e.g., Japan, France, China, and South Korea) already launched national plans of action to realize such evolution through the concept of a service-oriented industry [51].

Service-oriented robotics shifts the implementation of the robotic manufacturing processes, offering them as a service. A complex robotic application can be realized by a set of collaborative interoperable and platform independent services. As a consequence, capabilities like smart orchestration, dynamic scaling, and computation offloading become available. For example, a cloud-based service-oriented robot enables resources and other services to be shared in a convenient pay-as-you-go manner [52]. Industrial verticals are then empowered with capabilities like dynamic reconfiguration, programmability, on-demand usage, and sharing [53] aiming for minimal operational expenses, enhanced and faster production, and improvements in the time to market.

In this service-oriented concept, the physical and digital manufacturing processes are getting progressively interconnected, considered as the next generation of smart systems. Networked robotics arises as the key service to bridge both physical and digital worlds through the integration of robotic systems, computing, and communications technologies [54]. This concept truly embodies the cyber-physical integration within Industry 4.0, combining any industrial process achieved through closed-loop feedback mechanisms. The control loop starts with the robots sending sensor information and its current state to the computing infrastructure, which then closes the control loop by sending back commands in real-time. In doing so, industrial robots become software-enhanced objects that incorporate self-management capabilities and respond quickly to changes. Such an environment paves the way for novel robot applications where the control of the manufacturing processes is performed in a real-time, remote and collaborative way, optimizing the whole manufacturing process [55].

In order to implement service-oriented robots, advances on ICTs, such as edge computing and NFV, are playing a paramount role in satisfying the real-time KPIs in terms of latency, reliability, bandwidth, and scalability. Edge computing [56] offers low-latency, high bandwidth, network context-information, and location awareness in robotic services by providing computing capabilities in near proximity of the robots. Similar to cloud computing, edge computing is envisioned as a technology that relies on a virtualization environment and provides flexibility, scalability, and easy management. Motivated by these benefits, ETSI has been first to create a standardized edge environment through

MEC [15], which is supported by the virtualization framework of NFV [23] that was also pioneered by ETSI.

In the NFV framework, Virtualized Network Function (VNFs) are widely embedded in network operators' deployments. It allows them to migrate network functionalities from costly vendor hardware to general-purpose resources. Network Services are usually formed as a composition of VNFs, each providing a specific functionality of the whole service. In addition, functional network modules that compose the Network Service are split, centralizing a subset of functions into a data center. This functional split was proposed as a feature for the next generation of Radio Access Network (RAN), where the processing of all the base stations have been centralized into a data center. Pooling computational resources reduces the cost of 5G deployments and centralization enables easier coordination between next-generation base stations [57]. Applying the Network Service and functional split concepts to service-oriented robots gives the factory operator a unified and modular view of the system and extends the management capabilities beyond the networking aspects to implement a Robotics-as-a-Service (RaaS). These concepts enhance the flexibility of the infrastructure as well as monitoring and management operations, contributing to fulfilling the scalability and availability requirements.

This chapter aims at proposing an edge robotics system that integrates the aforementioned technologies to support the implementation of service-oriented robots. To do so, robotic applications for mobile robots and robotic arms are abstracted and split into virtualized functions in order to increase flexibility and reduce the burden on physical devices. This chapter departs from the design of an end-to-end edge robotic system as an example of a robotic service, incrementally validating the potential of such a solution through a prototype implementation that is realized via two experimental testbeds.

The main contributions of this chapter are:

- We design an exemplary end-to-end edge-native robotic system for mobile robots and robot manipulators.
- We implement a prototype of the edge robotic system for mobile robots, namely the UC3M testbed featuring two Kobuki robots.
- We implement a prototype of the robotic system for robot manipulators, namely the 5TONIC testbed featuring a Niryo One robot arm and multiple Radio Access Technologies (RATs).

2.2. System design

Our edge robotics system design and implementation (see Section 2.3) is motivated by realistic use cases where a fleet of mobile robots or robot manipulators are remotely controlled and coordinated to perform different tasks in a multi-access indoor environment. These use cases allow us to easily detect the new business actors that are enabled by using an end-to-end edge robotics system. Let's consider a factory floor as a potential deployment scenario for mobile robots.

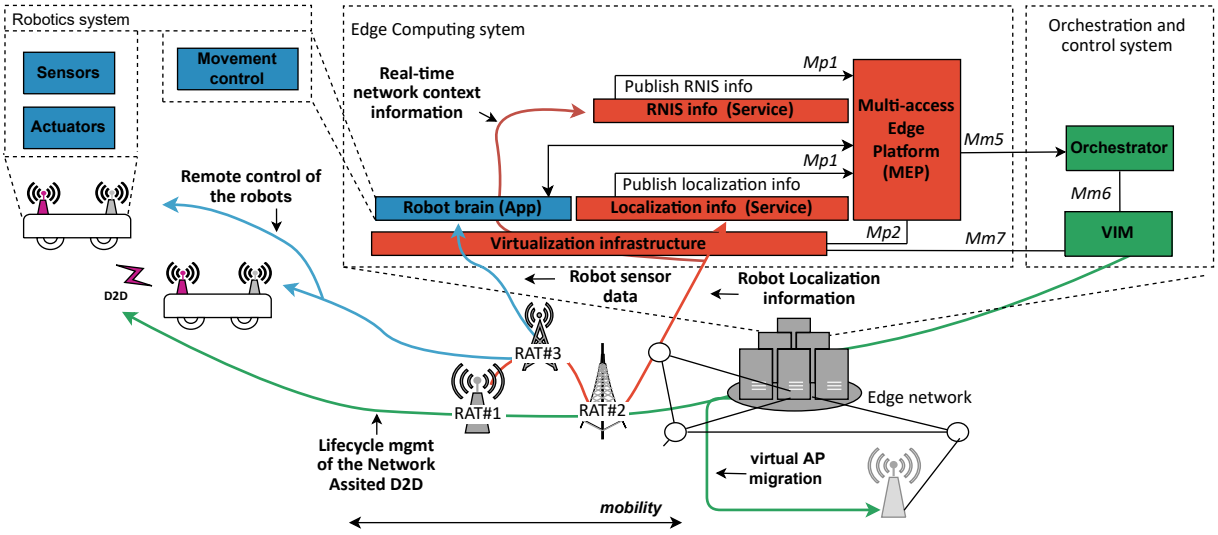


Figure 2.1. *End-to-end Edge robotics system for mobile robots.*

In this case, the infrastructure (e.g., the micro-data center at the edge) may be owned by the factory owner or even a third party that provides and manages the infrastructure for the factory owner. The automation of the factory can be achieved through a robotics service, such as video surveillance, and cleaning or transport of goods. This service is provided by a robotics service provider that delivers its applications through the infrastructure located in the factory.

In this section we propose two edge robotics system that are illustrated in Figure 2.1 and Figure 2.2 and they consists of 3 subsystems: *i)* orchestration system (green modules), *ii)* edge computing system (red modules) and *iii)* robotics system (blue modules). In general, raw sensor data are acquired by robots and sent to the edge computing system, removing the need for any data processing by the robots. Real-time navigation decisions are taken in the edge computing system to minimize the latency and share the data between different connected robots. The orchestration system provides additional services such as managing and controlling the underlying infrastructure. The difference between both proposed edge robotics systems presented in this section is the robotics system (blue modules) while the edge computing system (red modules) and orchestration system (green modules) are the same.

2.2.1. Orchestration system

The orchestration and control system (Figure 2.1 - green modules) is in charge of the application lifecycle management. It includes allocation, monitoring and enforcement, over the IT/OT convergent infrastructure while ensuring dynamism and elasticity for the robotics application. Its responsibilities are separated in two different modules: *i)* the Virtualized Infrastructure Manager (VIM) and *ii)* orchestrator.

The VIM interacts with the infrastructure and offers unified abstractions over the heterogeneous,

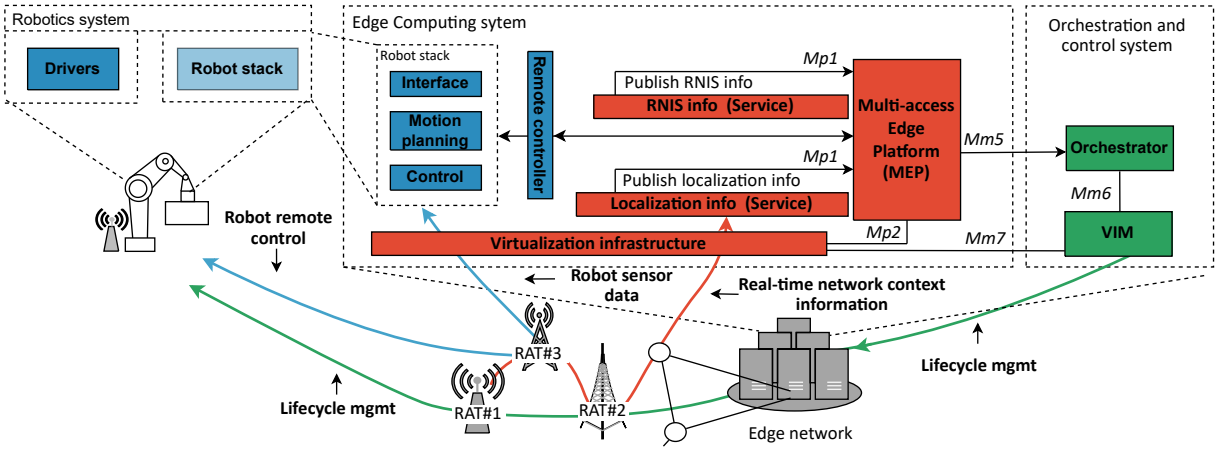


Figure 2.2. *End-to-end edge robotics system for robot manipulators.*

mobile, and volatile resources. It offers monitoring, allocation, and management of resources across the infrastructure and shares this information with the upper layers.

On the other hand, the orchestrator is responsible for *i)* storing catalog information of the robotics application components, including checking the integrity and authenticity of the application packages, *ii)* keeping records of the onboard components and enforcing the allocation, and *iii)* re-allocating components based on the information received from the robot and the edge computing system as well as from the VIM. Compared to cloud-like orchestrators, more advanced placement algorithms are required in order to select the appropriate host(s) for instantiating or migrating the components based on constraints defined by the application, such as latency, computing requirements, availability of resources and services, etc. Enforcement instructions are transmitted to the robots and the underlying infrastructure through the VIM.

2.2.2. Edge Computing system

The edge computing system shown in Figure 2.1 and Figure 2.2 (red modules) depicts the integration of applications, together with different services such as radio network information and location services. Both services are described in more detail later on. In particular, the edge computing system takes care of coordination and control of the robot movements based on network contextual information and location algorithms. The virtualization infrastructure provides compute, storage, and network resources for the applications. It also includes the basic networking capabilities for routing the traffic between services, applications, external networks, and Multi-access Edge Platform (MEP). The MEP offers API based domain for advertising, discovering, and consuming services. We can consider the MEP as a 'middleman' in charge of storing and distributing the subscription data of a service to the data subscribers via a suitable messaging protocol (e.g., MQTT [58], REST [59], DDS[60]).

Localization algorithms are run in the edge computing system as a localization service, where real-time sensor data is matched with an area of an existing map. Additionally, for situations when

the robot operates in partially or totally unknown environments, the signaling information received from the robot can be utilized to estimate its precise location. Simultaneous localization and mapping algorithms run as services. The real-time localization information is shared with the robot brain using the MEP.

The Radio Network Information Service (RNIS) provides information regarding the quality of the radio channel. Regardless of the radio access technology (e.g., 4G, Wi-Fi, Bluetooth, etc.), the RNIS monitors the connectivity and publishes real-time information about the signal strength, MAC layer parameters, packet loss, etc., of each robot. The use of the RNIS service can facilitate the robot's mobility in an indoor environment and optimize robot navigation. For example, based on the signal level and MAC layer parameters the Robot brain can detect quality degradation on the wireless channel and adapt the robot speed accordingly.

2.2.3. Robotics system

As mentioned before the differences between the two proposed edge robotics systems illustrated in Figure 2.1 and Figure 2.2 are in the robotic system and how the robot functionalities are split into different modules. While the edge robotics systems for mobile robots present a more general functional split, the edge robotics systems for robot manipulators brakes down the robot manipulator into 5 logical modules. In this section, we present the description of each of the modules that compose both robotics systems.

Mobile robots

The robotics system for mobile robots (Figure 2.1 - blue modules) is composed of modules distributed between the robots and the edge computing system. In our proposed design, the robots act only as sensors and actuators. They have embedded within them only the minimal components in charge of *i)* reading data from the sensors (e.g., odometry, camera, lidar), and sending them to the robot brain, and *ii)* executing navigation instructions received from the robot brain. The robot brain resides in the edge computing system and is responsible for the coordination and navigation of the robots. Control algorithms are run in the robot brain, where real-time network contextual information and localization data are used to adapt robot operations. The robots can be equipped with the most suitable RAT (e.g., Wi-Fi, 4G, 5G) based on the use case requirements. Additionally, D2D connectivity is envisioned between the robots to offer higher data rates, lower transfer delays, and improve power efficiency.

Robot manipulators

Figure 2.2 illustrates the main modules that comprise the robotic system for robot manipulators. The robot drivers act only as sensors and actuators (i.e., bare metal devices with minimal resources), while the robot stack implements the control, navigating, and interfacing operations on top of the robotic

systems. The robot drivers always remain in the robotic manipulator but the robot stack can be distributed across the robot or the edge computing system. Accordingly, the edge robotics system for robot manipulator is decomposed into five logical modules, described as follows:

- The robot drivers are low-level software interfaces that control and monitor the specific robotic arm hardware. Therefore, the robot drivers interact directly with the robot hardware and are responsible for *i)* executing in the robotic arm's hardware of any command received from the control module; and *ii)* making available sensor data and operational states to the control module.

- The control module implements generic control-loop feedback mechanisms used for robot manipulation. It takes a trajectory, composed of position commands, as input and runs a control loop, following a given frequency, towards the Drivers module. This module defines a set of controllers that are used by the remaining modules for robot manipulation.

- The motion planning module receives a movement command (e.g., go to position) and computes the trajectory for the robotic arm. The created trajectory consists of path commands, being sent to the control module in order to move the robotic arm from its current position to the desired one.

- The interface module is a high-level abstraction for the core robot functionalities. The main goal of this module is to hide the complexity by providing the interaction through high-level commands. It acts as the gateway between the user application and the robotic arm.

- The remote controller module enables remote control of the robot manipulator. It provides a set of tools such as a web interface and/or joystick that implement a close-loop mechanism for robot manipulation.

Figure 2.3 depicts the interactions between the different modules of the edge robotic system. When a user needs to remotely control a robotic arm, it issues a move joints (step 1) manipulation command using the selected interfacing device (e.g., joystick, AR/VR). The move joints command is sent to the Interface module which offers a custom-made interface (e.g., Python or REST API), translating it into a robot-specific movement command. Then, the Interface module sends the movement command (step 2) to the motion planning module. When the movement command is received, the motion planning module performs several command validations and generates the trajectory path consisting of an array of position commands. For each position command, each joint is given a specific position, velocity, and acceleration. Once the control module receives the trajectory path (step 3), it runs a control loop against the driver's module. The control loop starts with the driver's module sending the joint states (step 4) of the robotic arm to the control module. Next, the control module interpolates the received trajectory path to get the next position command. The control loop is then closed when the control module sends the position command (step 5) to the driver's module.

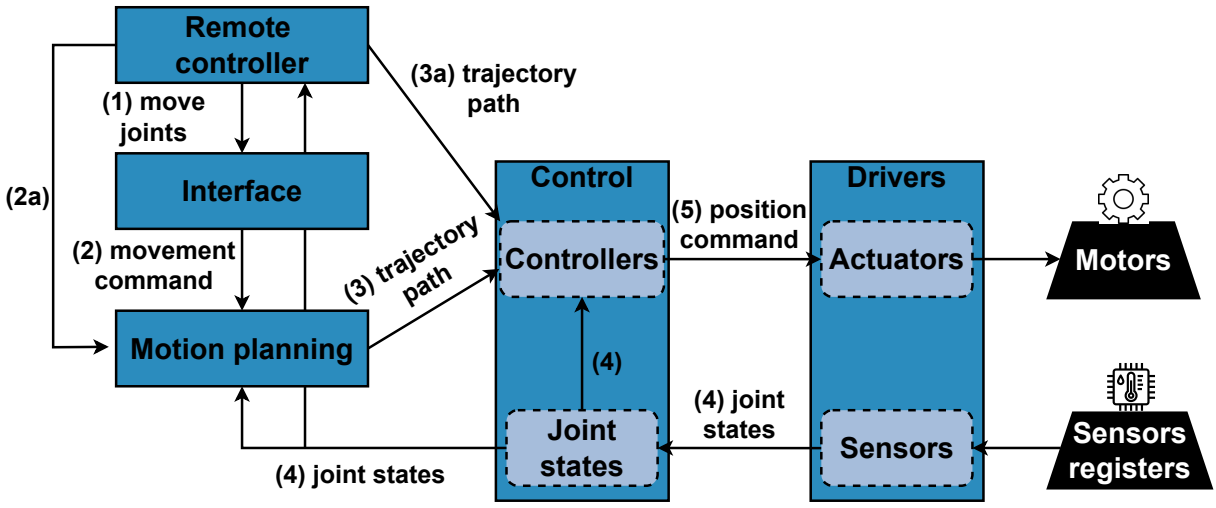


Figure 2.3. *Sequnce Message Diagaram for remote control.*

2.3. Deployment of real-world testbeds

To be able to evaluate the concept of edge networked robotics, exploit the benefits that it can bring to robotic systems, and address the open challenges that are defined in the scope of this thesis, we developed two separate testbeds where we deployed a prototype of the exemplary robotic systems designed in Section 2.2. The first testbed was implemented in the university hallways of Universidad Carlos III de Madrid and its objective is to address the **C2**, **C5** and **C6** while the second testbed was implemented in TONIC laboratory in Madrid (Spain) and its objective was to address **C3**, **C4** and **C7**.

2.3.1. The UC3M testbed

The UC3M experimental testbed shown on Figure 2.4 is build along the hallways in the telematics department in Universidad Carlos III de Madrid where we deployed a prototype of the edge robotics system for mobile robots illustrated on Figure 2.1. The tesbed is composed of: *i)* 6 Wi-Fi Access Points from which 4 are ASUS WL500G Premium v1 APs running OpenWrt 18.06.2 [61] (AP1, AP2, AP4 and AP5) and 2 are MiniPCs, with 4 CPUs and 8GB of RAM running hostapd [62] (vAP3 and vAP5); *ii)* 1 MiniPCs with 4 CPUs and 8GB of RAM acting as local fog server; *iii)* 1 PowerEdge C6220 server with 94GB of RAM and 16 CPUS, acting as edge server; *iv)* 1 PowerEdge R840 Rack Server7 with 94GB of RAM and 16 CPUs, acting as cloud server; and *v)* 2 Kobuki Turtlebot S2 robots equipped with Wi-Fi antennas, a laptop with 8 GB of RAM and 5 CPUs and a RPLIDAR A2[8] lidar.

The six APs and the local server are deployed along two corridors of the telematics department in the Universidad Carlos III de Madrid building (see Figure 2.4), while the edge and cloud server are located in different buildings, all of them interconnected by a 10 GB/s ethernet connection. The Turtlebot maximum speed is 0.75 m/s, while its minimum speed is 0.1 m/s with a ROS control fre-

[8] https://www.slamtec.com/en/Lidar/A2 [Accessed: Jul. 14, 2022]

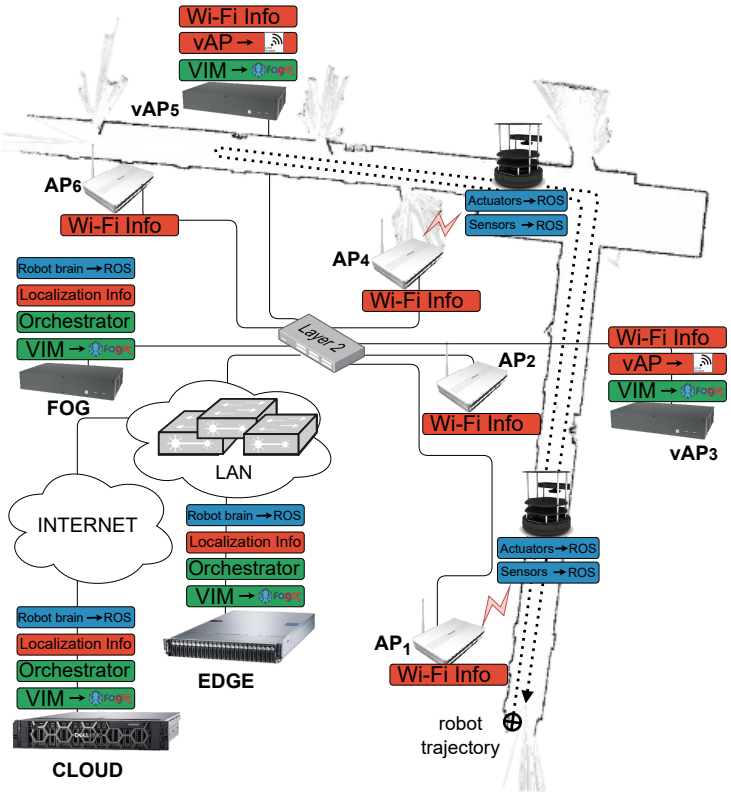


Figure 2.4. *Illustration of the UC3M testbed composed of two mobile robots, cloud, edge and fog servers and multiple Access Points that were distributed in the university hallway.*

quency of 10 Hz (i.e., 100 ms). The sampling frequency for reading the odometry sensor data from the robot's wheels is 16.6 Hz (i.e., 60 ms) and the rotating frequency of the lidar is 10 Hz with a guaranteed 8-meter ranger distance.

The robotics system is based on Robot Operating System version 1 (ROS-1)[9] and is distributed across the robot's and the fog, edge, or cloud servers. Basic robot sensors (lidar, odometry) and actuators handling are provided by the ROS-1 nodes deployed in the robot's laptops. The robot brain can be placed in the fog, edge, or cloud server and hosts the ROS-1 navigation stacks for both robots together with the static map of the covered testbed indoor environment.

The robots are connected to the infrastructure through a Wi-Fi link. We decided to use IEEE 802.11 technology mainly because of the high bandwidth, low latency, and widespread use that make it suitable for the requirements of our time-sensitive robotics application. An application implements the Wi-Fi access point capabilities and can be deployed on-demand as a hostapd docker container. It is important to mention that all the 6 APs are configured to use IEEE 802.11r [63] over the Distributed System (DS) in order to permit fast and secure handovers. Since we use Wi-Fi as our RAT, a RNIS service, namely, Wi-Fi information service, is deployed as a docker container together with the Wi-Fi

[9]Widespread framework for developing and testing multi-vendor robotics software.

access point and provides real-time context information about the connected robots. In addition, a location service can be hosted by the fog, edge, or cloud server. This service is implemented as a ROS node and provides probabilistic robot localization based on the lidar sensor.

Regarding the orchestration system, we implemented a custom-developed orchestrator that can be deployed as a container in the fog, edge, or cloud server. The orchestrator implements the base life-cycle management functionalities such as instantiation and termination of the robot application and services. The 3 MiniPCs together with the edge and cloud servers run open-source project Eclipse Fog05[10] as VIM, which embodies the principles described in Section. 2.2. Eclipse Fog05 addresses the requirements and characteristics of edge computing by providing a decentralized infrastructure that logically unifies computing, networking, and storage fabrics end-to-end, while addressing the challenges imposed by resource heterogeneity (e.g., virtual machines, containers, native applications). The docker plugin of Fog05 allowed us to use docker as virtualization technology for running all the components in the UC3M testbed.

2.3.2. The 5TONIC testbed

The 5TONIC experimental testbed shown on Figure 2.5 is build in 5TONIC[11] laboratory in Madrid (Spain) where we deployed a prototype of the system for robot manipulator presented in Section 2.2.2. The testbed is composed in three sections: *i)* laboratory floor (Figure 2.5 top), *ii)* RAN (Figure 2.5 middle) and *iii)* edge data center (Figure 2.5 bottom).

The Laboratory floor is composed of 6-axis Niryo One robotic arm that is equipped with Raspberry Pi 3 with 1.2 GHz 64-bit CPU, 1GB RAM, and a Wi-Fi antenna. The robot's maximum speed is 0.4 m/s for the steeper axes and 90°/s for the servo axis with configured ROS control frequency of 50Hz (i.e., 20 ms). The sampling frequency for reading the sensors registers (joint states) is configured at 50Hz (i.e., 20 ms). The control and the sampling frequencies are two independent and non-alignment open-loops. The Niryo One is an open-source collaborative robot designed for research in robotics and industry 4.0. It has a similar design and functionalities as 6-axis industrial robot. In addition, the low latency requirements of remotely controlled robot manipulators can be fulfilled with Niryo One robotic arm which makes it a good candidate for the studies presented in this thesis. In addition, in the laboratory floor we deployed 3 MiniPCs with 4 CPUs and 8GB of RAM acting as local fog servers and commercial 5G/4G CPE that provides connectivity towards 4G and 5G NSA networks.

The 5TONIC RAN provides connectivity to the 5TONIC edge data center using Wi-Fi, 4G and 5G NSA. Wi-Fi connectivity the robot relies on IEEE 802.11n compatible device working on the 2.4 Gz and 5 Hz bands, while for 4G and 5G connectivity the robot is connected via Ethernet to 5G/4G CPE (HUAWEI 5G CPE Pro Baloong 5000) that provide connectivity towards 4G and 5G networks. 5TONIC provides an implementation of 5G NSA (BB630 baseband and Advance Antenna System AIR 6488) and 4G (BBU5216 baseband and RRU 2203 with integrated antenna) provided by Ericsson, where teh

[10] Fog05 project: https://fog05.io/ [Accessed: Jul. 14, 2022]
[11] https://www.5tonic.org [Accessed: Jul. 14, 2022]

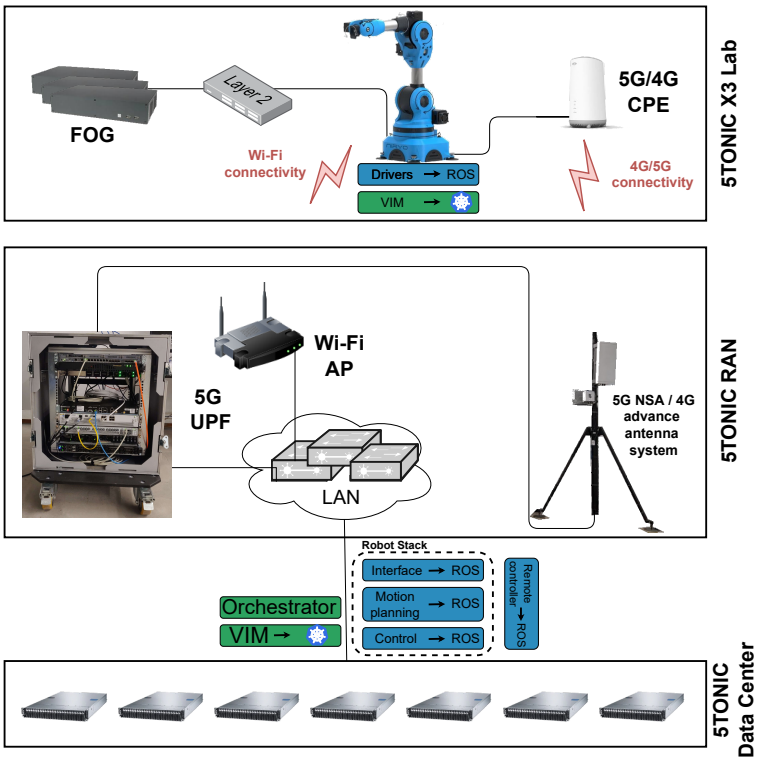


Figure 2.5. *Illustration of the 5TONIC testbed composed of robot manipulator, edge and fog servers and Wi-Fi, 4G and 5G NSA radio access technologies.*

5G UPF is installed in one half-size rack (5G UPF on Figure 2.5) which hosts IRU 8846, a Baseband BB6630, UPF server, app server, a switch and a firewall. Finally, the 5TONIC RAN also has a Wi-Fi AP working on 2.4 Gz and 5 Hz bands. Therefore, the communication between the robot and the 5TONIC edge data center comprises of wireless link, from the robot to the referent RAT, and a wired network, from the RAT to the edge data center. In the edge data center we used an PowerEdge C6220 server with 94GB of RAM and 16 CPUs to deploy a prototype of the edge robotics system for robot manipulator presented in Figure 2.2. It is worth mentioning that in the 5TONIC testbed the 5G UPF and 5G NSA/4G advance antenna system are Ericsson proprietary HW where the RNIS service was not implemented.

The robotics system is based on Robot Operating System version 1 (ROS-1) and is distributed across the robots and the edge server. Basic robot sensors and actuators handling are provided by the drivers ROS-1 nodes deployed in the robot's Raspberry Pi. The remote controller is deployed in the edge server, while the robot stack (interface, motion planning, and control) can be deployed in the robot or offloaded to the edge server. The robot stack contains the ROS-1 packages[12] used on Niryo One robot that are developed by the manufacturer.

Regarding the orchestration system, we used Kubernetes as orchestrator and VIM for managing the edge server and the robot manipulator. Kubernetes is a portable, extensible, open-source plat-

[12] https://github.com/NiryoRobotics/niryo_one_ros [Accessed: Jul. 14, 2022]

form for managing containerized workloads. It provides a Platform-as-a-Service offering: *i)* service discovery and load-balancing, *ii)* storage orchestration, *iii)* automated rollouts and rollbacks, *iv)* autonomous placement, *v)* self-healing and *vi)* secrets and configuration management. We used docker as virtualization technology for running all the modules in the 5TONIC testbed.

2.4. Conclusion

This chapter addresses **C1** of this thesis by presenting the design and prototype implementation of two exemplary edge-based robotic systems that offer an abstraction of the robotic services where robots can have access to information from different systems. As a result we deployed two testbeds: *i)* the UC3M testbed for mobile robots, and *ii)* 5TONIC testbed for robot manipulators. These testbeds were built with open-source software (e.g., ROS, Docker, LXD, Fog05, Hostapd, Python, and Linux) and low-price research hardware (e.g., Turtlebot2 and Niryo One) that makes it easy for the research community to recreate them. The main goal of these two testbeds is to serve as reference scenarios that will be later on used in this thesis for:

- studying the potential returns and performance improvements for verticals in Chapter 3.
- applying innovative orchestration solutions in Chapter 4.
- studying a forecast-based recovery mechanism for real-time remote control of robot manipulators in Chapter 5.

CHAPTER 3

5G-enable enhanced edge robotics through contextual information

The adaptation of network-assisted robots in Industry 4.0 is inseparable from the advent of 5G technologies, such as 5G connectivity and edge computing. 5G networks are architected to simultaneously support different types of real-time network service profiles in the shared infrastructure through the Ultra-Reliable Low Latency Communication (URLLC). Together with edge computing, they provide a context-aware communication link with low end-to-end (E2E) latency, low jitter, and localization awareness to robotic services. However, the potential of network-assisted robots to support context-aware services and RATs for managing automation or computing best real-time decisions to achieve dynamic adaptation is still to be exploited.

To explore this opportunity, and with aim of addressing the **C2**, **C3** and **C4** defined within the scope of this thesis, this chapter presents a study of exemplary enhancements for robotic applications where MEC is used. These enhancements include a mobile robot control algorithm that adapts the robot speed to the real-time radio information, a virtual access point offloading algorithm that uses the context-aware services available at the edge to extend the wireless range of mobile robots, and a D2D robot communication to improve the robot coordination. In addition, this chapter studies the capability of the end-to-end edge robotics system presented in Chapter 2 to offer potential savings by means of computation offloading and the impact of different wireless channels (e.g., 5G, 4G and Wi-Fi) to support the data exchange between robot manipulators and its virtual components.

The rest of this chapter is organized as follows. Section 3.1 introduces the main concepts that are used in this chapter and present the main contributions of the chapter. Section 3.2 reviews the re-

lated work and discusses the benefits of integrating MEC for robotics systems. In Section 3.3, above-mentioned context-aware enhancements for mobile robots are presented. In Section 3.4 we study the VNF resource consumption and impact of different RATs in remotely controlled robot manipulators. Section 3.5 discusses the potential research challenges and future research directions. Finally, Section 3.6 summarizes our findings and draws the conclusions from this chapter.

3.1. Introduction

Historically, Information Technology (IT) and Operational Technology (OT) have been two separate domains addressing distinct scenarios. While the former focuses on providing services in a cyber-only environment (e.g., content-delivery networks, video-on-demand), the latter targets applications in a cyber-physical environment (e.g., robotics systems, industrial automation) where the virtual and the real worlds are closely entangled [64]. With these different visions in mind, it is understandable that IT and OT have evolved in the past in different, yet related, directions.

As of today, IT is primarily characterized by large-scale and multi-tenant deployments (e.g., data-centers), short life cycles of homogeneous resources (e.g., servers, mobile phones), and the usage of best-effort technologies (e.g., Ethernet, IP) coupled with high-availability techniques (e.g., load-balancing, replicas) [65]. In contrast, OT is heavily characterized by confined and well-controlled environments (e.g., manufacturing plants), long life cycles of heterogeneous resources (e.g., assembly line), hard-real time guarantees (e.g., closed control loop), ad-hoc components (e.g., microcontrollers) and reliable technologies (e.g., Profibus, CAN bus) [66]. In recent years, however, the OT has started to move towards a more IT-oriented approach driven by the proven benefits of cloud computing in terms of flexibility and cost reduction.

One clear example is cloud robotics [5], which aims to integrate cloud computing resources into the robotics systems so as to increase the reconfigurability as well as decrease the complexity and cost of the robots. Moreover, by offloading the control logic from the robot to the cloud, it is easier to share services and information from various robots or agents to achieve better cooperation and coordination. However, the centralized nature of cloud computing poses critical challenges: cloud facilities usually reside far away from the robots (i.e., separated by one or more uncontrolled transit networks), making it hard to achieve high-bandwidth, low-latency, and bounded jitter [8]. These drawbacks prevent time-sensitive tasks, including real-time remote control, to be fully supported by the cloud computing substrate.

Edge [67] computing and fog [68] computing have emerged as paradigms to alleviate these problems by placing computing and storage resources deep into the network. This enables applications to execute closer to the users (i.e., the robots), resulting in more predictable communication and overall better system performance. Real-time information about user wireless connectivity is expected to be available at the edge, enabling the dynamic adaptation of the application's logic to the actual status of the communication (e.g., radio channel) [69]. This network contextual information can be used to achieve optimal operation and maintenance of robotic tasks, and energy consumption reductions.

ETSI MEC defines a set of exemplary context-aware services. For example, the Radio Network Information Service (RNIS) [18] provides radio network-related information, such as up-to-date radio network conditions, measurement and statistics information related to the user plane, and information related to users served by the radio nodes. While the original RNIS was designed for 3GPP networks, a new service is being defined to cover also Wi-Fi networks [19]. Another example is given by the location service [20] which provides location-related information about the users (e.g., all of them or a subset) currently served by the radio nodes. The location information can be geolocation, Cell ID, etc. Finally, the Bandwidth Manager service [70] allows the allocation of bandwidth to certain traffic routed to and from MEC applications and the prioritization of certain traffic. Additional services can be then defined upon necessity based on the same MEC framework.

Additionally, edge computing contributes to a reduction in the E2E latency, increases bandwidth savings in the transport network, and improves the availability of the whole service. It reduces the cost of robots by offloading computation modules in an edge server. Still, the sensor data can be processed locally in the edge without extra jitter or packet loss caused by the communication towards the cloud [71]. By facilitating local and real-time decisions, the accuracy of control processes in the robotic systems is improved with minimum cost. As a consequence, edge computing and fog computing are very well-positioned for overcoming today's cloud robotics challenges.

The adaptation of edge computing for robotic systems is inseparable from recent advances in wireless technologies such as 5G and Wi-Fi 6. However, while wireless is a must for mobile robots like Autonomous Guided Vehicles (AVGs), the implementation of wireless connections for robots manipulators also has many advantages such as greater flexibility, reduction of installation and maintenance costs, ease of scale, and less personnel exposure to hazardous situations [72]. Robotic vendors will decide whether to use wireless technologies in the licensed spectrum, such as 5G NR [73]; or in the unlicensed spectrum, such as IEEE 802.11 [74]. With the emergence of 5G, robotics vendors recognized the opportunity for a unified communication interface that can support the requirements of any type of robots through three main service profiles: *(i) enhanced Mobile BroadBand* (eMBB); *(ii) massive Machine-Type Communication (mMTC)*; and *(iii) Ultra Reliable Low-Latency Communications* (URLLC). 5G provides guaranteed QoS for time-critical applications as a built-in feature. Finally, network softwarization and virtualization are natively explored by 5G to support network slicing capabilities, allowing the creation of logically isolated and dedicated networks over the same and shared infrastructure tied to the specific requirements of each service and/or application.

Following the above mentioned opportunities for robotic systems, this chapter studies the practical feasibility of enhancing robotic systems through the use of MEC and 5G connectivity. For this purpose, this chapter builds on top of the UC3M and 5TONIC testbeds presented in Chapter 2 and presents exemplary context-aware enhancements for mobile robots as well as a detailed analysis of the possible resource consumption savings and performance impact of different RATs in remotely controlled robot manipulators.

The main contributions of this chapter are:

- We implement and experimentally evaluate a remote control algorithm that adapts the speed

of the robot based on the real-time radio information available in the MEC.

- We implement and experimentally evaluate a virtual access point offloading algorithm that extends the wireless range of mobile robots.
- We implement and experimentally evaluate robot-to-robot D2D communication for improving coordination.
- We experimentally evaluate the computation offloading possibilities for robot manipulators.
- We experimentally evaluate the performance of potential wireless technologies to support remote control of robot manipulators.

3.2. Related Work

Moving resource-demanding computation (used to build specific applications) to the cloud makes traditional robotics systems more service-oriented, interoperable, distributed, and programmable. However, such cloud offloading implies problems regarding availability [75], low latency [76], bounded jitter, high bandwidth [77], [78], security, and data filtering [79], which limit the use of cloud robotics for applications that require real-time sensitivity and precision. Edge computing and fog computing with their implementations have emerged to fill this gap by placing computational resources closer to the user. While ETSI provides an implementation for edge computing through a framework for MEC [80] over static substrates (e.g., data centers or servers), the OpenFog [81] working group of the Industrial Internet Consortium (IIC) [22] extends this definition to include less powerful, constrained and mobile computational substrates, including the end-user devices. The IEEE Communication Society adopted the OpenFog concept for fog computing through the IEEE 1934 standard [82]. ETSI MEC and IIC with their main features have the potential to provide major benefits for robotics systems. These benefits, together with some recent work for the efficient integration of edge computing and fog computing into robotics systems, are summarized in this section. It is worth mentioning that some of the existing studies do not specifically mention the use of IIC or MEC for robotics systems, but their work complies with the corresponding reference architectures.

Existing studies on using MEC and IIC for robotics systems mainly focus on the low-latency and computation offloading features that offer the potential to overcome the real-time constraints of cloud based solutions. By placing the time-sensitive robotics applications at the edge of the network, MEC and IIC can ensure high computation capabilities while leveraging low-latency communication and high bandwidth. Most of the recent experimental work focuses on distributing computation between the robots and the edge of the network, in particular, deep object recognition and grasp planning [83], visual odometry [84], [85], real-time control [86], [87], [88], [89], [90] and object detection [91], [92]. Besides the experimental work, several studies have motivated the application of MEC and IIC for different robotics systems by proposing collaborative architectures (e.g., healthcare [93], autonomy [94], tele-surgery [95], manufacturing [96–98] and multi robot systems [99]). The MEC and IIC enabled implementations allow the co-location of independent applications on a shared edge/fog node through

virtualized abstraction. As presented in this section, some of the existing studies where the main focus is on the low-latency and computation offload [93–98], [83], [86], [89], [90], [100], [99], [87, 88, 101] also adapt the concept of virtualization that allows robotic services to reuse the surrounding hardware and deploy applications on demand.

The close proximity of MEC and IIC has been studied in some recent existing experimental studies to offload the location-based robotics services from mobile robots with limited computational and low-energy resources [84], [85], [101–103]. Additionally, [96–98], [104], [99] elaborate on the reduced network pressure and improved security and privacy of the robot sensor data that can be offered by restricting the access within a trusted private infrastructure. For security, cloud-fog-edge security risk prediction method has been proposed in [105] to meet the current needs of the industrial Internet systems. In [106], the authors propose a fog-driven method to detect GPS spoofing of unmanned aerial vehicles.

IIC and MEC have the potential to enable robots to communicate directly in a D2D fashion through a network-assisted approach [43]. Tight coordination and cooperation between different Radio Access Technologies (RATs) in the edge can be used to enable multi-RAT communication in robotics systems. Moreover, the network under these two implementations is expected to be completely context-aware [107], allowing robotics applications to obtain real-time information for the radio network condition. Based on this information, robotics applications can adapt their operations or optimize the network to improve the user experience. However, as this section presents, most of the recent studies that address multi-RAT communication [96], [108], [100], [99] and network contextual information applicability to robotics systems [95], [109] focuses on proposing IIC and MEC compliant communication architectures. For what concerns experimental studies, in our previous work [86], we used the locally available context information to adapt the driving speed of a mobile robot. In [110], the end-to-end reliability of mission-critical traffic was investigated in softwarized 5G networks where a multi-RAT and multi connectivity access network is considered.

Relation with the work in this chapter: With the growing need for context-aware real-time collaboration, especially under mission-critical scenarios in robotics systems, it is strongly desirable to experimentally evaluate the implementations of context-aware platforms such as MEC and the suitability of the currently available IT/OT technologies in order to truly materialize edge computing and fog computing visions. However, as mentioned above, few existing studies are adapting MEC and network context information focused on improving the overall performance of robotics systems. Therefore, this chapter designs and experimentally validates edge robotics applications that leverage on MEC, 5G connectivity and real-time context-aware collaboration to optimize the performance of the end-to-end robotics systems.

3.3. Context-aware enhancements for mobile robots

This section presents the experimental evaluation of exemplary context-aware enhancements for mobile robots. The section starts by presenting the UC3M experimental setup that was used for prototyp-

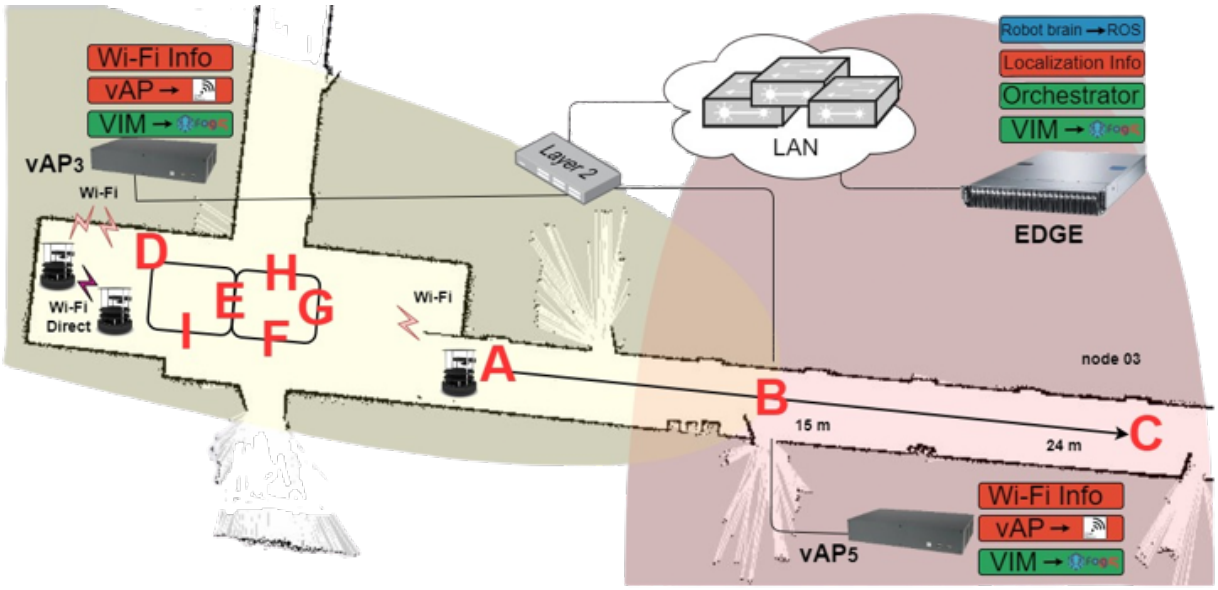


Figure 3.1. *Grid map of the deployed experimental testbed obtained using lidar and odmoetry data. In the map two separate experimental scenarios are presented, namely (A, B, C) and (D, E, F, G, H, I). The Wi-Fi coverage in the hallway is denoted with yellow and red ovals.*

ing and experimentation (see Section 3.3.1), followed by the Wi-Fi delay and signal characterization of the experimental testbed. Three exemplary context-aware entanglements are presented in this section, namely adaptive speed control, virtual access point offloading, and network-assisted D2D robot coordination. For each of these entanglements, the experimental methodology is examined together with the achieved results.

3.3.1. Experimental setup

To design, develop and evaluate the real-time context-aware enhancements for mobile robots, we used the north corridor of the experimental testbed developed in the UC3M hallway (see Section 2.3.1). In such an environment, we have reused the following components that are presented in Figure 3.1: *i)* three mini PCs namely vAP3, edge and vAP5, interconnected by a 10 GB/s Ethernet connection and *ii)* two Kobuki Turtlebot S2 mobile robots. Two of the mini PCs (vAP3 and vAP5 in Figure 3.1) are equipped with IEEE 802.11ac capable interfaces in order to ensure Wi-Fi coverage of the testbed area. The mobile robots are equipped with a laptop that has 2 Wi-Fi antennas. One antenna is used for robot-to-infrastructure communication and one for robot-to-robot communication. Additionally, a lidar was mounted on the robots in order to perform a 360-degree omnidirectional laser range scanning.

We used a version of the edge robotics system for mobile robots shown in Figure 2.1 on this testbed. The robotics system is based on ROS-1 and is distributed across the robots and the edge computing system. Basic robot sensors (lidar, odometry) and actuators handling are provided by the ROS-1 nodes deployed in the robot's laptops. The robot brain is placed in the edge server and hosts

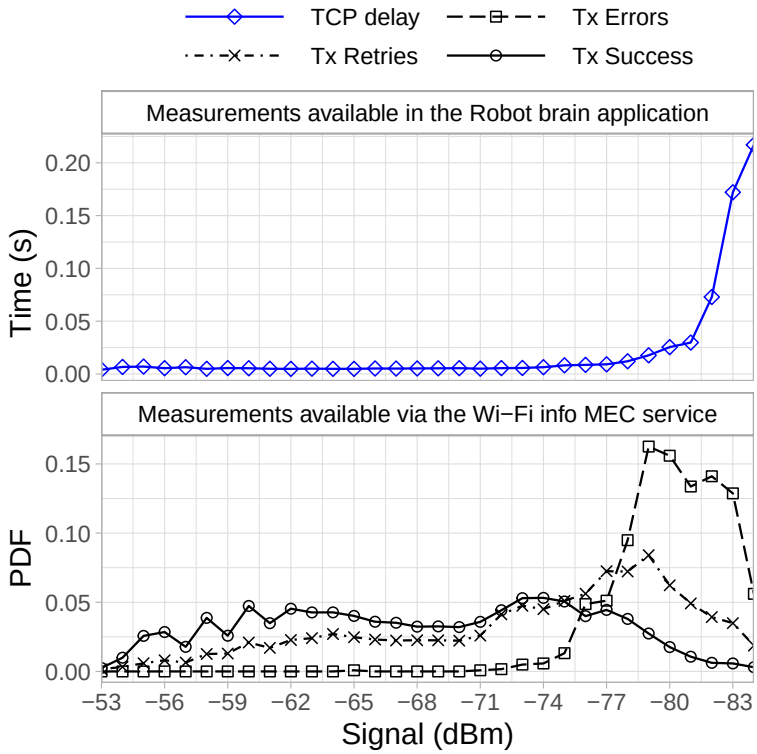


Figure 3.2. *Signal and Delay characterization.*

the ROS-1 navigation stacks for both robots together with the static map of the covered testbed indoor environment.

The robots are connected to the infrastructure through a Wi-Fi link. The Wi-Fi access point capabilities are deployed on-demand as a hostapd [62] docker container and they are configured to use IEEE 802.11r [63] over Distributed System (DS) in order to permit fast and secure handovers. A Wi-Fi information service is deployed as a docker container together with the Wi-Fi access point and provides real-time context information about the connected robots. In addition, the location service is hosted by the edge server. Regarding the orchestration system, we used the custom orchestrator deployed as a container in the edge server and for VIM we used fog05. Section 2.3.1 presents the details about the hardware specifications of the testbed (e.g., mobile robot maximum speed, lidar range, mini PCs capabilities) or the software implementation of the edge robotics system for mobile robots.

3.3.2. Delay and Signal characterization

This section aims at characterizing how the Wi-Fi signal quality impacts the delay in controlling the robot as perceived by the robot brain. Indeed, the publish-consume mechanism for exchanging data between the Robot brain and the ROS components is based on TCP. This means that any transmission failure occurring on the Wi-Fi channel (Layer 2) will trigger a retransmission at the TCP level (Layer 4), thus introducing an undesired delay in the closed-loop mechanism. Indeed, additional delays in

the delivery of the odometry sensor data result in longer reaction times in the Robot brain. Similarly, additional delays in the delivery of the movement instructions degrade the smoothness and precision of the driving. Such characterization is therefore necessary in order to adapt the closed-loop to also consider the Wi-Fi signal.

To that end, we performed 10 experiment runs where the robot brain was driving the robot at the minimum speed (0.1 m/s) and 10 experiment runs at maximum speed (0.75 m/s) using the (A, B) experimental scenario shown in Figure 3.1. Each run consists of the robot brain driving the robot on a straight line for 15 m as shown in Figure 3.1 (see (A, B) segment). The starting position of the robot is location A, connected to the vAP3 and approximately 7 meters away from it having a thin office wall (approximately 15 cm) separating the two. After having traveled for 15 m, the robot reaches location B and stops. During the driving, an additional thicker wall (approximately 25 - 30 cm) separates the robot from the vAP3. At the end of the driving, the robot is approximately 22 m away from the vAP3. All the edge robotics system components are synchronized and share the same time reference for accurate measurements. Throughout the duration of the experiment, we recorded in the robot brain the Wi-Fi information obtained via the Wi-Fi information MEC service, while on the robot itself we measured the delay in receiving the movement instructions.

The obtained data from both experiments are analyzed and aggregated to generate the results presented in Figure 3.2. Note that the results shown here are specific for our test-bed, and therefore can only be used as a particular realization that we use later to validate and evaluate the benefits of edge robotics. In overall, Figure 3.2 characterizes the quality of the Wi-Fi channel covering our experimental area. Regarding the measurements available via the Wi-Fi information MEC service, the MEC: Tx Success line shows the probability density function (PDF) of all the downstream frames successfully transmitted (from the access point to the robot) over the measured signal level in dBm. Similarly, MEC: Tx Retries shows the PDF of the downstream frames retransmissions. It is worth highlighting that Wi-Fi employs an automatic retransmission mechanism in case of packet transmission error, where frames are retransmitted up to 7 times, and if none of the retransmissions succeeds, a frame loss occurs. MEC: Tx Error shows the PDF of the failed transmissions.

It can be seen that for high dBm signal values (i.e., good signal level) the probability of successful frame transmission maintains a difference proportional with respect to the number of retransmitted frames. This is due to the fact that frame retransmissions constantly occur in Wi-Fi networks because of its best-effort design principle. For lower signal strengths (below -71 dBm), the probability of having a failed transmission increases. Such probability becomes drastically higher than the probability of successful transmission at signal levels lower than -77 dBm (it is actually evident that below -80 dBm it is very hard to have a successful transmission). TCP delay measurements (shown in the top graph of Figure 3.2) confirm this, with values as high as hundreds of milliseconds.

3.3.3. Adaptive speed control

In this section, we present an exemplary adaptive speed control algorithm witch aims to show how the edge-controlled robotics paradigm improves current cloud robotics techniques toward the industry demands of high speed and high precision in robotics applications. To that end, we have designed an experiment that was using the (A, B) experimental scenario shown in Figure 3.1 and described in the following.

Experimental methodology

The mobile robot is controlled in a closed loop by the robot brain application. The closed-loop starts with the robot brain (running in the edge server) sending movement commands to the motor drivers (running on the robot) using ROS messages, published in a specific topic devoted to movement commands. The movement command consists of a tuple (speed, distance), where the speed parameter presents the velocity that the robot should maintain while driving, and the distance parameter represents the distance that should be reached upon receiving the movement command. Therefore, the distance parameter presents the movement granularity instead of the final driving destination. Upon receiving a movement parameter through the wireless link, the motor's driver initiates the movement in the robot's wheels. The movement is uninterrupted for a length equal to the received distance parameter with a constant velocity equal to the received speed parameter. The loop is then closed by the robot continuously sending back the odometry sensor data to the robot brain application. The brain analyzes and combines the odometry data together with the Wi-Fi information provided via a MEC service by the Wi-Fi MEC application. The result of the brain algorithm is a new (speed, distance) tuple, which will serve as input to the next turn of the closed loop.

The experiment runs are similar to the one explained in Section 3.2 where the robot brain drives the robot on a straight line for 15 m in the (A, B) segment as shown in Figure 3.1. The main difference is that the robot brain drives the robot in accordance with the closed-loop mechanism that is described in this section. The robot is connected to vAP3 during the driving and it stops when it reaches location B which is approximately 22 m away from the vAP3.

Adaptive speed control algorithm

Based on the results provided in the previous section (Section 3.3.2), next, we present the design of a control algorithm that is able to adapt the robot's driving speed based on the Wi-Fi information service. The aim of the algorithm is to obtain a displacement accuracy similar to the one obtained while driving at the lowest speed while reaching the target destination faster. Through this algorithm we showcase the benefits of consuming context information for controlling the robot, nonetheless, we acknowledge that more advanced and optimal algorithms than the one proposed in this section can be eventually designed.

Algorithm 1 Adaptive control speed algorithm

```
1: info ← GetCurrentWi-FiInfo()
2: buffer ← buffer.removeOldestWi-FiInfo()
3: buffer ← buffer.add(info)
4: signalLevel ← buffer.average()
5: if signalLevel ≥ -71 dBm then
6:     speed ← 0.75
7: else if signalLevel ≤ -81 dBm then
8:     speed ← 0.1
9: else
10:    speed ← (signalLevel+81 dBm)/10 dBm + 0.1
11: end if
```

During the experiments described in Section 3.3.2, we collected the information on the Wi-Fi signal every 10 ms. We observed that the Wi-Fi signal level presents significant oscillations in the case of averaging it over a short time window (e.g., 50 ms). That is, two subsequent average measurements may report considerably different Wi-Fi signal levels. On the contrary, if we take a longer time window (e.g., 500 ms), the oscillations between subsequent average measurements are substantially reduced and the Wi-Fi signal varies in a smoother way. Based on this finding, the control algorithm will use the Wi-Fi signal level obtained by averaging it over a fixed time frame. Given the robot's speed bound between 0.1 m/s and 0.75 m/s, a time frame of 500 ms is considered to be a reasonable value. The computed Wi-Fi signal is then combined with the robot's odometry sensor data for adapting the robot's speed.

Algorithm 1 shows the pseudo-code of the control algorithm. The robot brain, in real-time, extracts the current signal level from the Wi-Fi MEC information service, stores it in a circular buffer, and computes the moving average of the Wi-Fi signal level. For each movement command, the adaptive speed and the adaptive distance are re-calculated. In Section 3.3.2 we observed that packet retransmissions and failures start increasing for signal values below -71 dBm, hitting their maximum between -79 and -81 dBm. Based on this observation, the control algorithm adapts the driving robot's speed to the maximum (0.75 m/s) for an average Wi-Fi signal level higher than -71 dBm. On the opposite end, the minimum robot speed (0.1 m/s) is selected for an average Wi-Fi signal level equal to or lower than -81 dBm. Between -71 dBm and -81 dBm, the control algorithm linearly adapts the robot's speed to the Wi-Fi signal level (e.g., 0.425 m/s with -76 dBm).

Experimental results

This section evaluates the adaptive speed control algorithm proposed in Section 3.3.3.2 and compares it with scenarios not making use of any context information. The following three scenarios are evaluated: *i)* the robot drives at minimum speed (0.1 m/s), *ii)* the robot drives at maximum speed (0.75 m/s), and *iii)* the robot uses our control algorithm to drive at adaptive speed. Following the experimental methodology described in Section 3.3.3.1, we performed 10 experiment runs for each scenario

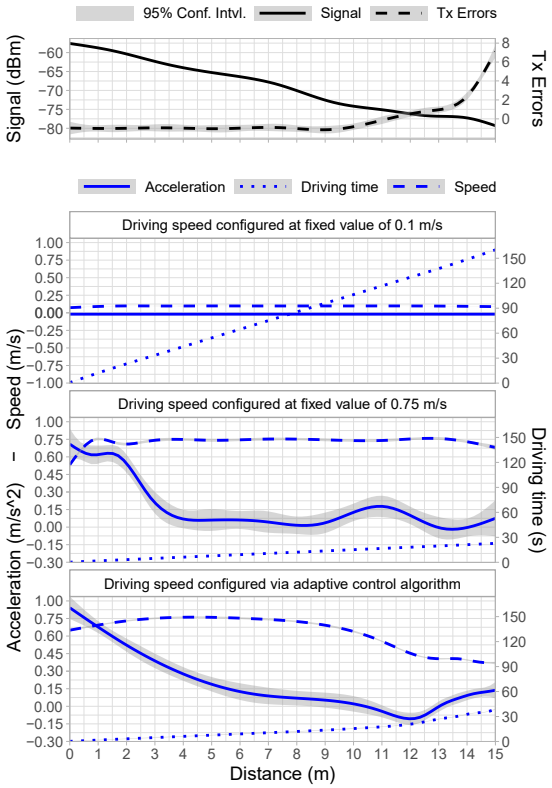


Figure 3.3. *Speed, acceleration, and driving time.*

(minimum speed, maximum speed, adaptive speed). In addition to the Wi-Fi information recorded in the robot brain, we record the odometry sensor data directly in the robot itself. This is because the data from the odometry sensors is not timestamped, and sending it over the Wi-Fi channel would not be suitable for measuring the speed and acceleration experienced by the robot (due to the risk of transmission failures over Wi-Fi).

The measured data is aggregated and analyzed to produce the results in Figure 3.3. The figure has four different graphs. On each graph, the x-axis is the distance traveled during the experiment, from the start (0 m) to the end (15 m). The first subgraph from the top presents the Wi-Fi signal level (y-axis on the left) and the transmission errors over the robot driving path (y-axis on the right). As it can be noticed, there is significant decay in the Wi-Fi signal quality in the last 5 meters of the driving path reflected by an exponential increase in the transmission errors. The remaining three graphs of Figure 3.3 present - for each evaluated scenario - the speed, the acceleration, and the driving time as measured via the odometry sensor data. Despite the acceleration and the speed having different units (m/s and m/s^2, respectively), they share the same y-axis on the left since they present the same range of values. The y-axis on the right represents the elapsed driving time since the start of the experiment run.

In the minimum speed experiment, the robot speed is set constant to 0.1 m/s from the start to the end. Similarly, the acceleration presents a constant value in the order of a few cm/s^2. Driving such

low-speed results in a smooth run that is not affected by the degradation of the Wi-Fi channel in the last segment of the path, since the slowness of the movement allows for more time to recover from possible transmission errors and retransmissions. As a drawback, the robot requires ~ 160 seconds to complete each experiment run. On contrary, the maximum speed experiment is the one requiring less time (~ 27 seconds). The impact of the decreasing Wi-Fi signal quality can be seen in the acceleration curve (notably in the last 5 meters of the path) where the acceleration fluctuates due to increased packet delay or delayed reaction, resulting in a stop-drive effect of frequent braking and spurring acceleration to full-speed. Effect of the stop-drive behavior, the driving direction is deviating from the straight driving path.

The bottom graph shows the motion behavior in the case of using the proposed adaptive speed control algorithm. A first observation is that the acceleration and deceleration in this case are smoother. At the start, the robot accelerates to full speed, since the received signal level is in the safe zone above -71 dBm. After crossing the -71 dBm threshold, the robot's speed is linearly reduced following the decrease of the Wi-Fi signal strength, reaching the end of the path and driving at minimum velocity. Regarding the driving time, the robot reaches the finish line ~ 10 seconds later than in the maximum speed experiment. Nonetheless, it is still ~ 120 seconds faster than the minimum speed experiment while performing a smooth ride. In concluding remarks, the results show that there is a trade-off between speed and smooth movement of the robot. By adapting the velocity of the robot with information on the quality of the Wi-Fi channel, the robot is able to move with maximum speed where the Wi-Fi signal channel is good and smoothly lowers the speed in the areas of weak Wi-Fi signal coverage, thus canceling any stop-drive effect.

Algorithm 2 Context-aware virtual Access Point offloading algorithm

```
1:  info ← GetCurrentWi-FiInfo()
2:  buffer ← buffer.removeOldestWi-FiInfo()
3:  buffer ← buffer.add(info)
4:  signalLevel ← buffer.average()
5:  if signalLevel ≤ -65 dBm then
6:      if signalLevel ≥ -69 dBm then
7:          currentAP ← info.GetAPinfo()
8:          newAP ← GetBestAvalaibleAP()
9:          deployNewAP(newAP)
10:         if newAP.ssid==avalaible then
11:             RobotHandover(newAP)
12:             StopOldAP(currentAP
13:         end if
14:     end if
15: end if
```

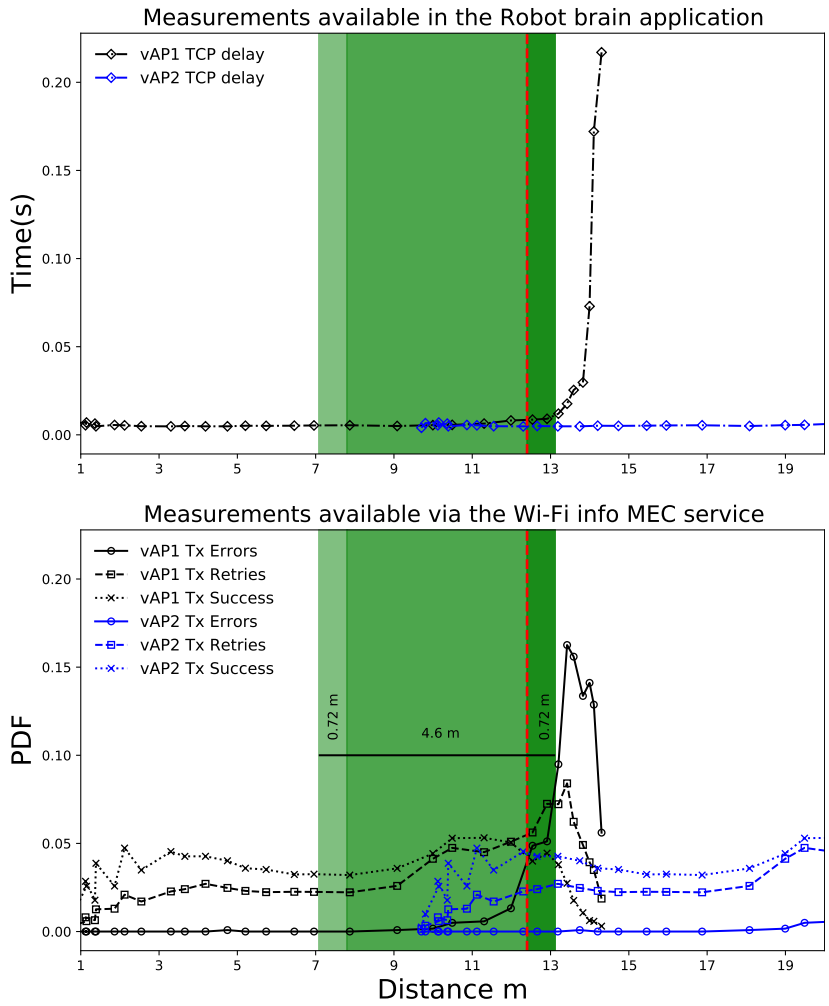


Figure 3.4. *Offloading of virtual Access Point.*

3.3.4. Virtual Access Point offloading

Algorithm design

Based on the conclusions drawn from Section 3.3.2, in this section, we present the design of a context-aware vAP offloading algorithm. The aim of this algorithm is to provide an uninterrupted robot delivery service while extending the network Wi-Fi coverage in indoor environments. Through this algorithm, we showcase the advantages of consuming context information for managing vAP. Nonetheless, we acknowledge that more complex and optimal algorithms than the one proposed in this section can be designed.

Algorithm 2 shows the pseudo-code of the vAP offloading algorithm. The orchestrator, in real-time, extracts the current signal level from the Wi-Fi Info service, stores it in a circular buffer, and computes the moving average of the Wi-Fi signal levels. In Section 3.3.2 we observed that our edge robotics system has significant degradation in performance for signal values below -80 dBm. In or-

der to avoid these values, the offloading algorithm must react proactively when deploying a new vAP. Based on this assumption, the offloading algorithm obtains information about the new AP and performs the deployment for an average signal level in the interval between -65 dBm and -69 dBm. Once the new vAP is up and running, the orchestrator triggers the robot handover from the currently associated to the newly instantiated vAP. In the end, the orchestrator stops the old vAP and releases the resources on the corresponding node. It is important to indicate that the vAP instances need to be configured to use IEEE 802.11r over DS in order to perform fast and secure handovers.

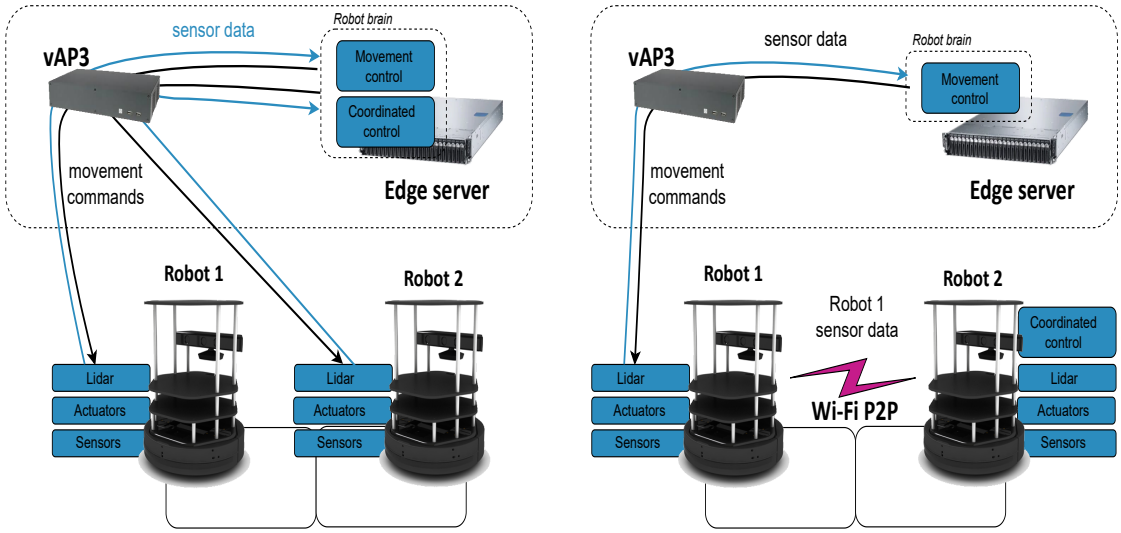
Experimental methodology

Testing the proposed context-aware algorithm was accomplished using the (A, C) segment shown in Figure 3.1. The robot was positioned at location A and connected to the vAP3. The Wi-Fi range of this AP is presented in Figure 3.1 with yellow. The robot, controlled by the robot brain, accelerates from position A to the target velocity of 0.2 m/s. During the drive, the orchestrator implements the vAP offloading algorithm and performs the vAP offloading from vAP3 to vAP5. After having traveled for 24 m, the robot stops at position C. In the orchestrator, we recorded the total vAP offloading time, which is the time elapsed from detecting that vAP offloading is needed until the old vAP3 is stopped. Moreover, to reduce the deployment time we have already made available a copy of the vAP container image in the nodes. By doing this, the orchestrator does not need to copy the vAP image over the network. In a separate laptop placed in the corridor, we recorded the IEEE 802.11 link layer overall downtime.

Experimental results

Figure 3.4 shows the available Wi-Fi context information from the departing vAP3 (in black) and the arriving vAP5 (in blue). The breakdown of the total offloading distance is reported in green. From left to right, Figure 3.4 depict the traversed robot distance during the Eclipse Fog05 Linux container vAP5 instantiation. Next, the distance traveled by the robot while the vAP5 is been provisioned (the moment when the SSID is available) and the overall handover distance are shown. Finally, the results for the robot traversed distance during the Eclipse Fog05 Linux container vAP3 terminations are presented.

Keeping in mind that the vAP function is deployed as a hostapd Linux container, the results show that the robot traveled distance is approximately 5.5 m during the vAP offloading procedure if driving at 0.2 m/s. Three factors contribute to this traveled distance: *i)* approximately 0.7 m of the robot traveled distance is for Eclipse Fog05 to deploy the vAP5 Linux container; *ii)* approximately 4.6 m of the robot traveled distance is for the hostpad Linux container to boot and the SSID to become available, and *iii)* approximately 0.7 m of the robot traveled distance for Eclipse Fog05 to stop the vAP3 Linux container. It is worth mentioning that during our experiments we recorded a value of 38 ms, as the time interval in which our robots are without Wi-Fi connectivity (link-layer handover downtime).



(a) *Centralized robotic control.* **(b)** *Network-assisted D2D robotic control.*

Figure 3.5. *Network-assisted D2D experiment scenarios.*

3.3.5. Network-assisted D2D robot coordination

One of the problems detected while performing coordination between the robots is the jitter introduced by the Wi-Fi link. Sometimes a coordinated order given to both robots, in separate messages addressed to each of the robots, results in different execution times, leading to the dis-coordination of the movement. To reduce this effect, our edge robotics system implements D2D communication to improve the reliability.

Experimental methodology

Network-assisted D2D communication testing was performed with two mobile robots where the floor layout is provided in Figure 3.1. The robots are connected to the vAP3 and positioned at location D, one behind the other with an approximate distance between them of 0.3 m. The first robot starts the experiment drive at a constant speed of 0.2 m/s. The second robot follows the first robot trying to keep a constant distance of 0.65 m. The robots moved from D to E, E to F, F to G, G to H, H to E, and E to I. The length of the hallway DFHI is 5 m and EG is 4m.

Figure 3.5 depicts two separate experiments scenarios. In both experiments, the robot brain implements a ROS-1 movement and coordinated control applications. The movement control application navigates Robot 1 through the known map. The coordinated control navigates Robot 2 by following the driving path of Robot 1 while maintaining a constant distance. In the centralized robotic control (see Figure 3.5a), the robot brain hosts both applications. The sensor data received from the robots (odometry, lidar) is used to calculate and execute driving instructions for the navigation of both robots. On the other side, in the network-assisted D2D robotic control, we have D2D communication between the two robots based on Wi-Fi Direct. The Coordinated control application is now placed

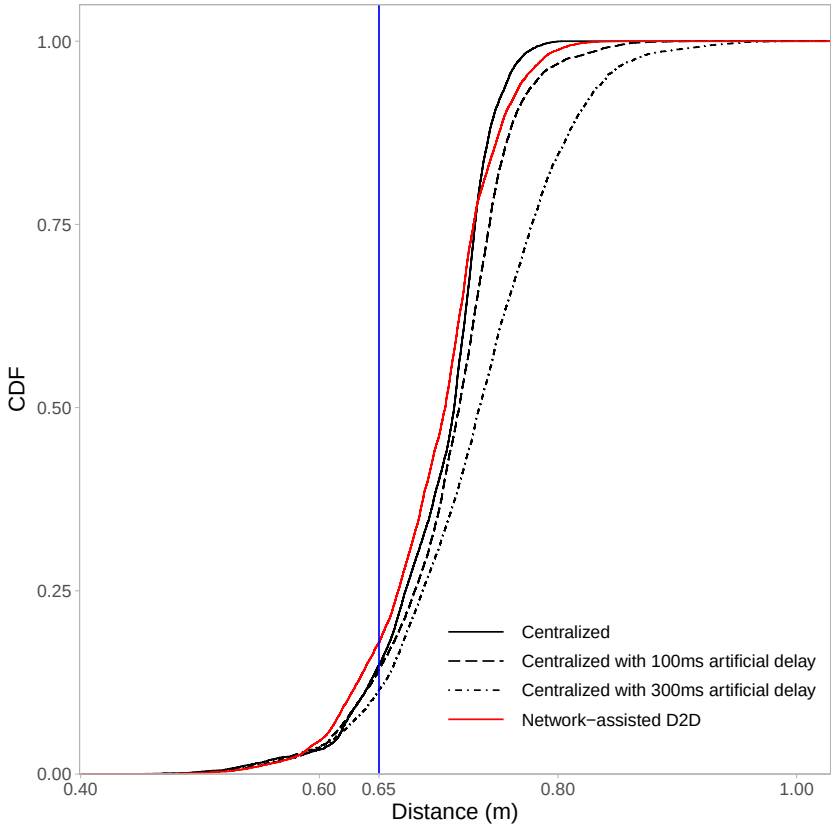


Figure 3.6. *eCDF of distance between the two robots.*

in Robot 2. Consequently, Robot 2 is now navigated by the Coordinated control app that consumes the sensor data of Robot 1 via the D2D communication channel. In order to emulate the effects of network interference, we introduced an artificial delay of 100ms and 300ms on the Wi-Fi link in the centralized robotic control. The selected values for the artificial delay were influenced by our robot sensor sampling frequency and control loop described in Section 2.3.1.

Experimental results

Figure 3.6 depicts the Euclidean distance between the robots. It is worth mentioning that, in our tests, the robots are making turns. Since we are measuring the straight line distance, this leads to shorter distances in our measurement set. The cumulative density functions (CDF) show that by using the D2D communication channel we can arrive closer to our target distance of 0.65 m during the experimental drive. This is because in the centralized robotic control experiments when we introduce artificial delay, robot location mismatch is triggered in the robot brain. The location mismatch then results in decreased precision when the robot brain tries to maintain a constant distance between the robots. Naturally, as shown in Figure 3.6, the Euclidean distance between the robots increases as we increase the artificial delay. However, Figure 3.6 also shows that the euclidean distance between the two robots in the non-artificially delayed centralized control is very close to the network-assisted

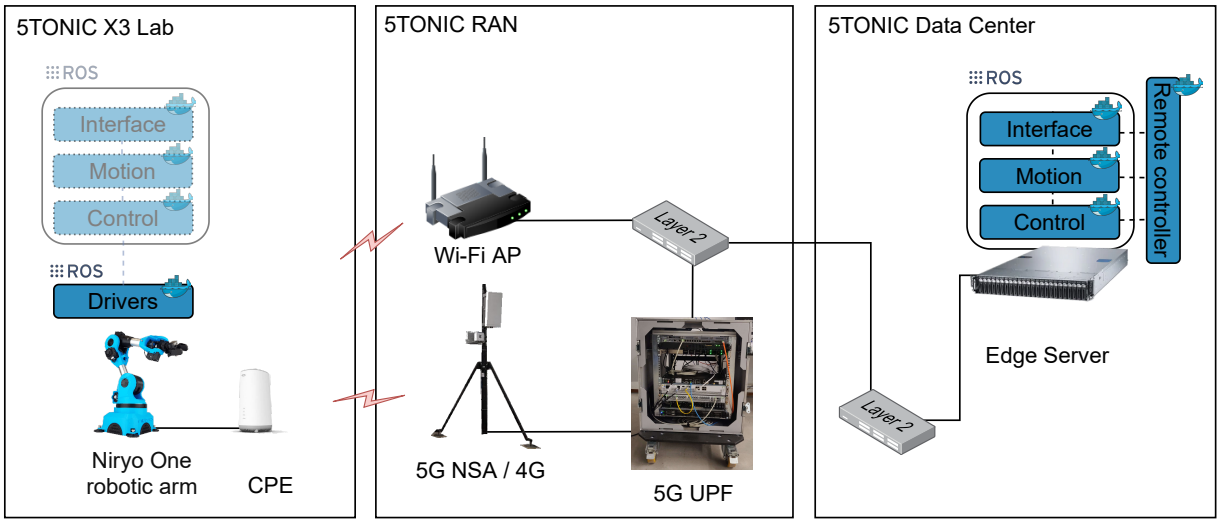


Figure 3.7. *The 5TONIC testbed used for experimentation.*

D2D. This is reasonable since we are using Wi-Fi as radio access technology for both experimental scenarios.

3.4. Resource consumption and impact of different RATs in remotely controlled robot manipulators

This section evaluates the potential benefits that MEC can bring to robot manipulators through the proposed edge robotics system in Section 2.3.1. First, in Section 3.4.1 we verify the proposed edge robotics system for robot manipulators by assessing the baseline performance of each component. Next, in Section 3.4.2 we look into the scale of the on-device resource consumption and potential offloading savings where several deployment configurations are considered for the edge robotics system for robot manipulators. Finally, in Section 3.4.3 the applicability of currently available RATs to support the proposed design is analyzed, together with their impact on the overall performance of the edge robotics system for robot manipulators.

In order to address the above mentioned objectives, we used the 5TONIC testbed presented in Figure 3.7. The testbed is composed of 6-axis Niryo One robotic arm that is connected to an edge server with 8GB RAM and 4vCPU@2.4GHz via Customer Premises Equipment (CPE). Therefore, the communication between the robot and the edge server comprises of a wireless link, from the CPE to the referent RAT, and a wired network, from the RAT to the edge server. Although Niryo One software stack is based on Robot Operating System version 1 (ROS-1) as provided by the robot manufacture[13], this work splits ROS-1 packages composing the Niryo One stack in several VNFs that are compatible with the developed edge robotic system for robot manipulators from Section 2.2.3.2. Moreover, VNFs are packaged either as virtual machines or containers.

[13]Widespread framework for developing and testing multi-vendor robotics software.

The different VNFs created from the provided Niryo One stack (i.e., drivers, control, motion planning, and interface VNFs) were implemented as a Docker container. The drivers VNF are deployed in the robotic arm, while the remaining VNFs of the robot stack are distributed across the robotic arm or the edge server envisioning different deployment configurations. The remote controller VNF is deployed in the edge server as a Docker container and implements ROS-based Python scripts for remote control of the robot manipulator.

Each VNF of the implemented prototype is following the interactions presented in Figure 2.3 and described in Section 2.2.3.2. Since ROS-1 is used as the robotic middleware, each VNF of this prototype is composed of one or more software components called ROS nodes. ROS provides a publish-subscribe messaging framework over which nodes exchange messages. By connecting to the ROS master, nodes register and locate each other. Once registered, nodes exchange messages via configurable topics in a peer-to-peer fashion over TCP. For more details about the hardware and software components of this testbed please see Section 2.3.2.

3.4.1. Edge robotics system for robot manipulators baseline performance

Table 3.1. *Baseline Performance of Interfacing with Different VNFs of the Robot Stack*

Interfacing Layer		**Closed-loop (ms)**	**Processing delay (ms)**
Interface	Robot	497.976 ± 5.023	109.723 ± 1.106
	Edge	409.535 ± 0.896	66.284 ± 0.145
Motion Planning	Robot	388.253 ± 2.365	365.178 ± 2.224
	Edge	343.251 ± 4.291	320.199 ± 4.002
Control	Robot	23.075 ± 0.005	3.075 ± 0.001
	Edge	23.052 ± 0.003	3.052 ± 0.001

Although this testbed is envisioning RATs for connectivity between the robot and the network, in order to obtain the most reliable baseline values for later comparison with wireless technologies (Section 3.4.3), the Niryo One robot is connected to the edge server using an Ethernet connection. Ethernet as technology is proven that can fulfill the remote control requirements. Two deployment configurations are considered: *(i)* the control, motion planning and interface VNFs deployed in the robotic arm; and *(ii)* the control, motion planning and Interface VNFs offloaded in the edge server. In both experiments, the remote controller VNF interacts with different VNFs from the robot stack, remotely controlling 1-axis of the robotic arm controlled for a period of 30 seconds and a movement offset of 0.010 rad.

Table 3.1 presents the closed-loop when interacting with different VNFs from the robot stack. The closed-loop can be seen as the remote control user experience and is defined as the time elapsed from the moment a remote command is issued from the remote controller until the respective robot

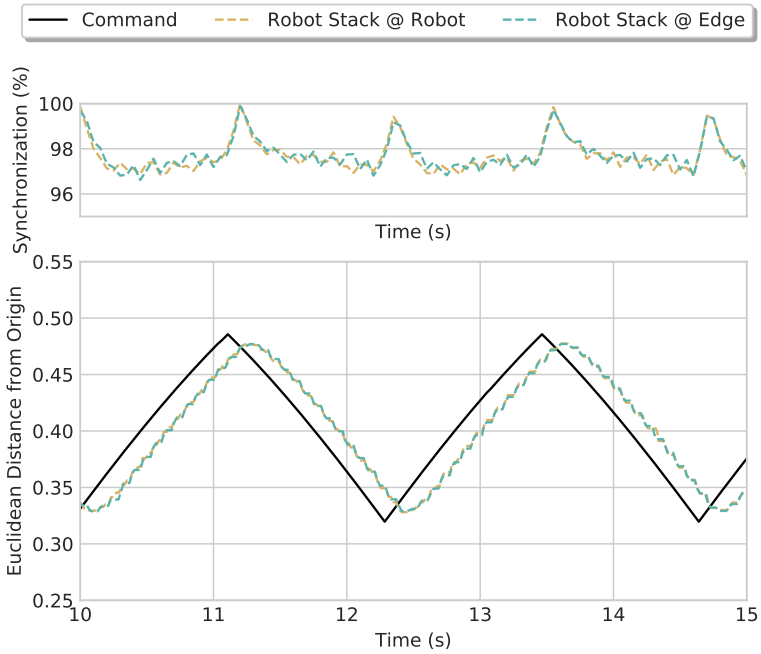


Figure 3.8. *Performance Comparison.*

joint states are received back. From the average values, the Interface and motion planning VNFs offer a slower closed-loop (498 ms and 388 ms, respectively) with respect to the control VNF (23 ms). This comes naturally as the Interface and motion planning VNFs represent the transition between the robot and remote control user. To do so, they implement a set of translations and validations that require additional processing for each command. In turn, the control VNF offers near real-time control by performing straight-forward operations. Regarding the processing delay, the control VNF results in the fastest processing because there are no command validations implemented, thus it may occur that a given command is enforced in the robotic arm even if it is not achievable. Moreover, it leaves the trajectory computation to the application developer, unlike the Interface and motion planning VNFs which offer optimal path computation. The optimal path computation and validation come at a price of processing delay, where the motion planning VNF needs 365 ms and the Interface VNF 112.7 ms to compute the trajectory for a simple command.

Figure 3.8 illustrates the position of the robotic arm and its remote controller over time (bottom) when interfacing with the control VNF (approximately 23ms closed-loop). Both positions are used to compute the synchronization accuracy over time (top). Results show that it is feasible to offload the robot stack to an edge server and still maintain similar synchronization levels between the robot manipulator and the remote controller (97.72±0.65% and 97.66±0.67%). This desynchronization is due to the non-alignment of the control and sampling frequencies, plus the time required to physically move all joints in the robotic arm. Moreover, Table 3.1 shows that there is a speed up on the closed-loop for the Interface and motion planning VNF when the robot stack is offloaded to the more powerful edge server. Such behavior is related to the faster computation capabilities of the edge server with respect to the limited computation capabilities of the robotic arm. At the control VNF such a de-

crease is not appreciated due to the low computational needs.

3.4.2. Resource consumption and savings potentials for robot manipulators

To look into the scale of the on-device resource consumption and potential offloading savings, several deployment configurations are considered in this section for the proposed edge robotics system for robot manipulators. This analysis aims to identify the main candidates for VNF offloading by assessing the potential resource savings in terms of CPU, MEM, network usage, and energy consumption. It is worth mentioning that the experimental evaluation in this section is performed over a 10Gb Ethernet connection. The main reason is that when investigating the potential on-device resource savings of our system, the bottleneck is not on the communication infrastructure.

Experimental methodology

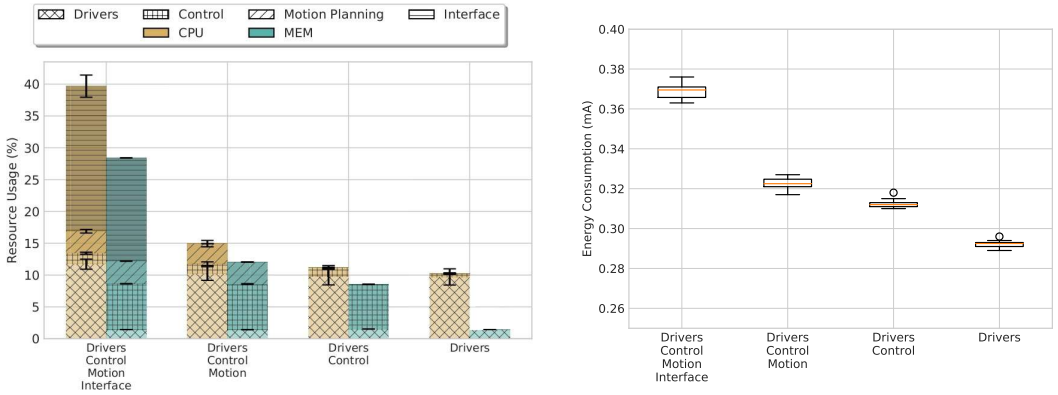
Each experimental run of this section consists of the remote controller VNF controlling 1-axis of the robotic arm for 30 seconds and a movement offset of 0.010 rad. The Interface VNF is used as the interfacing VNF to force the whole robot stack to process every command. Experiments consider the following on-device configurations: *i)* drivers VNF; *ii)* drivers and control VNFs; *iii)* drivers, control and motion planning VNFs; and *iv)* full robot stack. The described experiments are repeated 10 times for each deployment, being presented the average values and their standard deviation.

Figure 3.9 presents the average resources usage in terms of CPU, MEM, network and energy consumption for all the deployment configurations while processing navigation commands. The obtained data for the CPU, MEM and network consumption is measured in the robotic arm with a python script using the cross-platform library psutil[14]. The energy consumption data is measured with an energy consumption meter installed in the power supply of the robotic arm.

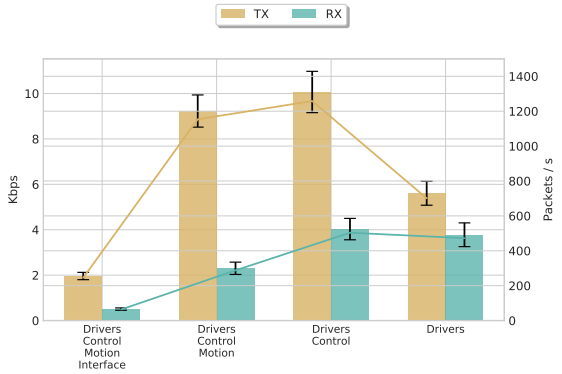
Experimental results

It can be seen from Figure 3.9a that the CPU consumption of the whole robot stack is approximately 40% of the total CPU consumption on the robotic arm. Excluding the drivers VNF that must be deployed in the robotic arm, the CPU consumption can be potentially reduced by around 30% by offloading all the remaining VNFs. In particular, around 24% by offloading the Interface VNF, around 5% by offloading the motion planning VNF, and around 1% by offloading the control VNF. The Interface VNF has higher computational requirements, needed to perform the command translation, validation, and real-time synchronization between the remote controller and the ROS systems. For what concerns the control and motion planning VNFs, the CPU consumption of the control VNF is mainly due to the straightforward control-loop mechanism that interpolates and forwards position

[14] https://pypi.org/project/psutil [Accessed: Jul. 14, 2022]



(a) *CPU and MEM.* **(b)** *Energy Consumption.*



(c) *TX and RX.*

Figure 3.9. *Resource Consumption on Robotic Arm.*

commands, while in the motion planning VNF this is a result of a simple path planning in our experiment runs.

Regarding MEM usage, the total MEM consumption on the robotic arm is approximately 28% as shown in Figure 3.9a. This reflects 2% used by the drivers VNF, meaning that around 26% MEM savings can be achieved by offloading the remaining VNFs. Similarly, the Interface VNF offers highest potential savings of around 15% of the total MEM usage, while the motion planning and control VNFs around 4% and 7%, respectively. The 15% of MEM used by the Interface VNF is partially due to the robot action server that handles remote controller concurrent requests, checks if the command can be processed, validates parameters and calls required controllers (e.g., motion planning VNF or control VNF). Additionally, for the motion planning and control VNFs, the MEM usage is significantly smaller since the robotic arm receives simplified navigation commands from the Interface VNF.

The energy consumption (Figure 3.9b) follows the CPU and MEM usage pattern from Figure 3.9a, where most energy is consumed when the robot stack is deployed in the robotic arm. As VNFs are offloaded to the edge, the energy consumption in the robotic arm decreases, proving that energy savings in devices can be achieved by offloading components in the edge. By moving the energy contribution mostly to the data centers at the edge, different energy-aware optimization models are easier to im-

plement [111].

Figure 3.9c shows that a deployment configuration where the whole robot stack is in the robotic arm introduces the smallest overhead in terms of network utilization. This comes naturally because in this deployment configuration the robotic arm transmits (TX) only the joint states to the remote controller and receives (RX) high-level commands from the remote controller processed by the Interface VNF at a lower frequency. Results also show that from all deployment configurations, deploying only the robot drivers VNF in the robotic arm introduces less overhead into the communication link. However, there is still an increment in the network utilization compared to a self-contained deployment configuration in the robotic arm. Such behavior is mainly due to the offloaded control VNF. While in upstream the robot drivers VNF need to transmit the joint states to the control VNF, in downstream this increment is due to the higher frequency of commands sent by the control VNF to the robotic arm. In the case when the control VNF is deployed in the robotic arm a significant increase of TX is witnessed. This is mainly because the control VNF needs to continuously interact with the Interface VNF in order to enable robot manipulation.

The resource consumption evaluated in the experiments above clearly showed that the Interface and motion planning VNFs are the most likely targets for potential resource savings. However, when considering the network overhead that the control VNF introduces on the communication link, its offloading should be also considered.

3.4.3. Impact of Radio Access Technologies in remotely controlled robot manipulators

A remote control system places stringent networking requirements that must be fulfilled by the underlying RAT. In this section, the applicability of currently available RATs to support the proposed design is analyzed, together with their impact on the overall performance of the remotely controlled robot manipulator.

Experimental methodology

To do so, the Niryo One robotic arm is connected to the remote controller using Wi-Fi, 4G and 5G. For more details about the Wi-Fi, 4G and 5G implementations please see Section 2.3.2. The benchmark of each RAT available at 5TONIC testbed is provided in Table 3.2. It is worth mentioning that, when the experiments were conducted, only 5G NSA was available as a 5G solution in 5TONIC and that is the reason why in the 5G related experiments only 5G NSA results are provided.

The first set of experiments assesses how the designed system operates over interference-prone Wi-Fi link, causing increased packet loss and jitter to occur. Throughout the duration of the experiment, the remote controller interacted with the control VNF and 1-axis of the robotic arm is remotely controlled for a period of 30 seconds with a control period of 20 ms and a movement offset of 0.010 rad. The remote controller records the instructed and executed commands with which the traveled

Table 3.2. *Wi-Fi, 4G and 5G NSA Benchmark at 5TONIC*

Metric	**Wi-Fi [2.4Ghz]**	**4G**	**5G NSA**[15]
E2E Latency	1.77±0.68 ms	23.88±5.84 ms	6.56±1.04 ms
Packet-loss	0.11±0.35%	0%	0%
Data Rate (Uplink)	37.76±7.71 Mbps	44±0.20Mbps	96±1.81Mbps
Data Rate (Downlink)	49.39±1.80 Mbps	72±1.04 Mbps	600±13.50 Mbps
Jitter	1.10±0.73 ms	2.32±0.35 ms	0.46±0.18 ms

distance is computed. NetEm[16] is used to introduce symmetric artificial jitter and packet loss in uplink and downlink.

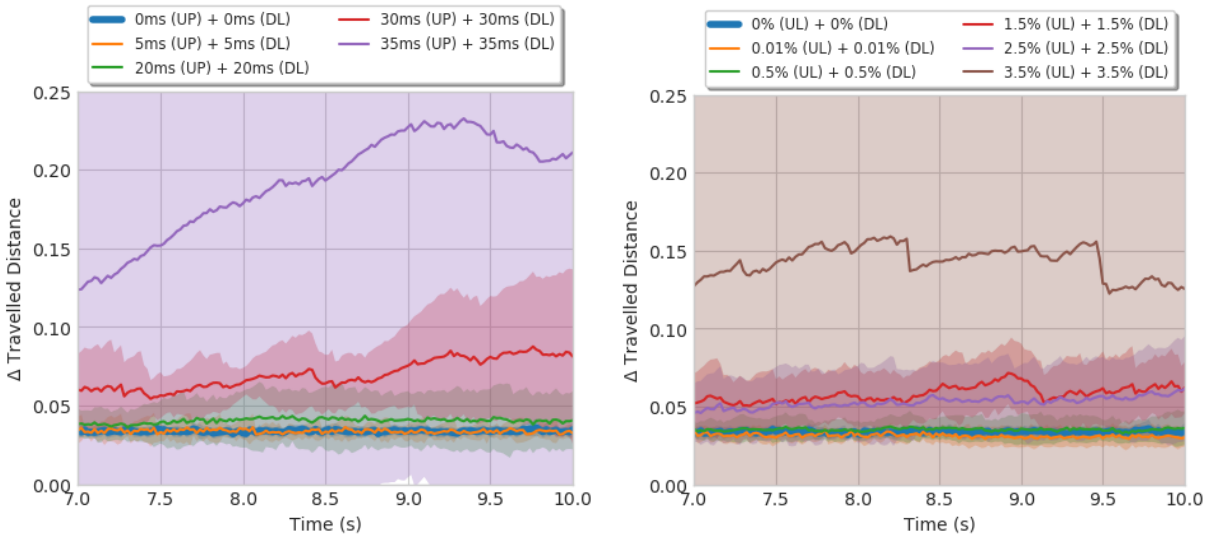
The second set of experiments aims to assess the minimum control period that can be supported in our system considering 4G and 5G generations of the cellular technologies, and Wi-Fi. Throughout the duration of the experiments, the remote controller interacts with the control VNF to remotely control 3-axis of the robotic arm for a period of 30 seconds with a movement offset of 0.010 rad. In the remote controller, the synchronization between the robotic arm and the remote controller is measured over time.

The third set of experiments aims to evaluate the synchronization between the robot manipulator and the remote controller, by measuring their distance offset over time. The Niryo One robotic manipulator is controlled by the remote controller that is continuously sending instructions (in a loop) to the robot and receives its pose at each instant. Such experiment is performed for a different combination of the following configurations: (*i*) 5G vs 4G (Net); (*ii*) remote controller located on the edge vs cloud (RC); and (*iii*) computation stack located on the robot vs offloaded to the edge (Comput). In doing so, results were measured for five different configurations. It is worth mentioning that in this set of experiments we emulated a more distant cloud by introducing an artificial delay of 7ms between the edge data center and the robot.

3.4.4. Experimental results

Figure 3.10 compares the average traveled distance for different values of jitter and packet loss on the Wi-Fi link. Results show that a 5 ms jitter in uplink and downlink does not have any impact on the remote controller operation. This is because the 20 ms control and update period of the remote controller are not violated by such an amount of jitter, allowing remote commands to be executed and the remote controller to be updated in real-time. With a 20 ms of uplink and downlink jitter, a slight increase in the deviation between the instructed and executed commands is noticeable, including the

[16] https://www.linux.org/docs/man8/tc-netem.html [Accessed: Jul. 14, 2022]



(a) *Jitter.* **(b)** *Packet Loss.*

Figure 3.10. *Impact of Network Conditions on Remote Control Synchronization.*

standard deviation of the measured values. This comes naturally since the control frequency constraint is not satisfied at all times, causing some commands to be discarded by the robotic arm (since their lifespan is expired upon reception). Moreover, the joint state updates to the remote controller are delayed, resulting in a desynchronized remote controller. As shown on Figure 3.10a, the deviation continues to increase with the jitter increase on the Wi-Fi link. When the jitter is approximately double of the control period (e.g., 35 ms), the robotic arm starts to move erratically, including unpredictable jumping moves, which makes the whole environment unsafe for operation.

In turn, in Figure 3.10b, a similar analysis is provided by taking into account the packet loss. Although ROS uses TCP as the underlying transport protocol, a lost packet is never retransmitted within the 20 ms control period of the robotic arm (i.e., TCP_RTO_MIN is configured with 200 ms). With a packet loss of 0.01% in uplink and downlink, no visible impact is witnessed on the operation of the remote controller. The reasons are two-fold: *(i)* there is a very small number of packets lost, and *(ii)* it is able to recover when the lost packet is retransmitted. When the packet loss is increased to 0.5% in both directions, a slight deviation of the instructed and executed commands is witnessed. However, these values are still manageable by the remote controller, without a noticeable impact on its operation. However, this is not the case when the packet loss increases up to 1.5% and 3.5%. In the latter case, the operation of the robotic arm through the remote controller is degraded significantly, making it unusable.

Figure 3.11 compares the achieved synchronization by the remote controller when implementing different control periods. Considering an average synchronization of 95% as the threshold below which the impact on the robotic arm operation is noticeable by a human operator, it is not possible to operate the robotic arm with the minimum control period (i.e., 20 ms) using 4G. In fact, the operation of the robotic arm through the remote controller is only usable with a control period of 30 ms and 40

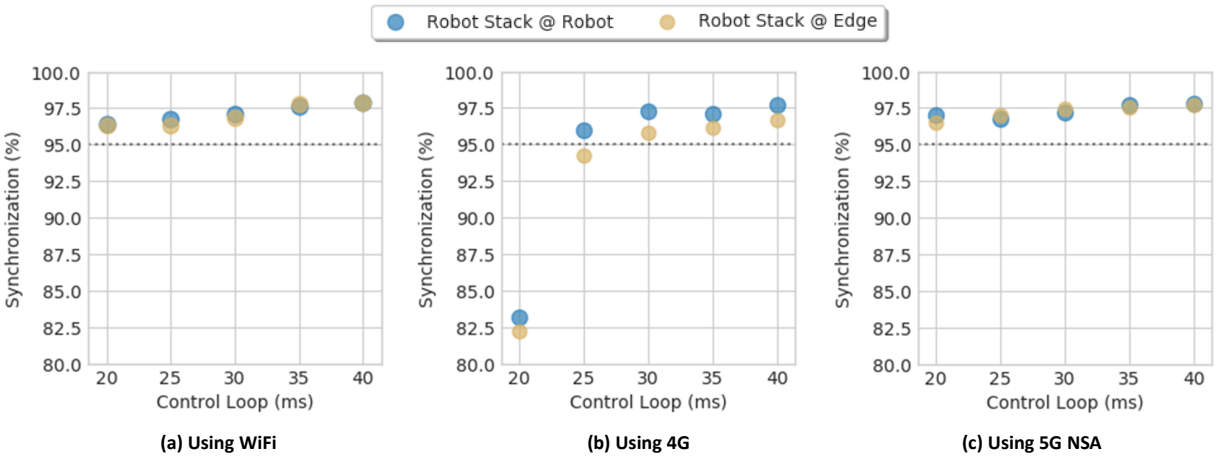


Figure 3.11. *Impact of Control Period Variation.*

ms, when considering the whole ROS stack on the robotic arm or offloaded into the edge, respectively. In turn, with 5G and an interference-free Wi-Fi channel, the operation of the robotic arm is achieved using the minimum control period supported. The reason for such results is the lower latency witnessed in the 5G NSA system (6.56 ms) and Wi-Fi (1.77 ms) when compared to the 4G system (23.88 ms). Still, for all experiments, whenever the distance increases (i.e., the latency between the remote controller and the robot arm), the control period must be reconfigured to a higher value than the E2E latency. By increasing the control period, commands are issued at a lower frequency and, therefore, the robotic arm moves slower, allowing higher synchronization to be achieved over time. In addition, when the control and update periods are around or below the E2E latency, the control loop is continuously missing the desired rate and the control and/or drivers VNFs may end up in an error state. This error state will break the remote control and the overall system will become unreliable and unstable.

Figure 3.12 shows the results for the five configurations considered (expressed as a 3-tuple *Net-RC-Comput*), namely the position of both the robotic arm and the remote controller time (bottom) and the synchronization accuracy (top). Considering as the baseline configuration (i.e., state-of-the-art solution) having the full robot stack in the robot, results show that the remote controller is able to achieve synchronization of 97.72±0.65% when leveraging on 5G connectivity (i.e., 5G-edge-robot). An error in the synchronization is witnessed because the Niryo One robot has a control loop of 20 ms, meaning that any update on the pose is only sent to the remote controller on the next control loop after receiving the instruction. When relying on 4G connectivity (i.e., 4G-edge-robot), the synchronization decreases down to 95.40±2.88%, making the robot move erratically. Since the latency is bigger than the robot's control loop, instructions whose lifespan is expired are discarded by the robot. In fact, when using 4G, the robot stack cannot be offloaded to the edge due to the latency witnessed in the link. When 5G is considered, it is possible to offload the robot stack to the edge (i.e., 5G-edge-edge), while maintaining the same performance levels between the physical and the digital replica (i.e., synchronization of 97.66±0.67%). Finally, when considering the remote controller in the cloud and the robot stack in the robot (i.e., 5G-cloud-robot and 4G-cloud-robot), the high latency caused

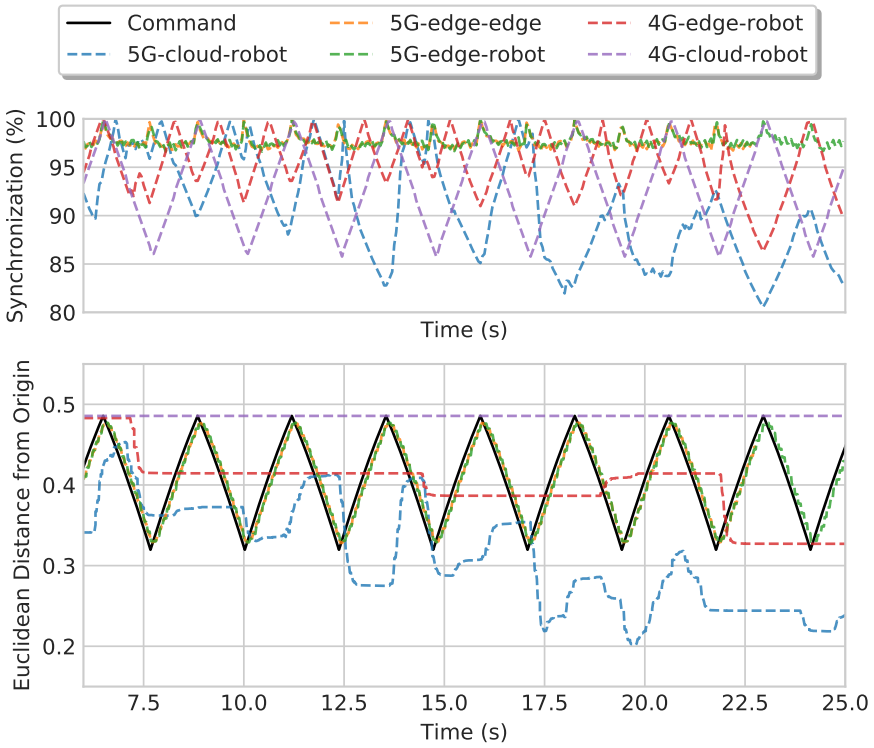


Figure 3.12. *Synchronization between the robot manipulator and the remote controller.*

unpredictable movements to be executed by the robot (when using 5G) or to not move at all (when using 4G), as expired instructions are discarded by the robot.

3.5. Discussion and future directions

Although the edge and fog ecosystem can bring advantages to the real-time robotics system by integrating IT and OT technologies, there are inherent gaps and challenges that are yet to be adequately addressed. This section presents the main issues that we have identified and recommends some future trends towards IT and OT convergence.

3.5.1. Operational technologies

Offloading the computation tasks from the robots to the edge of the network requires a distributed, scalable, and fast software framework. One of the open issues of ROS-1 is that the centralized design uses best-effort technologies (e.g., HTTP, IP) and depends highly on excellent network connectivity. Therefore, it is not suitable for multi-robot real-time embedded systems. To address this issue, the ROS community recently started ROS-2 which builds its communication system around Data Distributed Service (DDS) to achieve system decentralization. DDS is a widely deployed OT communi-

cation protocol for industrial and real-time critical systems. However, DDS as a middleware protocol tends to have scalability and reliability issues when implemented over *i)* wireless networks; and *ii)* non-local area networks.

In order to deal with such issues, initial efforts have been made to bring Zenoh in ROS-2[17] to solve the reliability issues of DDS over error-prone and large-scale networks (e.g., WLAN and Internet) whilst keeping a significant level of time and space efficiency.[18] Therefore, in the coming years, we can expect ROS-2 to gradually integrate Zenoh as a middleware so as to achieve: *i)* more hard real-time and reliable robotics applications; and *ii)* a higher degree of decentralization and distribution of robotics applications in more heterogeneous network scenarios.

3.5.2. Wireless local area network

WLAN is a part of the IEEE 802 set of LAN protocols, and, as a non-deterministic protocol, is unsuitable for hard real-time applications. The media access control protocol with its backoff algorithm prevents the network from supporting hard real-time communication due to its random delays and potential transmission failures. To address this issue, Time-Sensitive Networking (TSN) and IEEE 802.11 groups are working together towards equipping IEEE 802.11 wireless networks with real-time capabilities [112] [113]. Hence, we recommend that future researchers consider this extension of TSN to ensure the performance of the robotics system over WLAN.

3.5.3. Information Technologies

This work makes use of the virtualized edge to run access points, which is one example of an application requiring hardware support in the edge. Another application with such needs is to provide AI capabilities on-demand (GPU or hardware accelerators needed). Hypervisors (e.g., Hyper-V, KVM) and container systems support such virtual replication of specific hardware by enabling pass-through or user namespaces. However, in our case, using a hardware-specific (e.g., TPU, GPU, Wi-Fi/Bluetooth adapters) virtualization involves a lot of installation and integration which leads to high configuration complexity. Recently, Wireless Network Virtualization (WNV) has been proposed as an exciting innovation that enables physical wireless network infrastructure resources to be abstracted into virtual resources and shared by multiple parties [114]. As a future research direction, this technology may be used to introduce the RAT-as-a-Service (RaaS). Customers can run and manage access point capabilities in the edge without the complexity of developing and testing RAT applications.

[17] https://github.com/osrf/zenoh_evaluation [Accessed: 20 April 2022]

[18] The minimum wire overhead introduced by Zenoh is 4 bytes. The tiniest Zenoh implementation can run in 300 bytes of RAM on an 8-bit microcontroller. Information retrieved from the Zenoh project website: http://zenoh.io/ [Accessed: 20 April 2022]

3.5.4. Orchestration and control

Today's mainstream orchestration solutions are designed with an IT environment in mind, in which all resources are similar and reside in data centers, and resources are interconnected, leveraging the highly-reliable and high-throughput network technologies. In contrast edge and fog robotic environment is made of heterogeneous, volatile, and mobile resources interconnected by OT protocols and unreliable wireless technologies. Such differences require research work to improve the existing orchestration solutions. There is a gap in the area of describing resources and their inter-connectivity in a heterogeneous environment. This can be addressed by designing a unified information model that can describe both IT and OT infrastructures. There is a need to work on infrastructure monitoring. Existing solutions contain stream-based monitoring, thus requiring high-bandwidth networks to update the orchestrators. There is a need to develop new monitoring paradigms that can work over unreliable and low-bandwidth networks. Finally, volatility and mobility introduce a different life cycle for the resources. Research work is required to track the state changes for such resources to improve the reliability of applications over such unstable infrastructure.

3.5.5. Cross-disciplinary teaming

Moving robotics algorithms in the edge of the network made us realize that experience from different engineering fields is needed in order to cope with the complexity of developing and integrating a network-assisted robot. Tools and methods that are explicitly made for the fields of robotics, mechanical engineering, software and network development, and system engineering are exploited in a single system that needs to perform in an optimal way. This is a major obstacle in the applicability of networked robots in Industry 4.0. Frameworks for robotics systems have been introduced in the past (e.g., Rapyuta, RoboEarth, RoboBrain, etc.). Their focus is mostly to ease and automate the software development. However, when it comes to testing the end-to-end robotics system, we need to follow the stepwise refinement approach that slows down the development process mainly because it is tightly coupled with the physical infrastructure (e.g., access points, robots). To address this issue, simulation environments have been geared specifically towards robotics (Bullet, OpenRAVE, Gazebo, CoppeliaSIM) to improve the sim-to-real domain adaptation. Therefore, a future research direction is to develop models which can consider the edge robotics production environments. The reason is that an end-to-end edge robotics system is highly heterogeneous and has various types of resources in the physical infrastructure. A digital twin solution that supports such heterogeneous infrastructure can be involved in the development process to provide faster realization of an end-to-end edge robotics system.

3.5.6. Multi-RAT support for robotic systems

In an industrial environment, latency and transmission reliability can be improved by introducing multi-RAT connectivity. A robot starts by selecting the RAT, among the available ones on the factory

floor, that satisfies its connectivity requirements. Simultaneously, each robot can also associate itself with another RAT to create backup communication links, thus improving the transmission reliability. However, the support of multiple flows in a transparent way for applications is a critical challenge when designing a robotic system. One of the questions that arise is at which layer of the protocol stack to implement such a solution. Although recently standardized, Multipath TCP (MPTCP) [115] is a Transport layer protocol that can use one or more RAT interfaces to simultaneously achieve higher throughput and reliability. As such, MPTCP is gaining increasing popularity among vendors, telecom providers, and startups that are finding applicability in robotic solutions. Other solutions at lower layers, such as Packet Replication and Elimination Functions (PREF) [116] at the network layer or Frame Replication and Elimination for Reliability (FRER) [117] at Data Link Layer (only applicable to IEEE 802 link technologies), can also be leveraged to improve reliability.

3.6. Conclusions

With the aim of studying the potential enhancements for robotic systems by using the MEC, this chapter shows how the radio network information that is available at the edge of the network can be used for implementing advanced features in robotics systems. Moreover, this chapter gives insights into the potential savings that robot manipulators can obtain by offloading robotic functions to the edge. In this context, this chapter addresses the **C2**, **C3** and **C4** defined in the scope of this thesis by providing:

- Control algorithm which consumes context information about the Wi-Fi signal and adapts the robot's speed and achieves smoother driving.
- Virtual access point offloading algorithm which uses the context information about the Wi-Fi channel to extend the driving range of the mobile robots without the interruption of service.
- Network-assisted D2D communication between a fleet of mobile robots that can improve the coordination.
- Faster computation times of Interface and Motion planning VNFs for approximately 88 ms and 45 ms respectively, when offloading them to an edge server.
- 16% of CPU and 34% of MEM savings that can be achieved by virtualizing and offloading robot manipulator virtual functions in the MEC.
- 5G connectivity that enables remote control of 20 ms, appearing as the most promising radio access technology to support the main requirements for remote control.
- The combination of 5G connectivity with edge computing achieves a remote control synchronization of 97.66±0.67% while allowing the offloading of the robot stack in the edge.

CHAPTER 4

Orchestration of Robotic services

There is a rising interest in the networking community for providing support to different robotic use cases (e.g., security/surveillance, cleaning, delivery of goods, collecting products) in industrial deployments. The goal is always to meet the KPIs that mission-critical robotic applications require as it is in the case of remotely controlled mobile robots where the goal is to provide uninterrupted connectivity over the radio access network. However, ensuring these KPIs over wireless networks is very challenging and many times it requires the correct selection of the radio points of attachment and the use of resources that may span from the cloud to the edge of the network infrastructure (including MEC), to the fog (e.g., connected robots).

Due to this, and with the aim of addressing the **C5** and **C6** defined within the scope of this thesis, this chapter explores the feasibility of utilization of orchestration solutions for mobile robotic systems. To that end, Section 4.1 introduces the main concepts that are used in this chapter and present the main contributions. Section 4.2 provides the needed background and related work for: *i)* OKpi - an orchestration algorithm that is designed to meet the required KPIs accounting for the network and computing resource heterogeneity that is present in the cloud-to-thing continuum, and *ii)* the concept of DLT-based federation in edge robotic systems. Consequently, Section 4.3 and Section 4.4 present the implementation, integration and experimental evaluation of OKpi and DLT-based federation in mobile robotic systems respectively. Finally, Section 4.5 concludes the chapter by summarizing the most relevant contributions and elaborates on how this chapter addressed the above mentioned challenges.

4.1. Introduction

In recent years, edge robotics emerged as a consequence of the rapid development of edge computing to address the network performance (e.g., high latency, unpredictable jitter) related challenges that cloud-based robotic applications experience. By placing computing and storage resources near the edge, robotic systems can execute applications closer to the robots resulting in more predictable communication and overall better system performance. For the market, the edge robotics services are an opportunity for mobile robots to be employed in accomplishing a range of manual tasks (e.g., security/surveillance, cleaning, delivery of goods, collecting fruits, e-health emergency response, sports video coverage, etc.). As presented in Section 2.2, the linchpin of edge robotics is the network slicing where vertical industries, such as automotive, e-health, and smart factories, can define the robotic services through a set of Virtual Network Functions (VNFs), connected according to the so-called VNF graph, and a set of performance requirements, i.e., the KPIs. Motivated by the advent of these performance requirements, different standardization bodies and alliances, such as ETSI, 3GPP, 5G Alliance for Connected Industries and Automation (5G ACIA), and Next Generation Mobile Networks Alliance (NGMN), defined new use cases with distinct connectivity requirements that connect people, robots, processes and systems [10][11][12][13]. Table 4.1 summarizes the key connectivity KPIs of different industrial use cases mapped into the robotics categorization. For example, monitoring robotic systems can be applied to use cases such as industrial condition monitoring and process automation based on sensors. Simulation robotic systems (Digital Twins) can be applied to factory remote maintenance, whereas the operational robotic systems include remote operation, motion control, safe control, and closed-loop control use cases.

Table 4.1. *Connectivity requirements of Networked Robots*

Robotic service	**Latency**	**Data Rate**	**Reliability**	**Scalability**
Monitoring	50-100 ms	0.1-0.5 Mbps	99.9%	100-1000 nodes
Simulation	20-50 ms	1-1000 Mbps	99.99%	1-100 nodes
Operation	0.5-20 ms	1-100 Mbps	99.9999%	1-50 nodes

To fulfill these KPIs, the mobile robotic service implies the selection of the radio points of attachment and the use of resources that may span from the cloud to the edge of the network infrastructure (including MEC), to the fog (e.g., connected robots). Also, it is critical that robot VNFs are placed and connected, so as to *i)* meet the target glsshortkpi values, *ii)* make efficient use of the different available resources, thus avoiding resource shortage, and *iii)* minimize the service deployment cost, one of the main concerns for both mobile operators and vertical industries [21, 118–120]. This problem has been addressed in the literature by proposing OKpi [121], an efficient and effective solution strategy that can jointly make VNF placement and data routing decisions (including the selection of the ra-

dio points of attachment) while accounting for all resources that may be available from the fog to the cloud. Applying OKpi to edge robotic systems has the potential to fulfill the strict KPIs requirements imposed by mission-critical robotic applications while guaranteeing the required QoE.

In addition to orchestration algorithms, the adaptation of NFV and MEC in the 5G networks introduced the concept of federation. The federation as a 5G networks concept, extends the orchestration of resources and robotic services across multiple administrative domains. Often enough, an edge robotics service requires fast and short-lasting expansion of the service footprint over multiple administrative domains (e.g., delivery of goods for a big day-lasting event, emergency response for large area, video streaming of cycling events, etc.). For example, virtualized access networks enable the robotic service providers to request on-demand deployment of the virtualized access point, in an external administrative domain at a specific location through the federation process. With the introduction of the fog concept, where volatile and low-power consumption devices are used as an access network, federation extends the eco-system heterogeneity and variance in the access network coverage. Multiple administrative domains can simultaneously deploy virtualized wireless networks over a range of hardware devices thanks to fog computing and edge computing supported by NFV.

As a consequence, a higher number of involved administrative domains, eligible to provide on-demand federation of services and resources, increase the risk of security threats, maintaining SLAs, privacy violations, etc. The Distributed Ledger Technology (DLT) is a potential solution to counter the negative byproducts of the federation process. The Blockchain as a DLT, by default, provides trust, security, and cryptography to participants. Leveraging the DLTs, the administrative domain can discover, negotiate and federate services on-the-fly. Through the application of DLT-based robotic service federation, an edge robotics service would not be limited to a single AD and it would be able to extend the desired service footprint at any time, anywhere.

Following the above mentioned opportunities for robotic systems, this chapter studies the practical feasibility of utilizing innovative orchestration concepts in mobile edge robotic scenarios. For this purpose, this chapter builds on top of the edge-based robotic system for mobile robots presented in Chapter 2.2 where we deploy OKpi and trusty & untrusty Etherium blockchains.

With the objective to address the **C5** and **C6** that are in the scope of this thesis, the main contributions of this chapter are:

- We are showcasing the feasibility of applying orchestration solutions in edge robotics systems.

- We implement, integrate, and experimentally evaluate OKpi, an innovative orchestration algorithm that performs VNF placement and data routing decisions to meet the KPIs of remotely controlled mobile robots.

- We implement, integrate, and experimentally evaluate DLT-based federation for mobile robots that extends the robot wireless connectivity in an external administrative domain.

4.2. Background and Motivation

In this section we present the needed background and motivation for applying OKpi and DLT-based federation for mobile robotic systems. The DLT-based federation concept is explained in Section 4.2.1 where we elaborate on how DLT can be applied for private, secure, and trusty edge federation. Section 4.2.2 provides the summary of OKPi and elaborates on why OKpi (or similar orchestration algorithms) have the potential to solve existing challenges in robotic services.

4.2.1. DLT-based federation

Depending on how the service federation procedures (described in Section 1.1.6) are realized, the sequential completion of the whole federation process can take more than a minute or even an hour. In a dynamic and heterogeneous environment, where the underlying infrastructure of each domain is continuously modified, the state can change in the order of seconds. In recent years, the DLT or blockchain gained significant applicability in the networking world. Originated as a driving mechanism for Bitcoin [122], blockchain provides a distributed and secure ledger that records every transaction between anonymous users. The blockchain itself contains distributed time-stamped blocks filled with transactions that can contain any data. Each block points to the prior block, creating a chain of blocks in history until the genesis block (block 0). The blocks are generated by nodes (e.g., any computing device) interconnected in a peer-to-peer blockchain network. The goal is to maintain a single block creation at a time for the whole blockchain network. Thus the nodes run a consensus mechanism to validate the single block creation process. Various consensus mechanisms exist from more simple (Byzantine Fault Tolerance - BFT, Proof-of-Authority - PoA), to more computational expensive such as Proof-of-Work (PoW) (used in Bitcoin), or incentive-based such as Proof-of-Stake (PoS) [123]. The consensus mechanism has the role of maintaining the blockchain distributed, secure, trust and privacy features, without the need for a 3rd-party centralized entity.

There are two types of blockchain networks: permissionless and permissioned. Permissionless blockchain networks are open and referred to as public blockchain networks. The most popular persmissionless blockchain networks are Bitcoin, Ethereum [124], etc. Permissioned blockchain networks are private networks where the participating nodes and the users are known to each other or belong to a central organization, group, etc. Most popular permissioned blockchains are Hyperledger [19] and Corda [20]

Ethereum, adapts the concept of smart contracts. The smart contract is a set of binary code, similar to a computer application, that runs on top of the blockchain. Once the smart contract is deployed on the blockchain, it is immutable and operates independently (from its creator) with its blockchain address. Users send transactions with input data to the smart contract, which in turn executes itself (bytecode) to produce output data, that can be permanently stored on the blockchain. With the adop-

[19] https://www.hyperledger.org/ [Accessed: Jul. 14, 2022]
[20] https://www.corda.net/ [Accessed: Jul. 14, 2022]

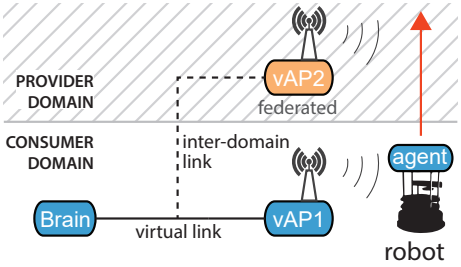


Figure 4.1. *Edge service*

tion of smart contracts, the application of blockchain technology significantly expands in solving lots of general problems that require a middle-man in the process.

To boost the federation process in secure manner, in [125] we propose to squeeze the whole service federation process (from Section 1.1.6) to run on a DLT. More specifically, the federation procedures are to be stored and deployed on a Federation Smart-Contract (SC) which is running on top of a permissioned blockchain. The design of the Federation SC is completely open. The focus of the smart-contract design is to maintain neutrality and privacy while overseeing the federation procedures that involve all ADs.

Each domain sets up a single node as part of the peer-to-peer blockchain network. The distributed nature of the blockchain network allows scalability while maintaining security. The ADs communicate with the Federation SC through transactions. The transactions are recorded in the blocks. The sealing or generation of blocks depends on the consensus protocol. The choice of the consensus protocol would determine the speed and the security level of the federation process. For example, the Proof-of-Authority (PoA) consensus increases the speed, while the Proof-of-Work (PoW) mechanism increases the security of the blockchain.

Each new joining AD establishes connectivity with at least a single node in the blockchain network using a new and locally deployed node. Then, it registers to the Federation SC with a single-transaction registration using its unique blockchain address. In the single-transaction registration, the Federation SC records the information of the registering AD and its service footprint. This way the registration procedure explained in Section 1.1.6 is relatively simple to be realized. Once the registration procedure is successfully completed, the AD is ready to consume or provide federated services. More details about how DLT can be applied for the federation procedures can be found in [125].

DLT-federation for mobile edge robotics service

The combination of radio context information and location coordinates allows the robot to move within the boundaries of a single administrative domain. Through the application of service federation, an edge robotics service would not be limited (to a single AD) and it would be able to extend the desired service footprint at any time, anywhere. A simplified service federation of an edge robotics service is illustrated in Figure 4.1. All colored blocks represent MEC apps as VNFs. The blue blocks

present the MEC apps of an exemplary edge robotics service deployed in a consumer domain. The robot brain contains the control logic that provides movement instructions to a robot agent, through the virtual access point (vAP1). The robot, via the agent, executes the movement commands using its actuators and provides real-time sensor data back to the robot brain. In short, this is a closed-loop that allows the robot brain to control the robot to accomplish different tasks (for more details refer to Section 2.2). When the robot leaves the coverage area of the consumer domain, a service federation is initiated by the consumer domain. The service federation includes the deployment of a new virtual access point ("vAP2") in a provider domain that can ensure extended network coverage. The federated "vAP2" establishes an overlay connection to the robot brain through an inter-domain link. Once the end-to-end connectivity is established, the closed-loop between the "Brain" and the robot continues through the federated "vAP2" without any service interruption.

4.2.2. OKpi: All-KPI Network Slicing Through Efficient Resource Allocation

Although several works have already addressed the VNF placement problem, the variety of KPIs introduced by 5G poses some relevant challenges that still need to be solved. Such KPIs are indeed both diverse and heterogeneous over different services, as well depicted by the ubiquitous ITU “pyramid” [126]: *diverse* as, for instance, the latency requirements of different use cases can vary by several orders of magnitude, while *heterogeneous* reflects the fact that 5G introduces several new performance metrics, including service availability (in both space and time) and service reliability. Satisfying all the relevant KPIs through an efficient resource allocation thus requires casting the problem into a new formulation and envisioning a totally new solution.

Additionally, existing studies have tackled to a limited extent specific aspects of network slicing, including *i)* the possibility that already-deployed VNF instances can be reused for newly requested services, with [127] only accounting for cost, *ii)* the opportunity of combining cloud- and edge-based services (with the exception of [128], which however only deals with caching), and *iii)* the need to make decisions on how to place *and* connect VNFs, thus jointly addressing VNF placement and data routing ([129–132] do so, but without considering points of attachments or VNF re-usage, and under some limiting assumptions, e.g., on the number of VNF instances). Overall, extending existing solutions to account for all 5G KPIs and the need for an efficient and low-cost resource utilization, would not be trivial and would, in general, jeopardize the complexity and/or competitive ratio properties of such solutions.

In [121] the authors propose OKpi, an efficient solution that can create high-quality, end-to-end network slices. This work presents a system model that captures the main aspects of NFV-based networks and can represent the availability of resources at different layers of the network topology, namely, cloud, edge, and fog, as well as the fact that existing VNF instances can be reused for newly-requested services. Leveraging a graph-based representation of the available resources, the possible decisions, and their impact on the KPIs, OKpi can make joint decisions on VNF placement and traffic routing that minimizes the cost of the resources, by applying a shortest path algorithm over a multi-dimensional graph. Importantly, such a graph can be built with different levels of detail and size,

which results in a tuneable trade-off between computational complexity and decision quality.

OKpi for mobile edge robotics service

One of the paramount issues in robotic systems is to meet all target KPIs required by the mission-critical services, in spite of the heterogeneous and limited resources that are available in the mobile network. Let's consider security or cleaning services that are provided by mobile robots on the factory floor. In the factory, different RATs can be available (e.g., Wi-Fi, 4G or 5G) for the mobile robots to use. Moreover, the factory infrastructure can provide heterogeneous computing resources where the mobile robots can offload their processing. These computing resources can range from resource-limited industrial mini PCs that are deployed on the factory floor to powerful edge and cloud servers deployed in the factory data centers. Through the application of orchestration algorithms like OKpi, an edge robotics service would be able to use the available RATs and computing resources in the most efficient way in order to fulfill the tight requirements of autonomously controlled mobile robots.

4.3. DLT federation for mobile robots

To prove the feasibility of the DLT federation for edge robotics and evaluate the solution, we used part of the UC3M testbed presented in Section 2.3.1 where on top we run a trusty & untrusty blockchain.

4.3.1. Experimental setup

The hardware and software components that were reused from the UC3M testbed are shown in Figure 4.2. The testbed consists of a single robot, Ethereum blockchain node, and two administrative domains (ADs) - consumer and provider domain - with their underlying infrastructure. The consumer domain infrastructure consists of two hosts depicted as edge server and vAP3 on Figure 4.2. KVM and LXD virtualization is running on top of the edge server, while only LXD on top of vAP3. Both hosts are orchestrated by the consumer orchestrator which in this case is a simple custom-developed orchestrator determined for the whole scenario process. An on-boarded edge robotics service is similar to the service described in Figure 2.2. The consumer orchestrator deploys the edge robotics service over the underlying infrastructure (edge server & vAP3) through the distributed Virtualized Infrastructure Manager (VIM) - Fog05[21]. As described in Section 2.3.1, the mobile edge robotics service is deployed as VNF-MEC apps (shown as blue rounded boxes on Figure 4.2):

- robot brain is a MEC app deployed over the edge server.
- vAP is deployed over the vAP3 as hostapd, inter-connected through the virtual link to the robot brain.

[21] https://fog05.io/ [Accessed: Jul. 14, 2022]

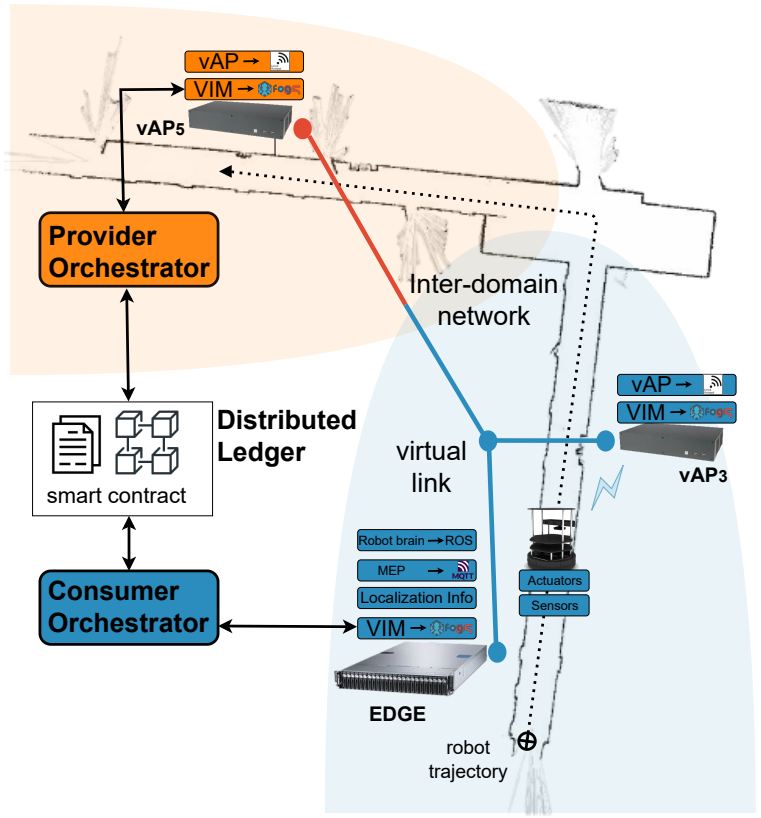


Figure 4.2. *Edge robotics experimental test-bed & scenario*

- Sensors and Actuators are deployed over the robot hardware. The robot hardware consists of motor wheels as actuators, 802.11 connectivity, and sensors (lidar & odometry).

A MQTT broker is substituting the role of a MEC platform. The robot brain, as a main MEC application, is consuming a Localization MEC service via the MQTT broker.

The provider domain is isolated from the consumer domain. Contains a single host (illustrated as vAP5 on Figure 4.2). The provider orchestrator is a replica of the consumer orchestrator that orchestrates the virtualized infrastructure (LXD) through a new instance of Fog05. The provider orchestrator has only the on-boarded image of the vAP MEC application on Fog05 (illustrated as orange rounded box on Figure 4.2). The Distributed Ledger contains two instances of Ethereum blockchain. The instances are deployed over a virtual machine on a server at the university network. Both instances contain a Federation SC. The first instance is running Proof-of-Authority (PoA) consensus for trusty communication, and the second instance Proof-of-Work (PoW) for untrusty communication.

4.3.2. Experimental methodology

The experimental scenario is mimicking a real use-case where the robot is instructed to provide security or cleaning services in an area at the university, following a path as illustrated with the black

dotted line in Figure 4.2. In order to finalize the task, the robot needs to drive from the blue (consumer) domain to the area of coverage of the orange (provider) domain. The robot brain is aware of the real-time robot's location by consuming the localization MEC Service. The brain triggers the federation procedure to the consumer orchestrator when the robot approaches the boundaries of the vAP3 coverage. On triggering event, the consumer orchestrator proceeds with the federation procedure. The provider domain, as a winner, establishes an overlay inter-domain link to the consumer domain and deploys the vAP5 (as depicted on Figure 4.2). After the deployment of the federated vAP5 has finished, the provider orchestrator confirms the deployment to the Federation SC by storing the BSSID of the deployed vAP5. The consumer domain delivers this information to the robot brain. Finally, the robot brain instructs the robot, or the drivers, to switch connectivity to the BSSID of vAP5. The driver connects to the vAP5 while the closed-loop (robot brain to robot) is not broken. The closed-loop data in both directions starts passing through the overlay inter-domain link.

We evaluated the time performance of the edge robotics federation using DLT by running the experimental scenario as described in above. We run the experimental scenario using *i)* the PoA-based blockchain instance that uses trusty communication between the domains, and *ii)* the experimental scenario with the default PoW-based consensus and untrusty communication. As already mentioned, in the untrusty communication the consumer provides only the inter-domain link endpoint, and the provider domain provides back the BSSID of the vAP5, upon deployment. We made several experimental runs for each of the PoA-based and PoW-based scenarios. In the rest of the section, we present the average times for each step in the process.

4.3.3. Experimental results

Three graphs of the time it takes to finalize all the federation procedures are shown in Figure 4.3. In all graphs the time bars are colored:

- orange - for all federation-related procedures as described in Section 1.1.6.
- blue - for all procedures that involve the deployment of the edge robotics service or part of it.

To that end, the top graph of Figure 4.3 presents the accumulated times of the federation procedures in both the consumer and provider domain. The average federation time is 19.038 seconds - or the time it takes from the trigger at the consumer orchestrator to the robot connected to the vAP5. The breakdown in all phases that occur in the consumer domain is presented in the middle graph of Figure 4.3. It takes 12.97 seconds for the deployment of the vAP5 to be confirmed at the consumer domain (or phase "federation completed"). In other words, the consumer domain retrieves the BSSID of vAP5 in the provider domain in 12.97 seconds. Then it takes around 6 seconds for the brain to instruct the robot to discover vAP5, disconnect from vAP3, and connect to vAP5.

The bottom graph of Figure 4.3 breaks down all the phases in the provider domain, that occur within the previously mentioned 12.97 seconds. The negotiation or bidding process until the provider domain is elected as a winning provider takes 3.98 seconds. More specifically, it takes 3.98 seconds

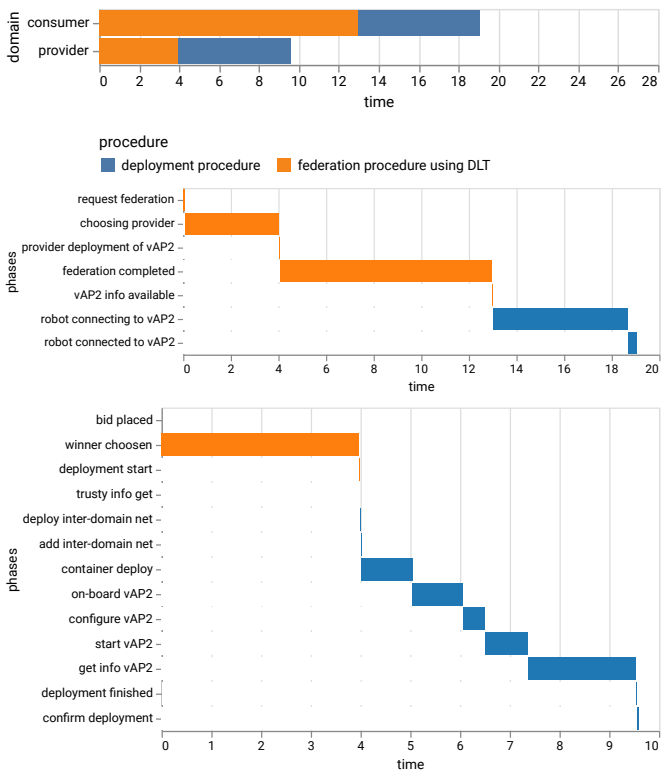


Figure 4.3. *Federation using trusty communication - PoA consensus: (top) summarized phase times; (middle) consumer AD; (bottom) provider AD;*

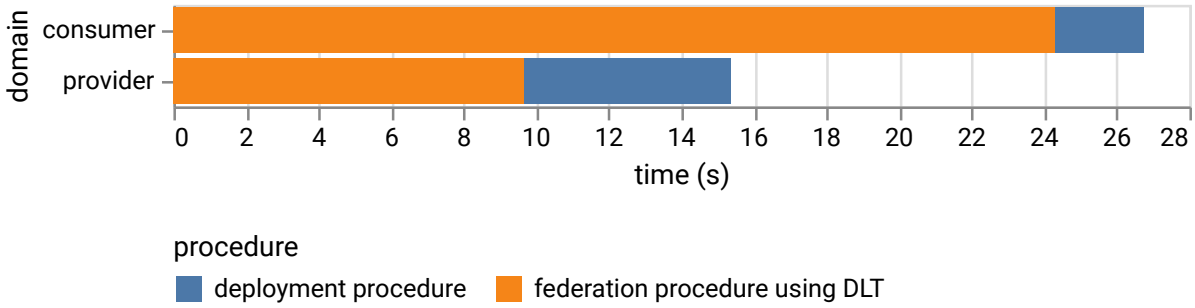


Figure 4.4. *Federation using untrusty communication - PoW consensus: summarized times*

from the time that the provider domain receives the broadcast announcement until the deployment starts. The establishment of the inter-domain link, on-boarding & instantiation of the vAP5 takes additional 5.58 seconds.

The results of the PoW-based scenario and untrusty communication is shown in the Figure 4.4. The graph shows only the accumulated times for both domains. Compared to the PoA-based solution, it is clear that the PoW-based solution takes significantly more time to negotiate and complete the federation process using the Blockchain/DLT. Due to the PoW consensus mechanism the "federation completed" phase is completed within 24.3 seconds, nearly double the time of the PoA-based solution.

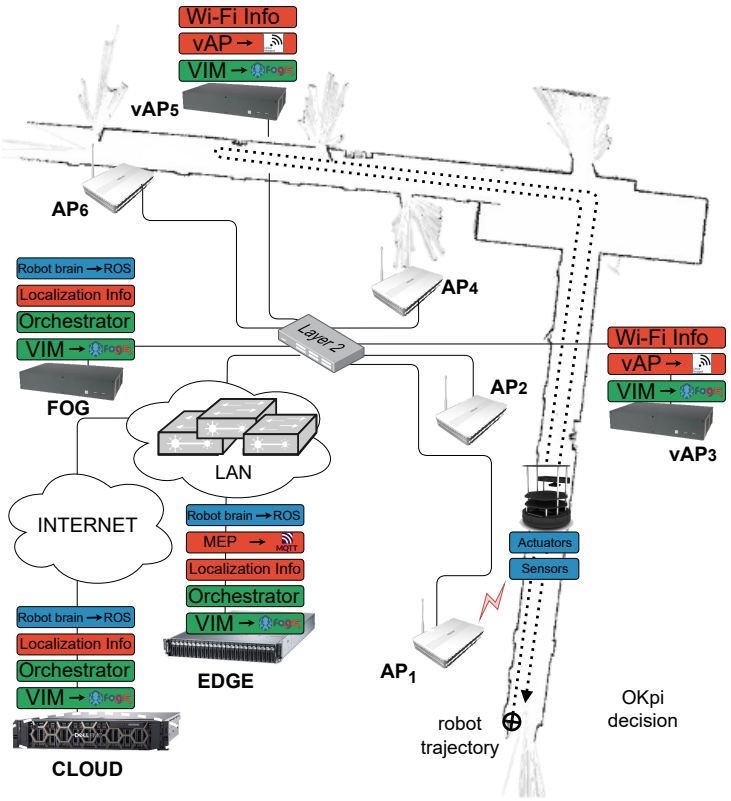


Figure 4.5. *Illustration of the mobile robot, the UC3M testbed: the robot starts at the bottom end of the corridor and comes back once it has reached the other end at the top left. Dashed circles highlight possible VNF deployments.*

4.4. OKpi for mobile edge robotics service

To prove the feasibility of applying OKpi for edge robotics and evaluate how an orchestration algorithm can fulfill the required KPIs in a heterogeneous testbed, we have used the UC3M testbed that was presented in Section 2.3.1 where on top we deployed an experimental prototype of OKpi.

4.4.1. Experimental setup

Figure 4.5 shows the UC3M testbed that was used for experimentation. The testbed consists of: *i)* 4 APs; *ii)* 3 MiniPC, two used as an AP and one as a local fog server (i.e., located close to the APs); *iii)* 1 edge server; and *iv)* 1 cloud server. The six APs and the local server are deployed along two corridors of the Universidad Carlos III de Madrid building (see Figure 4.4), while the edge and cloud server are located in different buildings. To emulate different levels of link congestion and heterogeneity, we leverage NetEm [133] to artificially introduce some latency on the connection between the APs and the servers, as reported in Tab. 4.1. Note instead that the latency on the AP-robot link never exceeds 6 ms. Additionally, we used a single ROS-compatible Turtlebot S2 robot equipped with a laptop and

lidar.

The laptop hosts the Sensor and Actuator VNF, which, as explained in details in Section 2.2, *i)* probes the robot sensors (e.g., odometry, LIDAR), *ii)* transmits the sensors data to the robot brain, and *iii)* executes the instructions received from the robot brain or OKpi. The robot brain hosts an autonomous navigation algorithm that based on the robot sensors data navigates the robot in the university hallway. The robot brain VNFs can be hosted at any of the available servers. For more details about the hardware specifications of the testbed (e.g., mobile robot maximum speed, lidar range, mini PCs/servers capabilities) or the software implementation of the edge robotics system for mobile robots please visit Section 2.3.1.

4.4.2. Experimental methodology

Table 4.2. *Testbed scenario: AP-server latencies*

Access Point	**Cloud**	**Edge**	**Local**
$AP1_1$, AP_2	9 ms	4 ms	3 ms
AP_3, AP_4	18 ms	8 ms	9 ms
AP_5, AP_6	27 ms	12 ms	9 ms

The target of the experiment is to ensure that the End-to-End (E2E) latency of the edge robotics service remains within 15 ms [134] during the robot's drive. The experiment starts with the robot positioned at the bottom end of the corridor and connected with AP_1 (see Figure 4.5). Also, the initial decision by OKpi is to deploy the robot brain VNF in the cloud server. The robot brain then navigates the robot along the trajectory shown in Figure 4.5 and, as the robot moves, OKpi determines which AP the robot should connect to[22] and which server (cloud, edge, fog) should host the robot brain VNF. Both the AP and server selection change according to the robot position and latency of the AP, respectively. In particular, depending on which AP the robot is attached to, OKpi decides which server should host the robot brain VNF by accounting for the latency values reported in Table 4.2, in such a way that the overall service latency remains below 15 ms. During the experiments, OKpi recomputed the AP selection and robot brain VNF embedding, in less than 1 s. The robot position is reported to OKpi through the Localization info service that is based on the robot sensor data, while the APs coverage, which may vary over time, is acquired through locally available channel measurements (e.g., signal level).

4.4.3. Experimental results

Figure 4.6 compares the temporal behavior of the E2E latency obtained in the following cases:

[22] As the robot moves, it roams to the selected AP using 802.11r Fast Transitioning [63].

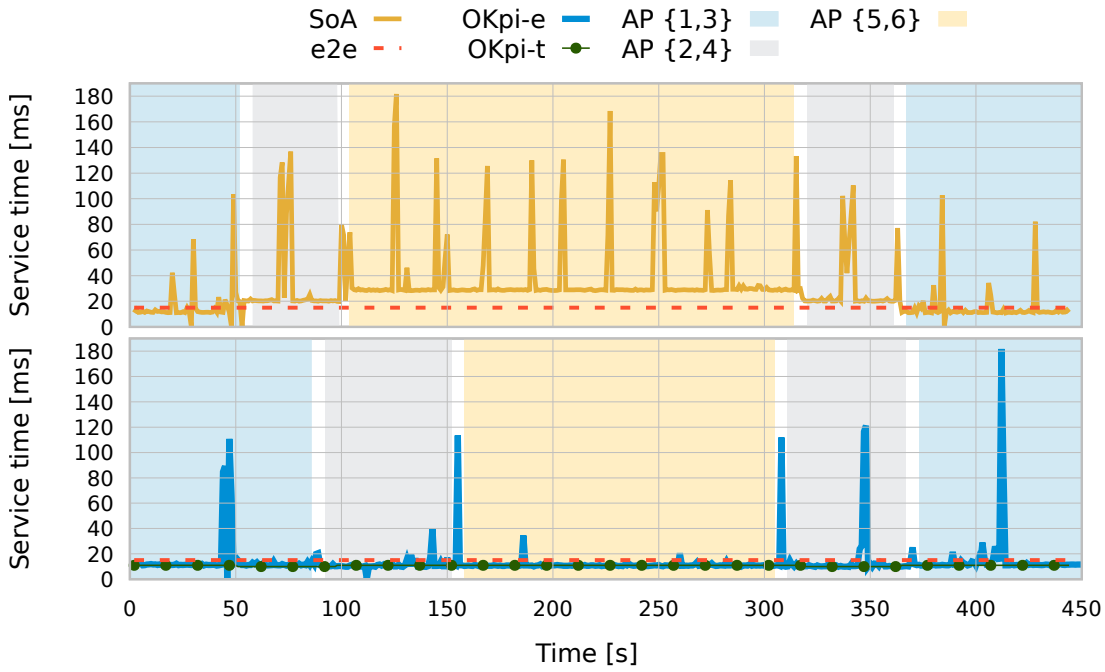


Figure 4.6. *Service latency experienced during the robot's trip. Different background colors refer to time intervals in which the robot is connected to different APs.*

- **SoA**, i.e., when the solution based on [62] and [135] is used. In this case, the brain VNF is always placed in the cloud and the robot performs roaming checks every 20 s in order to ensure connectivity to the AP with the strongest signal. The roaming checks consist of Wi-Fi active scan to acquire the signal-related information, followed by automatic handover. For our testbed, we discovered that 20 s was the lowest check frequency that allows the robot to perform successful roaming along the navigated trajectory.

- **OKpi$_t$**, i.e., when OKpi makes decisions in a simulated environment that mimics the testbed, using the theoretical delays that each AP offers.

- **OKpi$_e$**, i.e., when OKpi operates in the developed testbed, it uses the robot's real-time location and available channel measurements to trigger the 802.11r roaming. The roaming message is send via MQTT to the robot Actuator VNF. In this case, the periodic roaming checks are prevented and the robot is configured to perform handovers only when a roaming message is received.

Figure 4.6 compares the service time measured during a single run execution of the SoA, OKpi$_e$, and OKpi$_t$ solutions. At the beginning of the experiment, neither the SoA nor the OKpi$_e$ violate the target E2E latency of 15 ms, except for the peaks due to the robot Wi-Fi scans. While under SoA the robot performs a scan every 20 s, under OKpi$_e$ it does so only when it has to connect to a new AP, as per the OKpi decision. Under SoA, 50 s later the robot connects to AP3, and the latency jumps above 20 ms because the robot brain VNF are still running in the cloud server. In the case of OKpi$_e$, instead,

when the robot connects to AP3 at 89 s., the robot brain VNF are moved to the edge server, so that the E2E latency remains below 15 ms. The same behavior is observed at time 102 s, when the robot connects to AP5 in the case of SoA and the E2E latency increases up to 29 ms. On the contrary, $OKpi_e$ still meets the target E2E latency upon making the robot connecting with AP5 (at time 156 s), since it now places the VNFs in the local server. In the rest of the time elapsed, we can observe similar performance, as the robot returns to its initial position.

Table 4.3. *OKpi and SoA service latency*

Solution	**Average**	**Std. deviation**	**E2E violations**
$OKPi_e$	13.63 ms	15.21 ms	7%
SoA	29.61 ms	26.18 ms	74%

As a final remark, Figure 4.6 highlights that the performance of $OKpi_e$ is always close to the E2E latency exhibited by $OKpi_t$. Furthermore, the target E2E service latency (15 ms) was only violated the 7% of the times during the experiment by $OKpi_e$ (see Table 4.3), while it was violated the 74% of the times under SoA.

4.5. Conclusions

In this chapter of the thesis, we showed how the edge robotics concept enables the application of novel orchestration solutions for mobile robots to meet the connectivity and KPI requirements for uninterrupted service operation. In particular, the environment considered was the UC3M testbed which employs the designed edge-based robotic system for mobile robots. We extended this testbed by implementing OKpi as an orchestration algorithm and Ethereum based Distributed Ledger solution for federation. Experimental results demonstrated the functionality of OKpi and its ability to make effective decisions when applied in an edge robotics scenario. In addition, the results demonstrated how the complete DLT federation process in an edge robotics scenario can be concluded in around 19 seconds with a lower level of security. In this context, this chapter addressed **C5** and **C6** that were defined in the scope of this thesis by showcasing:

- DLT-based federation process can extend the robot wireless connectivity where the complete federation process is concluded in around 19 seconds with a lower level of security. In 42% of the federation time, the consumer domain generates an announcement, collects bids, and chooses a winning provider domain. For higher security and anonymous negotiation among domains, the federation concludes in 28 seconds.

- OKpi that can meet the target E2E latency (15 ms) of an autonomously controlled mobile robot by selecting the correct Point-of-Attachment and VNFs location. In this context, the Okpi experimental analyses show how OKpi violates the target E2E latency (15 ms) only by 7% of the time, while the SoA solution violated it for 74% of the time during the experiment.

CHAPTER 5

AI-assisted control for edge robotics

Industry 4.0 aims at supporting smarter and autonomous processes while improving agility, cost efficiency, and user experience. To fulfill its promises, properly processing the data of the industrial robots and infrastructures are required. AI appears as a strong candidate to handle all generated data and to help in the automation and smartification process. Additionally, Industry 4.0 claims wireless communications as a game changer for future manufacturing plants, enabling flexible production chains, as machinery and other components not to be restricted to a location by the rigid wired connections on the factory floor. However, the presence of electromagnetic interference in the wireless spectrum may result in packet loss and delay, making it a challenging environment to meet the extreme reliability requirements of industrial applications. In such conditions, achieving real-time remote control, either from the edge or cloud, becomes complex.

In this chapter, we overview the networked robots as a true embodiment of a Cyber-Physical System (CPS) in Industry 4.0, showing the mission of AI in such concept. We present the key enabling technologies of the networked robots such as edge, fog, and 5G, where the role of AI in each technology domain is identified by analyzing a set of AI agents at the application and infrastructure level. Next, we select one of the proposed AI agents and we investigate the feasibility of applying a forecast-based recovery mechanism for real-time remote control of robotic manipulators (FoReCo) that uses ML to infer lost commands caused by interference in the wireless channel. We evaluate FoReCo through both simulation and experimentation in interference-prone IEEE 802.11 wireless links and using a commercial research robot that performs pick-and-place tasks.

This chapter is organized as follows. Section 5.1 introduces the work and highlights the main

contributions of the chapter. Section 5.2 presents the envisioned concept of Industry 4.0 that incorporates robots, computation and networks, focusing on the specific aspects of the networked robots that can be enhanced with AI capabilities. The identified AI agents, both at application and infrastructure level, are also discussed in Section 5.2. Section 5.3 studies the feasibility of the Movement Prediction AI agent where Section 5.3.1 reviews the related work, followed by the problem statement that is defined in Section 5.3.2. Then, Section 5.3.3, presents FoReCo (solution based on the Movement Prediction AI agent) and elaborates on how FoReCo infers delayed/lost commands through ML. Section 5.3.4 explains the analytical model of IEEE 802.11 that that is used to test FoReCo in simulated scenarios with wireless interference. Later in Section 5.3.5 FoReCo is validated via simulation and real experiments and Section 5.3.6 discusses the main insights from the obtained results. Finally, Section 5.4 concludes the chapter.

5.1. Introduction

The rapid advancements in ICTs are transforming the industrial sector towards a full digitalization and integration concept. This transformation, known as Industry 4.0, enhances industrial systems where innovations such as robotics, automation, and artificial intelligence are set to take over. Consequently, the industrial world can improve productivity, logistics, and lower production costs [136]. Cyber Physical Systems (CPSs) are the main linchpin for Industry 4.0 to move towards a fully automated industrial infrastructure that relies on real-time capabilities, distributed control systems, virtualization, service orientation, and modularity [137].Networked robot is defined as a robotic device connected to a computing infrastructure via a communications network such as the Internet or LAN [138]. This concept truly embodies the cyber-physical integration within Industry 4.0, combining any industrial process achieved through closed-loop feedback mechanisms. The computing infrastructure hosts virtual models of tools, machines, operatives, products, etc., as well as behaviors, rules, physics, and analytic models that analyze the robot sensor data. The outputs of the robot algorithms that reside in the computing infrastructure are executed in the robots via wired or wireless networks to improve the robot performance.

One example of a networked robot application that is seen as the key enabler for Industry 4.0 is wireless real-time remote control and coordination of robots. While wireless is a must for mobile robots like Autonomous Guided Vehicles (AVGs), the implementation of wireless connections for robots manipulators also has many advantages such as greater flexibility, reduction of installation and maintenance costs, ease of scale, and less personnel exposure to hazardous situations [72]. Industry 4.0 scenarios will decide whether to use wireless technologies in the licensed spectrum, such as 5G NR [73] or in the unlicensed spectrum, such as IEEE 802.11 [74].

Nowadays, industrial verticals implement IEEE 802.11 technologies for factory automation through commercial solutions such as Industrial WLAN (IWLAN) developed by Siemens. The low cost, good performance (e.g., low latency, high throughput), and extensive implementation in commercial equipment make IEEE 802.11 a suitable candidate to fulfill the tight timing constraints of industrial au-

tomation. However, achieving the reliability, transparency, and stability for real-time remote control required in many applications remains a critical challenge in IEEE 802.11. due to the highly unpredictable, unreliable, and interference-prone wireless channel, which introduces delays, packet loss, jitter, throughput bottlenecks, and even loss of connectivity [48]. The presence of packet collisions and electromagnetic (EM) interference in the shared medium results in delayed or even lost control commands. While the delayed delivery of control commands to the robot breaks the transparency of the remote control system (e.g., lag between the executed remote control commands and the robot movements), the lost commands directly influence the stability resulting in a deviation from the desired trajectory.

In this sense, the cyber space mirrored through the networked robots arises as the perfect playground for the development of AI agents [139] that can be used to solve existing challenges in CPSs. Moreover, ML is a strong candidate to implement such agents, as an alternative to heuristic or decision-tree based solutions, among others. Networked robots provide the tools for transferring the domain expertise of specialized personnel into raw data in the cyber space, which can be later used to train and cross-validate different ML algorithms used in AI agents. These agents not only develop expertise in specific tasks but also extend and optimize it beyond human capability due to the volume of data they can handle to make decisions. Ultimately, smarter and more accurate robots can be devised where autonomy is achieved through AI-controlled processes that operate in all types of environments and conditions.

This chapter first overviews the networked robots as a CPS solution, where AI is introduced as the missing piece in its integration with networks and computing. In particular, it focuses on identifying AI agents, both at the application and infrastructure level with relevant applicability to improve and enhance existing robotic systems. Next, based on the identified AI agents, we propose FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators. FoReCo is suitable for autonomous or human-assisted remote control of robot manipulators that perform repetitive tasks such as welding, materials handling, picking and placing, or assembly. In case the robot does not receive a remote control command on time due to IEEE 802.11 collisions or EM interference, FoReCo *i)* infers the delayed command; and *ii)* injects it in the robot driver loop so the operator does not perceive misbehavior in the remote control process.

This chapter contributes to the state-of-the-art as follows:

- we introduce a re-modeled concept of the networked robot, where cloud, edge, and fog computing are integrated with emerging networking technologies such as 5G, Wi-Fi 6E, and physical processes.
- we identify and analyze exemplary AI agents for the networked robots, spanning from the application to the infrastructure level.
- we formulate an optimization problem to minimize the trajectory error due to delayed and lost real-time remote control commands.
- we propose FoReCo to infer delayed and lost commands using ML algorithms.

- we validate FoReCo via simulation using an IEEE 802.11 analytical model [140] that accounts for an interference source, and packet collisions due to channel neighbors.
- we show experimentally that FoReCo works in a commercial research robotic arm, and mitigates the effects of electromagnetic interference created with a real jammer.

5.2. AI benefits for edge robotics

5.2.1. Towards Intelligent Integration of robotic systems with computing and networks

This section overviews the networked robotics concepts, and their integration with the underlying computing and network infrastructure, emphasizing still open challenges to be tackled by AI.

Networked robots

Networked robots in Industry 4.0 integrates any industrial process achieved through the implementation of closed-loop feedback mechanisms. It integrates advanced algorithms of the robots in the cyber space, that analyze the robot and environment sensor data, and provide feedback mechanisms for control. The control-loop starts with the robot sending sensor information and its current state to the algorithms, which then closes the control-loop by sending back commands in real-time. In doing so, industrial machinery become software-enhanced objects that incorporate self-management capabilities and respond quickly to changes. In this way, cyber-physical integration is achieved by providing a new set of tools to monitor, control and predict behaviors, and accurately optimize the factory floor.

Operational digital twin applications are an example of industrial networked robot applications that demonstrate superiority over the traditional solutions in the areas of design, production, and system health checks. These allow reinforcing the collaboration between design and manufacturing, mimicking the real factory environment to ease remote control operations, and facilitate the detection of machinery problems, respectively.

Computing

In Industry 4.0, robots are composed of either low-performance and constrained hardware or hardware tailored to a specific task. Owing to the development of virtualization, software components of the robots can be represented as modular virtualized functions, which execution is outsourced into more powerful computing resources. Cloud-based solutions have been initially exploited for implementing such concepts [141], by providing elastic and powerful computing capabilities required to support the networked robots. However, cloud providers cannot ensure the performance of the network between the robots and the algorithms that reside in the network, worsening with their network distance and the number of providers in-between. As a result, cloud-based robotic systems

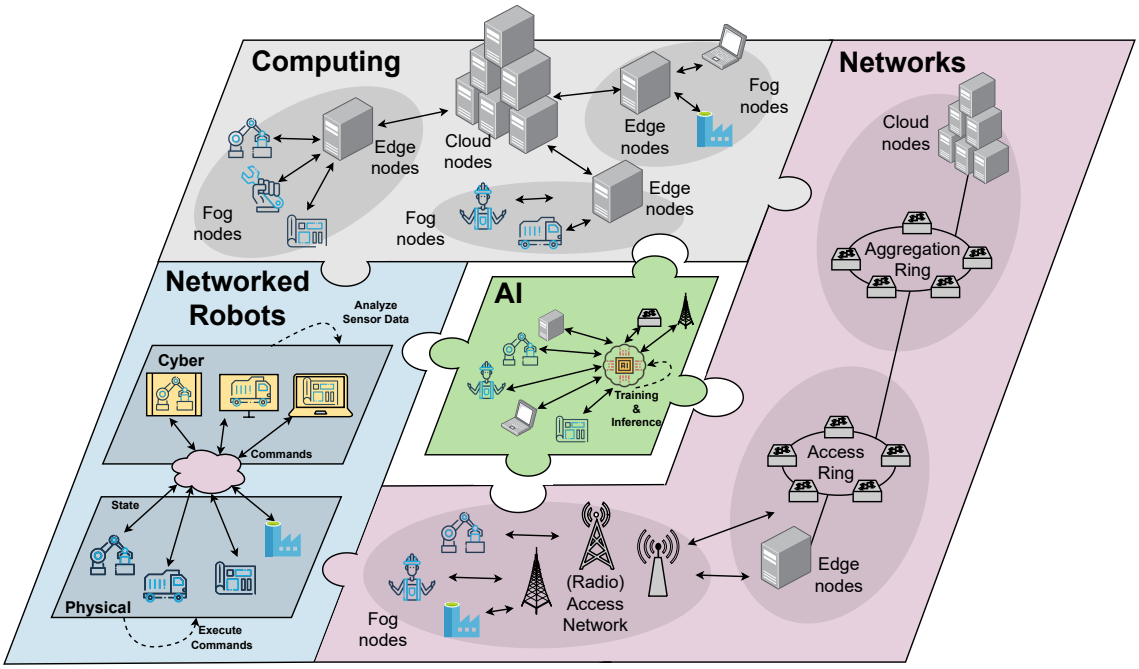


Figure 5.1. *General concept of Networked robots*

suffer from time-varying network delay, unpredictable jitter, limited bandwidth, or data loss. These drawbacks prevent time-sensitive tasks, including real-time remote control, to be fully supported by the cloud computing substrate. To overcome the shortcomings of cloud computing, edge and fog emerged as a natural extension. While edge computing provides computing capabilities near the physical objects via static substrates, fog computing also integrates volatile, constrained, or mobile resources (including the physical objects). By exploiting edge and fog computing, the networked robots can offload time-sensitive processing from the robots, which in turn contributes to further optimizations of the hardware costs. Additionally, new algorithms for efficient data filtering, envisioning privacy and security improvements [142], can be applied and the data can be restricted within a trusted private infrastructure. Finally, due to the close proximity, edge-based robotic systems can use the available radio network information to adapt the operations of the robots or to optimize resource allocation in order to improve the QoE.

Networks

The underlying network infrastructure of the networked robots comprises different dynamic and heterogeneous topologies. It can be divided in three segments, as shown in Figure 5.1: *(i)* Aggregation Ring; *(ii)* Access Ring; and *(iii)* (Radio) Access Network ((R)AN). The Aggregation Ring resides far from the robots, relying on wired connectivity to connect cloud-based robotic systems that are suitable for human-scale responsive services and delay-tolerant tasks (e.g., monitoring). The Access Rings go closer to the robots, interconnecting multiple (R)ANs. The Access Rings are locally present and ex-

pose radio network information (e.g., radio channel) to edge-based robotic systems, namely for time-sensitive tasks (e.g., remote manipulation). Finally, the (R)AN is in the vicinity of the factory floor, providing a connection to the robots using both wired and wireless connectivity. Different radio access technologies (RATs) are available (e.g., Wi-Fi, 4G, 5G), differing in their capabilities with respect to latency, range, data rate, power profile, and scalability.

Wired technologies are most suitable for fulfilling the communication requirements of networked robots. Due to their limitations in terms of flexibility, mobility, and high-density connections, wireless technologies are becoming more appealing in the (R)AN. However, the critical processes within industrial environments are sensitive to radio-frequency interference, requiring RATs to be interference-free, to work on licensed bands, and to provide an extremely controlled environment. Industry 4.0 claims 5G as a key enabler to fulfill the communication requirements set by networked robots [143], not only through radio enhancements but also by employing network slicing and virtualization as core features. At the same time, Wi-Fi 6E appears as another candidate for Industry 4.0, with trials already showcasing its capability to sustain the presence of interference and noise and to meet the stringent requirements of most use cases.

The aforementioned opportunities demand network-assisted robotic systems that satisfy the expected real-time and secure performance. On top, the network-assisted robotic systems should also tackle the problems derived from their integration with the network and computing infrastructure. AI agents are strong candidates to handle such challenges, as they can benefit from ML algorithms to exploit existing data sources with context information at both the application, and infrastructure level.

5.2.2. AI agents for robotic systems

In order to realize the opportunities that are presented in the previous section, exemplary AI agents for networked robots are identified (Table 5.1), including possible ML algorithms [144] to implement them. The *In-Network* or *On-Device* deployment strategies are envisioned while leveraging the pervasiveness of the cloud-to-things continuum (i.e., fog, edge, and cloud). Moreover, AI agents can be trained in the cloud (cloud learning) for computation-intensive training, or in the edge (edge learning) for local training considering enormous real-time and private data generated by the industrial processes.

Application Related Enhancements

The following AI agents describe different enhancements on top of networked robots, which improve the robustness and reliability of the existing processes and pave the way for novel features and capabilities that rely on highly automated processes.

1) **Movement Prediction:** Remotely controlling a robot over a wireless channel from the edge or cloud can be prone to unpredictable radio-frequency interference that introduces high jitter

Table 5.1. *Summary of AI agents For Networked robots*

	AI agent	Input Data	Outcomes	ML algorithm(s)	Location
Application	Movement Prediction	Historic of commands, real-time commands	Predictions on the N next commands	VAR, TCN, GRU, LSTM	Fog, Edge
	Task Learning	Demonstrations of the task from different knowledge domains (e.g., physical object states)	Generalized task policy	IL, RL	Fog, Edge
	Risk Reduction	Sensor data, video streams, localization data and machinery states	Identification and forecasting unsafe situations	CNN	Fog, Edge
	Predictive Maintenance	Machinery and environmental sensor data (e.g., motors status, vibration, temperature)	Failure predictions	ARIMA, LSTM + LR, SVM	Edge, Cloud
Infrastructure	Dynamic Scaling	Resource usage, date and time, task, number of instances, application KPIs and SLAs	Scale in/out or up/down suggestions	RL, RT, RF, MLP, BN	Edge, Cloud
	Privacy, Security and Intrusion Detection	Infrastructure and network context information, traffic flows patterns, service and infrastructure KPIs	Security breaches and suspicious flows	PCA, K-means, Autoencoders	Edge, Cloud
	HetNet Selection	Radio network information, available resources, mobility patterns, application KPIs and SLAs	RAT and handover candidate selection	RL, ANN, Fuzzy Logic	Fog, Edge

ANN: Artificial Neural Networks; ARIMA: Autoregressive Integrated Moving Average; BN: Bayesian Network; CNN: Convolutional Neural Networks; GRU: Gated Recurrent Unit; IL: Imitation Learning; LR: Logistic Regression; LSTM: Long-Short Term Memory; MLP: Multi-Layer Perceptron; PCA: Principal Component Analysis; RL: Reinforcement Learning; RF: Random Forest; RT: Random Tree; SVM: Support Vector Machines; TCN: Temporal Convolutional Networks; VAR: Vector Autoregressive.

and packet loss. Consequently, the remote operator experience lagged behavior that breaks the real-time control of the physical object and creates an unsafe environment. A solution that recovers from such unpredictable behaviors, keeping the remote control uninterrupted, is required. AI stands out as a strong candidate that can forecast future movements providing extra reliability in case movement commands are lost. In this context, Movement Prediction uses the historic of commands to predict future ones using ML time-series algorithms like VAR, TCN, GRU or LSTM (see bottom of Table 5.1 for description). Whenever the next command is lost or does not arrive on time, the Movement Prediction triggers the forecasting of such command to keep the remote control uninterrupted. To prevent packet loss and high latencies, the Movement Prediction has to be deployed in the fog, executing when a failure occurs, or in the edge, piggybacking predictions with every real command.

2) **Task Learning:** In industrial scenarios, there are still highly complex and dynamic tasks that require human expertise/presence. Traditional agents based on finite state machines are not suitable for automating such tasks, as they cannot react under unforeseen situations, such as the appearance of unpredictable obstacles. Task Learning AI agents based on IL and RL algo-

rithms are potential solutions to overcome such situations, as they are designed to learn and, afterward, interact with a dynamic environment. The Task Learning is first trained through observations of human-based operations, in a trial and error fashion, through the simulated environment enabled by a virtual replica of the robot. Then, its behavior is validated in the simulated environment, which includes unexpected and random situations. Finally, its run-time deployment is envisioned on the factory floor (i.e., fog or edge) to ensure secure, reliable, and low-latency execution of the task. For example, Task Learning is able to introduce generalization and adaptability to drive a lift truck for package delivery. Thus, the virtual replica can learn how to act robustly upon the introduction of obstacles in the path, or changes in the shapes or position of packages.

3) **Risk Reduction:** As remote control mechanisms emerge and robots become more autonomous, safety plays a critical role in the design of networked robots. When considering human-robot collaboration scenarios, failures of either humans or robots may suppose a risk to safety. Factory floors equipped with surveillance cameras could reuse them to perform image segmentation, and pattern recognition in order to identify and mitigate dangerous situations. Over the recent years, it has been proved that AI solutions based on CNN algorithms achieve the best performance on computer vision-related tasks. Hence, Risk Reduction uses CNN algorithms to identify dangerous situations by analyzing a video stream, helping the in-network algorithm to act preventively, such as blocking the robot or adapting its operation. Since fast counter-measures are required, its runtime deployment is best fit in the edge or, in scenarios with a higher degree of autonomy, in the fog. For example, based on a real-time video stream, Risk Reduction can detect that a human operator is in dangerous proximity to an industrial robot manipulator, and use this information to block the robot.

4) **Predictive Maintenance:** Industrial robots have always been held to a higher reliability and predictability standard than any general-purpose systems. Industrial companies consider unplanned downtime and emergency maintenance caused by failures a major challenge. For preventing eventual failures, the future state of a given robot component (e.g., gripper, joint) must be forecasted and classified in order to verify if it requires maintenance. ML-based solutions provide high accuracy to solve both prediction and classification problems. Thus, Predictive Maintenance AI-agent is a suitable candidate to preemptively detect failures or repair needs by using a combinations of algorithms as ARIMA, LSTM, LR or SVM. The Predictive Maintenance checks if the available sensor data might lead to failure situations and, if so, schedule the maintenance of the robot component. Since this AI-agent is not performing time-sensitive operations, it can be deployed anywhere from the edge up to the cloud. For example, if historical data reported high vibrations upon the break of a screw, the Predictive Maintenance can forecast future vibrations (e.g., LSTM), and decide if maintenance is required (e.g., SVM).

Infrastructure Related Enhancements

The following AI agents highlight several enhancements applicable to the computing and network domains, which have the potential to impact and optimize the performance of a robot.

1) **Dynamic Scaling:** With the recent development of virtualization technologies, smart factories benefit from having robotic systems coexisting under the same cloud-to-thing continuum. During the lifetime of a given application, an adequate scaling of resources is required, so that robot-related KPIs (e.g., latency) are satisfied without deteriorating the performance of others. Such a problem is analyzed in the existing literature as an NP-hard problem, that is, optimal scaling policies cannot be found in feasible run times. Consequently, AI solutions based on Markov Decision Processes can be used to find near-optimal scaling policies in feasible times, using algorithms based on RL, RT, RF, MLP and BN. The Dynamic Scaling follows scaling policies learned with the aforementioned algorithms, training with data such as resource consumption, date and time, task, number of instances, and sessions. The Dynamic Scaling can then compute scaling decisions in order to fulfill KPIs and Service Level Agreements (SLAs). The runtime deployment of this AI agent is most suitable on the network side (i.e., edge or cloud), depending on inference time and network latency towards the orchestrator. For example, whenever a new robotic arm is added to the factory floor, the Dynamic Scaling increases the allocation of vCPUs to the virtual instance in charge of holding its motion planning algorithms, allowing its processing delay to stay below a threshold.

2) **Privacy, Security and Intrusion Detection:** By employing networked robots in an industrial environment, huge volumes of network traffic are distributed in the cloud-to-things continuum in order to create a so-called digital factory. This makes the detection and diagnosis of security breaches and intrusions very challenging and complex for the infrastructure operators and their tenants. Performing an exhaustive analysis of all the network traffic would take a vast amount of time, which is unfeasible to early detect intrusions, or security breaches. ML learning algorithms, like PCA, K-means, or autoencoders, are ideal solutions to shrink traffic volume and speed up traffic inspection. Privacy, Security, and Intrusion Detection use these algorithms to detect malicious traffic and, consequently, block the remote control of robots through their remote controller. Moreover, federated learning and transfer learning appear as ML approaches that boost collaborative training across different industrial players, which, by not centralizing the training data, retain the privacy and locality of private data. The edge and cloud are candidate locations to deploy this AI agent, depending on whether on-site security operations are required or not.

3) **Heterogeneous Network (HetNet) Selection:** In an industrial environment comprising multiple RATs, the challenge of being always best-connected arises, directly affecting the design and performance of networked robot systems. RAT selection is traditionally solved by applying rules derived from the network infrastructure with prior domain knowledge and experience by experts. However, applying this type of RAT selection to robots is often complex to man-

age in dynamic and heterogeneous industrial environments. A HetNet Selection AI-agent that uses ML algorithms (e.g., RL, ANN and Fuzzy Logic) appears as a tool to mitigate the aforementioned challenges. It exploits the locally available radio context information to select the best RAT for each robot in the factory floor and, if required, the best handover candidate. The radio context information is defined by ETSI MEC, and provided by different radio information services, such as Radio Network Information Service (RNIS) and WLAN Access Information Service (WAIS) [145]. Based on such information, the HetNet Selection detects when e.g., an AVG will be out of coverage and lose the connection to the point of attachment. The robot algorithms that reside in the edge of the network can use this information to preemptively transfer state information to the new point of attachment and to instruct the AVG to change its RAT in order to seamless move within the factory floor. Since this AI agent depends on the locally available information, its preferable deployment location is the edge or fog.

From the aforementioned AI-agents, we select the Movement Prediction for studying its feasibility and performance in the continuation of this chapter. After reviewing the current State-of-the-Art, we analytically define the interference problem that current networked robots face in remote control systems. As a solution, we propose FoReCo, a forecast-based recovery mechanism for real-time remote control of robotic manipulators that is a prototype implementation of the Movement Prediction AI-agent. The proposed solution is evaluated in simulation and experimentation.

5.3. AI-assisted remote control

5.3.1. Related Work in remote control systems

There is a rising interest in the networking community for providing support to Industry 4.0 cases in commercial deployments. The goal is always to meet the reliability that industrial processes require as it is in the case of remotely controlled robots. Under the umbrella of the 5G-Public-Private-Partnership (5G-PPP) and the European Research Council (ERC), the European networking community has been focusing on how 5G and beyond 5G architectures can support Industry 4.0. 5G-DIVE [32] and 5Growth [33], are two examples of platforms that manage the adequate resource provisioning and allocation of services like remote control in Industry 4.0. More recent European research projects [146] as Daemon [147] and Hexa-X [148] also provide support to Industry 4.0 applications by bringing intelligence to the network in order to meet strict constraints such as reliability.

In the case of teleoperation applications (i.e., applications providing remote control of systems), there is a plethora of work in the robotic and mechatronic literature on how to operate robotic manipulators. Reference [149] presents a prototype designed for the remote operation of an industrial robot manipulator using augmented reality, and [150] studies how to overcome, with the help of predictions, the collision of a robotic manipulator with objects due to remote operator errors. Works as [151] and [152] propose teleoperated robotic manipulators that assess complex and high precision tasks with the help of ML assisted solutions, and gravity compensation approaches, respectively.

However, the robotic and mechatronic literature many times fails to consider the latency induced by the network in teleoperated/remotely-controlled systems. Indeed, none of the aforementioned works account for the network latency in the problem formulation, nor in the experimental stage, as authors control the robotic manipulator with a computer directly attached to the robot.

Works as [149–152] introduce errors in remotely-controlled robotic manipulators when applied in networks that suffer from high latencies or packet losses. Also, they cannot rely on network platforms like [32], [153], [147] and [148] to overcome such problems, as these platforms make a best-effort approach by means of network resource allocation and life-cycle management. That is, platforms such as 5Growth [153] make the best to allocate network and computing resources for applications as [152], but they do not assist the remote-control application to recover when control commands are delayed or lost in the network

Therefore, it is up to the remote-control application to decide how to react when control commands are delayed or lost. Mainly the robotic/mechatronic literature relies on control theory to overcome issues produced by delayed control commands. Works as [154–157] propose time domain passivity-based approaches, in particular [154] proposes to use a two-layer and a switching passivity-based [158] approach to deal with delays in the network. Other solutions [159–161] cope with the delay in remote-control using wave variable passivity-based approaches [162]. Another option is to resort to adaptive and robust control mechanisms to ensure the remotely-controlled robot stability upon delayed commands, as done in [163–166]. For example, [166] uses a Radial Basis Function Neural Network [167] based on Proportional Differential (PD) control [168] to mitigate the effect of external uncertainties and delayed commands in the robotic manipulator. However, the cited control-theory solutions in robotic/mechatronic literature takes unrealistic assumptions about the remote control commands' delays. In particular, [156] and [164] assume that the delay in the network is constant; [155], [157], [166], [159] and [160] assume that delays are constant or with small variations; and [154], [165], [161], and [163] take the causality assumption for the delay, i.e., the network delay cannot increase faster than time[23]. All of the aforementioned assumptions on network delay are not suitable for IEEE 802.11 wireless networks.

In this chapter, we aim to improve the application and communication reliability by solving the problem from the networking perspective. We use command predictions in order to recover from the loss or delay of control packets. There are also works in the state of the art that follows a similar approach, in particular, [169] presents a control communication protocol that takes into account the wireless Signal to Noise Ratio (SNR) and uses a reinforcement learning [170] approach [171] to find the optimal speed of an AVG; and [172] proposes an AVG path tracking application using a Kalman filter to provide delay estimations for successful operation. However, [169] assumes that the success of the wireless transmission is captured by a first-order Markov process [173], and [172] assumes that the command delay in an IEEE 802.11 network follows a Gamma distribution. Both assumptions neglect the presence of EM interference in the wireless channel, as well as the back-off and re-transmission mechanisms of IEEE 802.11 wireless channels that we investigate in this paper. Moreover, both [169]

[23] Following our notation, the causality assumption is expressed as: $|\Delta(c_{i+1}) - \Delta(c_i)| \leq |g(c_{i+1}) - g(c_i)|$

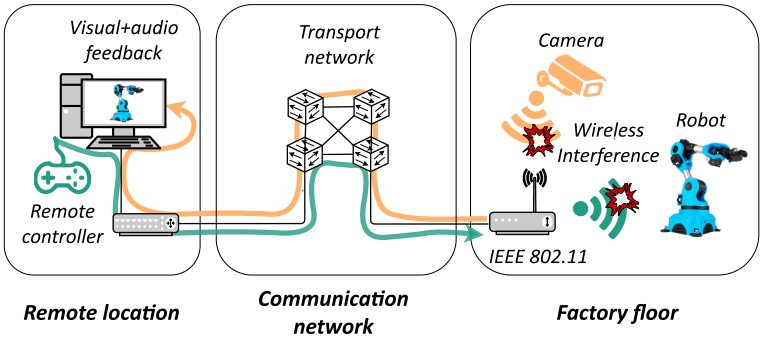


Figure 5.2. *Diagram of a industrial robotic remote control*

and [172] are solutions to enhance the reliability of real-time remotely controlled AVGs in wireless networks, rather than robotic manipulators.

Relation with the work in this chapter: The potential advantages from the wireless remote control of a robot manipulator are significant, but realizing such systems over an IEEE 802.11 network remains a challenging task. To the best of our knowledge, the state of the art disregards the presence of EM interference in real-time remote control and makes assumptions about the remote control commands' delay that do not apply to IEEE 802.11 wireless networks. To fill these gaps, we propose FoReCo, an ML-based solution that aims to minimize the robot trajectory error by predicting the missing remote control commands without making assumptions about the commands' delay.

5.3.2. Problem Statement

A remote control system in industrial environments generally consists of three main parts: *i)* remote location; *ii)* communication network; and *iii)* factory floor (see Figure 5.2). The remote site resides away from the factory floor, where a remote controller (fully autonomous or human-assisted) sends control commands to the factory robot in an open-loop fashion following a given frequency. Control commands which are sent every Ω [ms] have to transverse the transport network and wireless link to reach the robot. Additionally, the remote site contains visual and audio feedback that is streamed back to the remote location in order to provide similar conditions as those on the factory floor. While the control commands are very sensitive to packet losses and jitter, the visual and audio feedback can provide good quality of experience with up to 1 % of packet loss and 30 ms of jitter. Studying the visual and audio feedback over the inconsistent wireless channel and how it can impact the remote control of the robot is an interesting topic that is out of the scope of this work.

In this work considers an IEEE 802.11 wireless link, as its low price makes it an appealing solution for Industry 4.0. However, the unlicensed IEEE 802.11 spectrum leads to packet collisions, backoff times, and re-transmissions that introduce delay in the control commands. That is, since the moment a control command c_i is generated $g(c_i)$, up until the moment it is delivered to the robot $a(c_i)$, the transport network and wireless link introduce a delay $\Delta(c_i) = a(c_i) - g(c_i)$. Note that the latter is the addition of the delay introduced by the transport network $\Delta_T(c_i)$, and the delay introduced by the

wireless link $\Delta_W(c_i)$, i.e.: $\Delta(c_i) = \Delta_T(c_i) + \Delta_W(c_i)$. In this paper, we make the following assumption on the transport network delay:

Assumption 1. *The delay introduced by the transport network $\Delta_T(c_i)$ is upper bounded by a constant D:*

$$\Delta_T(c_i) \leq D, \quad \forall i \tag{5.1}$$

Since each network entity (e.g., switch or router) in the transport network has finite queue sizes, the transport network can be modeled as a Jackson network [174]. Thus, we choose D as a constant higher than the summation of waiting times and processing time at each queue within the remote control path.

However, even if D is very small, the delay introduced by the IEEE 802.11 link $\Delta_W(c_i)$ might lead to a laggy behavior in the remotely controlled robot. This means that the remote controller will experience a lag between the time it moves the controller, and the time the robot moves. Since remotely controlled robots can only tolerate waiting for τ milliseconds to receive the next command, if $\Delta(c_i) > \tau$ the command c_i will exceed the tolerated delay, and the robot will not execute it. Thus, it is necessary that the control commands delays satisfy $\Delta(c_i) \leq \tau$.

Control commands are sent every Ω ms and the robot expects to receive those commands in the same interval (Ω ms) for smooth operation. However, due to the network delay the next control command c_{i+1} might not arrive until $\Omega + \Delta(c_{i+1})$ ms have passed.

Overall, the robot will not execute commands that arrive out of time $\Delta(c_i) > \tau$, or are lost $\Delta(c_i) \to \infty$. In any of these situations, the command is not executed, resulting in a deviation from the ideal trajectory that the robot should follow (see Figure 5.3). Note that the remote controller will notice this error in the real trajectory via the visual feedback that it receives from the factory floor (see Figure 5.2). Thus, it is necessary to recover the discarded packets to minimize the error in the real trajectory.

Problem 1. *Given the random variables $\Delta(c_i)$, a distance $d : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, a tolerance τ, and a record of the last R commands; find $f : \mathbb{R}^d \times \underbrace{\ldots}_{R-2} \times \mathbb{R}^d \mapsto \mathbb{R}^d$ to solve*

$$\min_{\hat{c}_N} \quad \lim_{N\to\infty} \frac{1}{N} \sum_{i}^{N} d(\hat{c}_i, c_i) \tag{5.2}$$

$$s.t. \quad \hat{c}_i = f\left(\{\hat{c}_j\}_{i-R}^{i-1}\right) \mathbf{1}_{\Delta(c_i)>\tau} + c_i \left[1 - \mathbf{1}_{\Delta(c_i)>\tau}\right], \forall i \tag{5.3}$$

With $\mathbf{1}_{\Delta(c_i)>\tau} = 1$ if command c_i is delayed more than τ ms, and zero otherwise. Problem 1 targets to find a function f to derive those commands that did not arrive on time. Additionally, the derived commands $\hat{c}_i$ should minimize the error (i.e., the distance $d(\hat{c}_i, c_i)$) with respect to the original command c_i sent by the remote controller. That is, the objective of (5.2) is to minimize the dashed error region in Figure 5.3 by using forecasts $f\left(\{\hat{c}_j\}_{i-R}^{i-1}\right)$ when a command c_i does not arrive on time $\Delta(c_i) > \tau$ (see (5.3)).

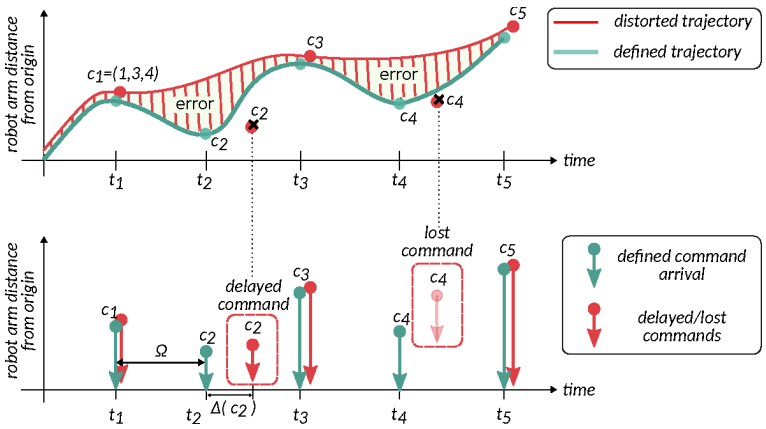


Figure 5.3. *Impact of delayed and lost commands in the robot trajectory.*

5.3.3. Forecast-assisted remote control (FoReCo)

In this section, we present FoReCo as a forecast-based recovery mechanism to minimize the trajectory error of remotely controlled robots via wireless connectivity.

The FoReCo Building Block

As discussed in Section 5.3.2, whenever the command delay exceeds the tolerance $\Delta(c_i) > \tau$, the robot considers the command c_i to be outdated and does not execute it. Depending on the robot, the absence of the command c_i may result in the robot stopping, or keep feeding the prior command c_{i-1} to the robot control loop, which is implemented with solutions as Proportional-Integral-Derivative (PID) controllers (see [175]). Either way, the command c_i will not be executed and the robot trajectory will deviate from the expected, i.e., the trajectory executed by the remote controller. It is at this point that FoReCo predicts the command c_i that has not arrived on time and transparently triggers its execution into the robot. Hence, FoReCo stands as a complementary solution for any remotely controlled robot using a wireless link, while being agnostic to the implemented robot controller (control theory-based or not).

To predict control commands out of time, FoReCo follows an ML based approach, which has been proven to be effective with intention prediction and estimation of future trajectories of objects, such as vehicles, bikes, and humans. The learning model consist of predicting incoming control commands $c_i, c_{i+1}, c_{i+2}, \ldots$ with the help of the prior $c_{i-1}, c_{i-2}, \ldots$ commands. To do so, we advocate for an ML based methodology due to *i)* the repetitive nature of the industrial tasks performed by remotely operated robots; and *ii)* the difficulty to solve this problem with traditional dynamic programming algorithms.

Figure 5.4 shows the conceptual components of the network control system we use to remotely control a robot (in-line with Figure 5.2). The system shows the details of the interactions between the remote site and the factory floor over a communication channel. First, a real-time video stream of the

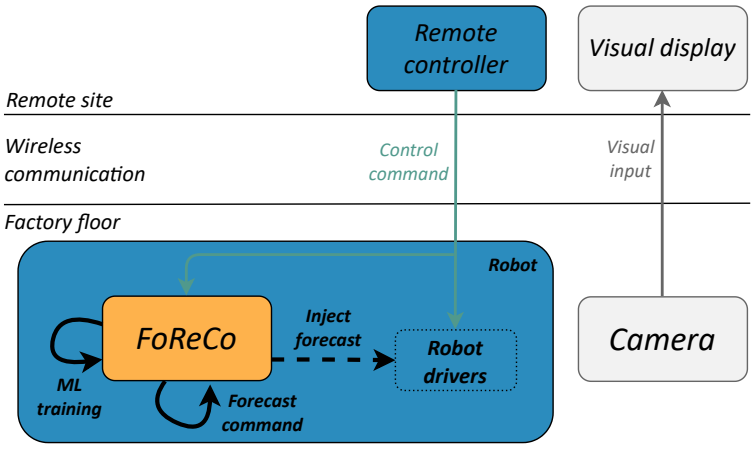


Figure 5.4. *FoReCo building block and remote control system.*

robot is presented to a visual display over a wired communication channel. For simplicity, we assume that the uplink channel is error and delay-free and the video input is delivered to the remote operator immediately. The remote controller, with the help of the visual input, sends control commands over the wireless communication channel, and the commands are received by both the robot and FoReCo. With the received commands, FoReCo performs two actions:

1) **ML training**: to solve Problem 1, FoReCo resorts to ML to derive $f\left(\{c_j\}, \vec{w}\right)$, with $\vec{w}$ being the weights to learn (see Section 5.3.3.2). To obtain $\vec{w}$, FoReCo creates a dataset with the commands it receives from the remote controller. The dataset contains a history of H commands, and FoReCo uses αH of them for training, and βH for testing; with $\alpha + \beta = 1$. As in Problem 1, the training procedure aims to minimize the distance between predicted commands $\hat{c}_i$, and the ones sent by the remote operator c_i. Hence, FoReCo trains its ML solution $f(\{c_j\}, \vec{w})$ s.t.:

$$\min_{\vec{w}} \frac{1}{\alpha H} \sum_{i}^{\alpha H} d\left(c_i, f\left(\{c_j\}_{i-R}^{i-1}, \vec{w}\right)\right) \tag{5.4}$$

   With the obtained weights $\vec{w}$, FoReCo tests the ML predictions accuracy in the testing set βH.

2) **Command forecast, validation and injection**: FoReCo awaits a control command c_i each Ω ms, and it triggers the forecasting if the next command c_{i+1} arrives latter than $a(c_i) + \Omega + \tau$. In this case, FoReCo will forecast the next command as $\hat{c}_{i+1} = f\left(\{\hat{c}_j\}_{i-R}^{i}, \vec{w}\right)$, i.e., using the ML solution f and the weights $\vec{w}$ obtained from the training stage. Next, FoReCo will validate the forecast by checking if the forcasted command offset is withing the acceptable boundaries with respect to the current position of the robot. This validation is performed by FoReCo in order to prevent forecasts that can lead to accident, malfunction, or robot misuse. The valid forecast command $\hat{c}_{i+1}$ is then injected in the robot drivers (as illustrated in Figure 5.4) with the latter assuming that it received a command on time. In the case a command arrives on time $a(c_{i+1}) \leq a(c_i) + \Omega + \tau$, FoReCo will just store the command in the dataset and later use it for training and forecasting purposes. Note that we refer to $\hat{c}_i = c_i$ if the command arrived on time $\Delta(c_i) \leq \tau$, so it satisfies

the constraint stated in (5.3). Thus, the forecasting receives as input $\{\hat{c}_j\}_{j-R}^{i}$ commands that arrived on time, and the forecasts of previous commands that did not arrive on time.

Studied ML Forecasting Algorithms

In the following, the selected ML algorithms used to implement FoReCo are described, as well as how they perform the command forecasting.

FoReCo is designed to forecast commands of remotely controlled robotic arms as the one in Figure 5.2. Each command consists of d joints (remember $c_i \in \mathbb{R}^d$) that move together to shift the arm manipulator position, so the latter reaches the object of interest. The rotation angle or shift of a joint, depending on the joint nature is represented as command coordinate $c_i = (c_i^1, \ldots, c_i^d)$.

The multivariate nature of the robotic arm suggests the use of vector autoregressive techniques as VAR. Appendix details how the Box-Jenkins methodology points out that VAR is the most adequate model to infer future commands $\hat{c}_{i+n}$ of the robotic arm. On top of VAR, we have also investigated a seq2seq model to check if it can outperform the well-established Box-Jenkins methodology. Both solutions are compared against a moving average that serves as a benchmark for the command inference accuracy.

We acknowledge that not all ML forecasting algorithms are covered in this section. However, in this paper we focus in the following ML forecasting algorithms, for they were accurate enough to outline the benefits of FoReCo later in the results Section 5.3.5:

- **Vector Autoregression (VAR)**: this regression solution is designed to predict multi-dimensional time-series with correlation across dimensions. This is the case of robotic arms, whose joint coordinates typically present correlation $\forall i, \exists k, m : c_i^k \sim c_i^{k+m}$; as they have to move together to reach and grab an object. VAR derives the prediction of a command as follows:

$$\hat{c}_{i+1}^k = f^k\left(\{\hat{c}_j\}_{i-R}^{i}, \vec{w}\right) = b^k + \sum_{l=1}^{d} \sum_{j=i-R}^{i} w_{i,j}^l \cdot \hat{c}_j^l, \quad k \leq d \tag{5.5}$$

  with b^k being the bias for the k^{th} coordinate, $w_{i,j}^l$ the regression weights, and $f^k(\cdot)$ denoting the k^{th} coordinate of the resulting prediction. Both b^k, $w_{i,j}^l$ elements lie within the weight vector $\vec{w}$.

- **Sequence to sequence (seq2seq)** [176]: this ML solution is based on a Neural Network (NN) that receives as input a sequence and produces an output, that in our case is just a single output. These seq2seq models are known as many-to-one, as we feed it with a sequence of past commands $\{\hat{c}_j\}_{i-R}^{i}$ to produce a single one $\hat{c}_i$. The seq2seq architecture we use has: *i)* an encoder layer of 200 Long Short-Term Memory (LSTM) neurons with Rectifier Linear Unit (ReLU) activations; *ii)* and a decoder layer of 30 LSTM neurons, also with ReLU activations. The motivation behind the use of a seq2seq solution is to learn and encode the characteristics of robot movements, so the decoder layer "interprets" the encoded characteristics and guess the next

command $\hat{c}_{i+1}$. Analytically, the seq2seq encoder and decoder layers are represented as follows:

$$e_i^k = \phi^k (W_0 \hat{c}_i + W_1 a_{i-1} + W_2 m_i), \quad k \leq d \tag{5.6}$$

$$\hat{c}_{i+1}^k = f^k \left(\{\hat{c}_j\}_{i-R}^i, \vec{w} \right) = \\ = \phi^k \left(W_4 e_i^k + W_5 a'_{i-1} + W_6 m'_i \right), \quad k \leq d \tag{5.7}$$

with $\phi(x)$ denoting the ReLU function[24], W_i denoting weight matrices (whose values are unrolled to derive the weight vector $\vec{w}$), a_{i-1}, a'_{i-1} being the output of the LSTM activation units, and m_i, m'_i referring to the memory cells of the encoder and decoder; respectively.

- **Moving Average (MA)**: we resort to this algorithm to have a benchmark for VAR and seq2seq. The MA derives the command prediction using:

$$\hat{c}_{i+1} = f^k \left(\{\hat{c}_j\}_{i-R}^i, \vec{w} \right) = \frac{1}{R} \sum_{j=i-R}^{i} \hat{c}_j \tag{5.8}$$

Note that FoReCo is flexible to support other forecasting algorithms, which can be integrated in a modular fashion.

Training of the Selected ML Forecasting Algorithm

Next, we detail how FoReCo trains the selected ML algorithms described above.

- **VAR training**: to train the VAR algorithm, we resort to Ordinary Least Squares (OLS), i.e., the weights are computed as follows:

$$\vec{w} = \underset{\vec{w}}{\operatorname{argmin}} \sum_{i}^{\alpha H} \sum_{k}^{d} \left(c_i^k - f^k(\{c_j\}_{i-R}^{i-1}, \vec{w}) \right)^2 \tag{5.9}$$

over the training portion of the dataset αH. Here $f^k(\cdot)$ refers to (5.5). Note that minimizing the summation in (5.9) is equivalent to minimizing the expression in (5.4), taking as distance $d(c_i, \hat{c}_i) = \sum_k \left(c_i^k - \hat{c}_i^k \right)^2$.

- **seq2seq training**: we train the seq2seq solution using Adam, a stochastic optimization method invariant to small gradients, as it is the case of our experimental study, e.g., a the 3rd robot joint takes values like $c_i^3 = 0.001$. We use Adam to iteratively minimize the error of the forecasts within a batch B_i of commands from the training set, i.e., $B_i < \alpha H$. Hence, at each training step Adam minimizes:

$$\min_{\vec{w}_t} l \left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t \right) = \sum_{i}^{\alpha H} \sum_{k}^{d} \frac{\left(c_i^k - f^k(\{c_j\}_{i-R}^{i-1}, \vec{w}_t) \right)^2}{B_i} \tag{5.10}$$

[24] $\phi(x) = 0, x \leq 0$ and $\phi(x) = x$ otherwise

with $l(\cdot)$ denoting the loss function, and $\vec{w}_t$ the updated weight vector at the step t of the training stage. Note that minimizing (5.10) is equivalent to minimizing (5.4) over the batch B_i, rather than the whole training dataset αH, taking the sum of squared distances. At each step the weights are updated as follows:

$$\vec{w}_{t+1} = \vec{w}_t - \eta \frac{m_{t+1}}{1-\beta_1^{\alpha H}} \frac{1}{\sqrt{\frac{v_i}{1-\beta_2^{\alpha H}}} + \varepsilon} \tag{5.11}$$

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) \nabla_{\vec{w}} l\left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t\right) \tag{5.12}$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2) \left[\nabla_{\vec{w}} l\left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t\right)\right]^2 \tag{5.13}$$

with m_t, v_t being the estimates of the first and second moment of the loss function gradient $\nabla_{\vec{w}} l(\cdot)$, η the step size, and $\beta_1, \beta_2, \varepsilon$ other hyper-parameters.

5.3.4. IEEE 802.11 with Electromagnetic Interference

So far we have discussed how FoReCo works in Section 5.3.3.1, and the ML solutions that we consider to assess the command forecasting in Section 5.3.3.2 and §5.3.3.3, respectively. In this section we explain the analytical model we consider to derive the delay that control commands experiment $\Delta_W(c_i)$ in IEEE 802.11 wireless links under EM interference. The analytical model is latter used in Section 5.3.5.1 to derive the $\Delta_W(c_i)$, and assess the performance of FoReCo in a simulated scenario as close as possible to real IEEE 802.11-based real-time remote control.

In this paper, we resort to the analytical model presented in [140] to derive wireless delays. This work models the MAC layer of IEEE 802.11 with CSMA/CA, and studies how neighboring nodes and non-IEEE 802.11 interfering sources impact the wireless delay. The work is based on a refinement [177] of Bianchi's characterization of IEEE 802.11 [178]. The particularity is that [48] extends the underlying Markov chain to also capture the presence of an interference source that is active during T_{if} transmission slots, and emits with a probability p_{if}. The proposed model also captures both the back-off mechanisms and re-transmissions (RTX) of frames upon collision in the IEEE 802.11 wireless links.

With the aforementioned model, [140] obtains the steady-state vector of each state, in particular, they derive the probability that a frame has to be transmitted after j unsuccessful re-transmissions, which is denoted as a_j. Moreover, [140] also derives $\mathbb{E}_j[\Delta_W(c_i)]$, that is, average delay that the command c_i experiences in the wireless transmission after j unsuccessful re-transmissions. Based on such expression, we derive in the Appendix some theoretical results around the analytical model given in [140], that give some insights about the delay of control commands. In particular, the theoretical results in the Appendix conclude that in the considered IEEE 802.11 scenario:

1) $\Delta(c_i)$ is only bounded on average, but not always (see Lemma 1);

2) $\Delta(c_i)$ diverges (see Corollary 1); and

3) the causality assumption does not apply (see Corollary 2).

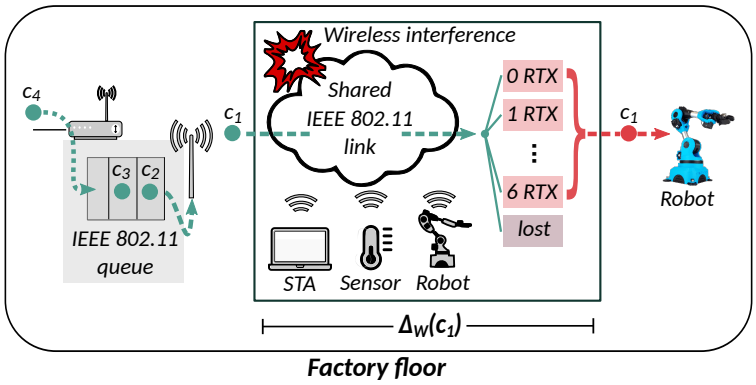


Figure 5.5. *Impact of wireless interference, retransmissions (RTX), and factory devices in the delay $\Delta_W(c_i)$ that control commands experience in an IEEE 802.11 link.*

Hence, the delay assumptions taken in the solutions presented in Section 5.4 do not hold. In other words, we cannot bound the delays that the remote control commands c_i are experiencing. Still, we resort to the analytical model presented in [140], as such unbounded delay behaviors are actually realistic in IEEE 802.11 scenarios upon the presence of interference sources.

To derive the value of $\Delta_W(c_i)$ we follow [140] and model the transmission of control commands c_i over IEEE 802.11 wireless links as a queuing model. From the problem statement formulation presented in Section 5.3.2, we know that control commands have an arrival rate $\frac{1}{\Omega}$. These commands are queued in the IEEE 802.11 access point before they are transmitted to the shared wireless link. Following the IEEE 802.11 standard, a frame is re-transmitted up to 6 times (7 total transmissions). After this threshold is exceeded, the frame and therefore the control command is assumed to be lost and no further re-transmission is executed (see Figure 5.5).

Depending on the number of RTX, the control command delay $\Delta_W(c_i)$ will be higher or lower. This system behaves as an $G/HEXP/1/Q$ queuing model, with Q being the length of the access point queue, and the service rates of the hyper exponential distribution corresponding to the average delay that control commands see after j RTX, i.e., $\frac{1}{\mathbb{E}_j[\Delta_W(c_i)]}$.

Given this $G/HEXP/1/Q$ queuing model, we can derive $\Delta_W(c_i)$ in the desired IEEE 802.11 wireless scenario accounting for the number of transmitting devices and the probability and time that the wireless interference is active. These are the delay values used in the simulation scenarios in Section 5.3.5.3, and we derive them using the CIW discrete event simulation library [179].

5.3.5. Results

In order to evaluate FoReCo, we consider a realistic industrial application where a robot manipulator is remotely controlled to perform a pick and place task. This remote control application allows us to select the most suitable forecasting algorithm for FoReCo, and to evaluate a prototype implementation of FoReCo under simulation and experimental scenarios. It is worth mentioning that in this

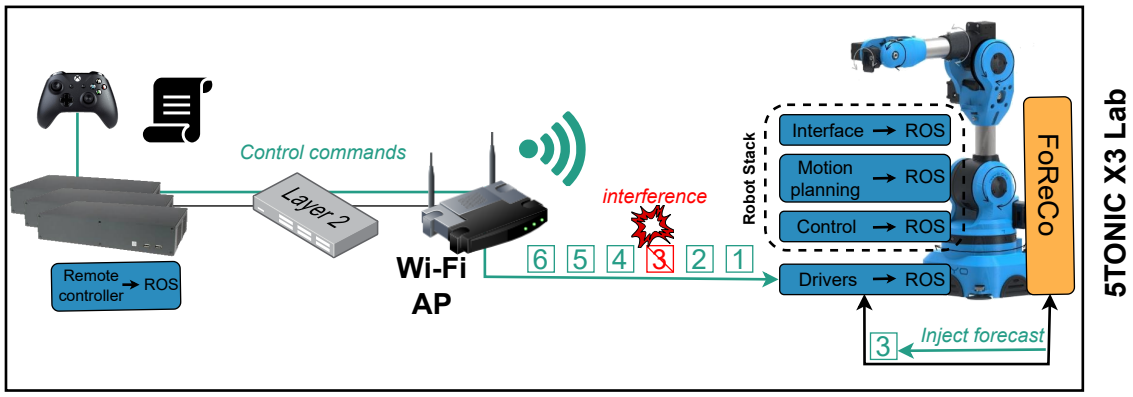


Figure 5.6. *Testbed setup.*

section we compare the performance of FoReCo with the baseline Niryo controller mainly because of the lack of state-of-the-art remote controllers for robot manipulators. We leave the comparison to more recent AVG approaches, e.g., learning-based AVG controllers [169], for follow-up work, where the AVG controllers need to be adapted for robot manipulators in order to employ their benefits.

Testbed setup and dataset collection

Figure 5.6 shows the part the 5TONIC experimental testbed (for more details see Section 2.3.2) that was re-used for the experimental analyses of FoReCo. The testbed is composed of: a 6-axis Niryo One robotic manipulator, a 2.4 GHz IEEE 802.11 AP, an L2 switch, a 2.4 GHz Silvercrest Wireless Transmitter that is used as a Wi-Fi jammer, and a joystick that is connected to a fog node. The communication between the robot and the joystick is comprised of a wireless link from the robot to the AP and an Ethernet link from the AP to the fog node using the L2 switch. The complete robot stack VNFs are deployed in the robot arm while the remote controller is deployed in the fog node. The robot stack expects to receive control commands each Ω ms and considers that a packet did not arrive on time otherwise, i.e., the tolerance is $\tau = 0$. robot stack uses the prior command $\hat{c}_{i+1} = c_i$ in case $\Delta(c_{i+1}) > \Omega$. Every received command is passed to the motion planning VNF (MoveIt), which uses Proportional–Integral–Derivative (PID) control. All the details about the hardware components of the testbed and remote control application implementation can be found in Section 2.3.2.

Figure 5.7 shows part of the dataset created by performing pick and place actions. The pick and place actions were manually repeated 100 times by two different human operators, an experienced and inexperienced human operator resulting in the creation of two separate datasets. To do so, they used the joystick as a remote controller, issuing a new control command every 20 ms. The inexperienced/experienced operators' datasets' contain $H = 187109$ commands. Both datasets store the joint states c_i of the robot manipulator under ideal network conditions, i.e., low latencies and absence of packet collision. To achieve such conditions, the datasets were obtained using Ethernet to send the remote controller commands. The experienced dataset was used to train the ML models while the inexperienced data was used for remote control and testing. In this way, we ensure that the trained

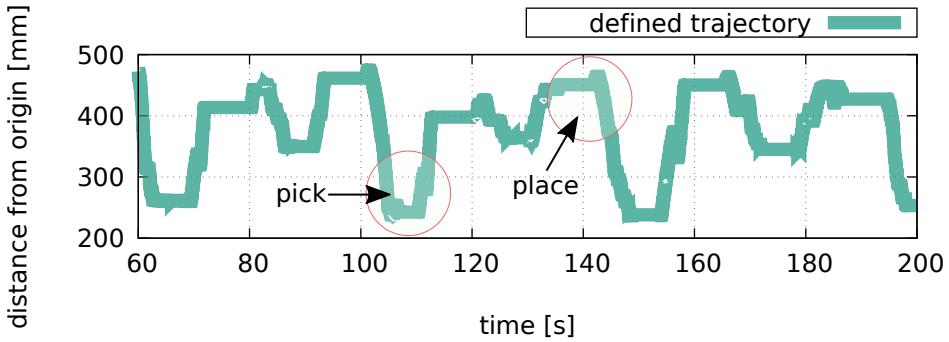


Figure 5.7. *Robot trajectory dataset with pick and actions of an inexperienced operator.*

ML model operates on data that is tightly related but not exactly the same as the training data.

Forecasting accuracy

We now evaluate which of the selected forecasting algorithms achieves the highest forecasting accuracy in the collected datasets. The VAR algorithm was implemented using statsmodel v0.12.1, and seq2seq using Tensorflow 2.1.0. Figure 5.8 shows the RMSE accuracy of each algorithm as we increase the forecasting window, i.e., how many consecutive commands are forecasted (a command is sent each $\Omega = 20$ ms). For every algorithm, we considered a record of the last $R = 1, \ldots, 20$ commands, and Figure 5.8 plots the best-performing R parameter for each algorithm. For the training stage (see Section 5.3.3.3), we used the $\alpha = 80\%$ of the experienced human operator data for training, and a $\beta = 20\%$ of the inexperienced operator data for testing. It is worth mentioning that we performed the 80%-20% split of the datasets only for testing purposes mainly because of the repetitive nature of the movements where by using 20% of inexperienced operator data is sufficient to give us insides into what is the most accurate model. Later when we evaluate the models in simulation and experimentation we use 100% of the experienced human operator data for training and 100% of the inexperienced operator data for validation. For seq2seq we resort to the standard hyper-parameter selection: $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-07$.

Results show that VAR has slightly better accuracy than MA, while seq2seq has the worst performance. It was expected that VAR would outperform MA since it is designed for correlated time-series – like the 6-axis time-series of the Niryo One robotic arm. However, seq2seq yielded worse accuracy than MA due to the vast number of weights $|\vec{w}| = 163803$ to learn, thus, it did not converge to an optimal solution. Given the results in Figure 5.3.3.3, we use the MA and trained VAR solution as forecasting techniques in our simulation analysis.

Simulation evaluation

In the following, we evaluate how FoReCo behaves under a a simulated environment with wireless interference. We consider a transport network with negligible transport delay, i.e., $D \simeq 0$ ms in *As-*

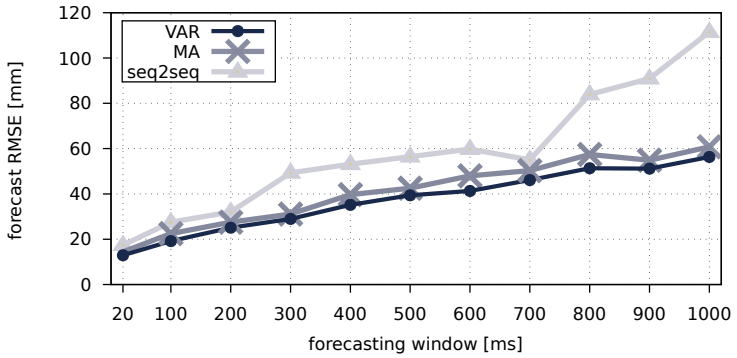


Figure 5.8. *Forecast accuracy for different forecasting windows.*

sumption 1, thus, commands' delays are dominated by the wireless delay $\Delta(c_i) \simeq \Delta_W(c_i)$. To derive $\Delta_W(c_i)$, we resort to an analytical model of IEEE 802.11 with non-IEEE interfering sources [140], and use the parameters reported in [140]. The goal of the simulation validation is two-folded: *i)* evaluate the precision of the forecasted commands by FoReCo, and *ii)* assess the scalability with up to 25 robotic arms sharing a wireless medium with interferences. All the details about the simulation implementation of FoReCo and the IEEE 802.11 analytical model can be found our publicly available git repository[25].

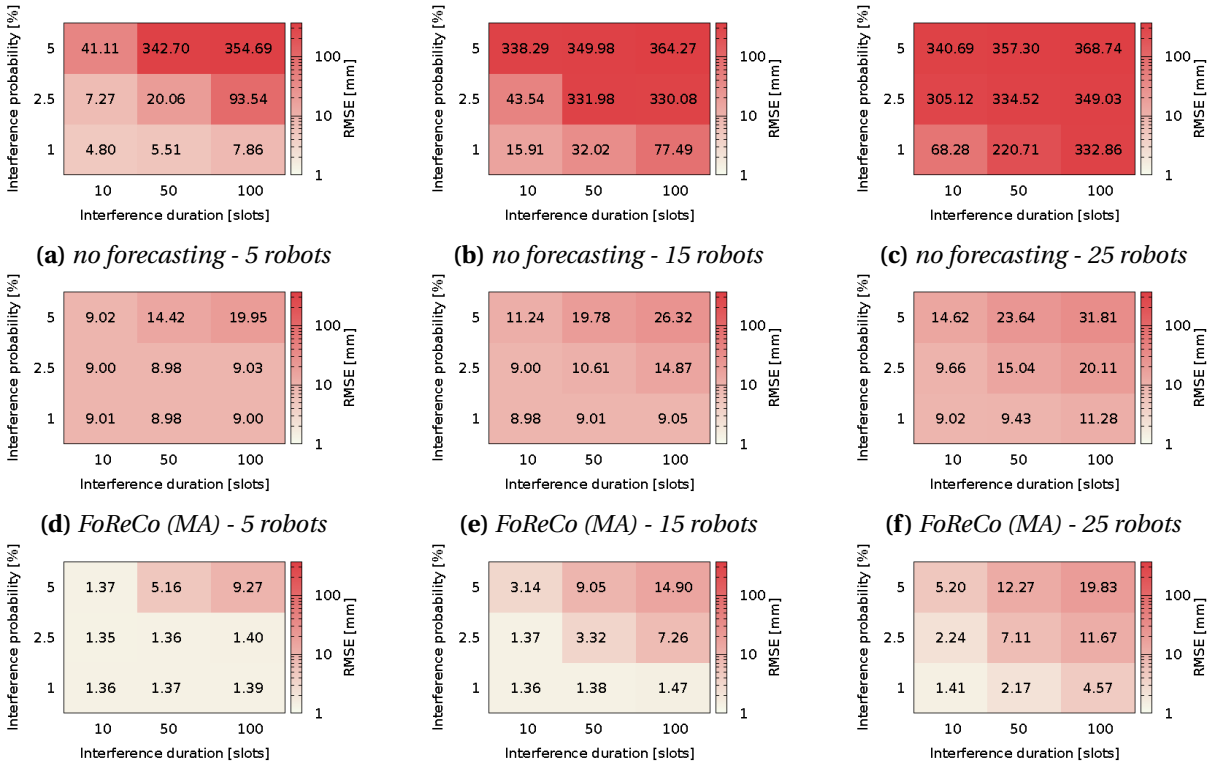
Simulation methodology

Each simulation issues the commands of an inexperienced human operator and introduces command delays $\Delta_W(c_i)$ following [140]. Figure 5.9 compares, in the upper, middle and lower rows, the error experienced by the robot trajectory when the state-of-the-art solution is used (i.e., repeat the prior command $\hat{c}_{i+1} = \hat{c}_i$ upon delays), when FoReCo recovers packets as specified in (5.8) and when FoReCo recovers the packets as specified in (5.5). Since the introduced wireless delay $\Delta_W(c_i)$ is a random variable, we repeat each simulation 40 times. Note that, in each simulation, we vary the time and probability of the active interference. Each square in the Figure 5.9 heatmap illustrates the averaged RMSE of the 40 simulations done for every pair of interference duration, and probability. The RMSE is computed over the entire robot trajectory induced by the inexperienced human operator, and it considers commands arriving on time $\Delta(c_i) \leq \tau$ and out of time $\Delta c_i > \tau$, without using control command forecasting (upper in Figure 5.9), and with FoReCo using MA and VAR (middle, and bottom rows in Figure 5.9; respectively).

Simulation results

The RMSE error in Figure 5.9 is represented in logarithmic scale, and we can appreciate that FoReCo command recovery constrained the robot trajectory error below 19.95 mm, 26.32 mm and 31.81 mm using MA (middle row) and 9.27 mm, 14.90 mm and 19.83 mm using VAR (bottom row) for 5, 15 and 25 robots on the factory floor, respectively. Figure 5.9 shows that the VAR solution outperforms the MA solution in every simulation scenario for approximately 10 mm. On the other hand, the no forecast-

[25] https://gitlab.it.uc3m.es/5g-team/FoReCo [Accessed: Jul. 14, 2022]



(a) *no forecasting - 5 robots* **(b)** *no forecasting - 15 robots* **(c)** *no forecasting - 25 robots*

(d) *FoReCo (MA) - 5 robots* **(e)** *FoReCo (MA) - 15 robots* **(f)** *FoReCo (MA) - 25 robots*

(g) *FoReCo (VAR) - 5 robots* **(h)** *FoReCo (VAR) - 15 robots* **(i)** *FoReCo (VAR) - 25 robots*

Figure 5.9. *Robot trajectory error upon interference without forecasting (top); with FoReCo using MA (middle); and FoReCo using VAR (bottom).*

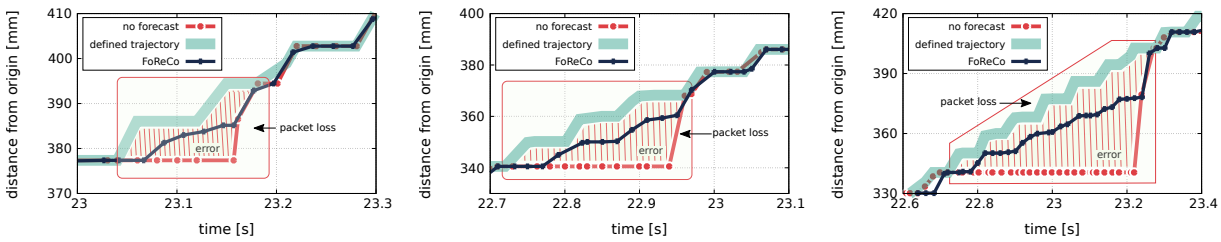
ing solution resulted in an RMSE in the order of ~ 350 mm in the worst cases, no matter the number of robots. Thus, simulations indicate that *i)* The VAR solution outperforms the MA solution by minimizing the error for additional 10 mm; *ii)* FoReCo based on VAR will not exceed errors of 20 mm, and *iii)* FoReCo reduces the experienced error by more than one order of magnitude. In particular, FoReCo using VAR reduces by more than 94.4% (368.74 mm with no forecasting and 19.83 mm with FoReCo (VAR)) the experienced error in factory floors of 25 robots

Experimental evaluation

Motivated by the simulation results, we implemented and integrated a prototype of FoReCo that is based on VAR within the real remote control system presented in Figure 5.6. The prototype is following the principles defined in Section 5.3.3 and uses ROS to interact with the remotely controlled Niryo One robotic arm. All the details about the prototype implementation of FoReCo can be found in our publicly available git repository. For what concerns the details about the remote control application please refer to Section 2.3.2.

Experimental methodology

We performed two separate experimental analysis. In our first experimental analysis, we manually



(a) *5 consecutive losses* **(b)** *10 consecutive losses* **(c)** *25 consecutive losses*

Figure 5.10. *Robot trajectory with controlled packet losses using no forecasts, and FoReCo.*

introduced loss of control commands in order to evaluate the improvements that the prototype offers under a controlled environment. The complete dataset from the inexperienced user was used to develop a remote controller that was randomly dropping consecutive control commands. Every time consecutive control commands were lost, FoReCo injected predictions from the VAR model. We executed 3 different sets of experiments, where the remote controller randomly introduced 5, 10, or 25 consecutive losses. Each experiment run was 30 seconds, with the robot joint states being recorded in the robot itself.

In our second experimental analysis, we emulate a realistic interference scenario where the Silvercrest Wireless device is used to transmit synchronized radio waves in the same frequency as the robot, introducing unpredictable network delays $\Delta(c_i)$ and packet losses – same as the analytical model [140] used in prior simulations in Section 5.3.5.3. The dataset from the inexperienced user was used in order to develop a remote controller that sent the control commands over the jammed wireless network. Every time control command(s) were lost or delayed due to the interference $\Delta(c_i) > \tau$, FoReCo injected predictions from the VAR model. The experiment was executed for 30 seconds and in the robot, we recorded the robot joint states.

Experimental results

Figure 5.10 shows the trajectories followed by the robot arm when consecutive control commands were lost for the case of no forecasts, and the FoReCo solution. The results show that FoReCo can mitigate the negative effects of lost commands by reducing the trajectory error in the 3 different sets of experiments for approximately 50%. Such results are in line with the simulation scenario where 5 robots share the IEEE 802.11 channel (see Figure 5.10(d)) and the trajectory error is reduced by approximately 71.42% (4.80 mm with no forecasting and 1.36 mm with FoReCo (VAR)). Moreover, results show that the RMSE for all three experimental sets is between 1.35 mm and 9.27 mm, which makes it consistent with the simulation results obtained for 5 stations. It is worth mentioning that as we increase the number of robots (e.g., 15 or 25) that share the wireless medium in simulation, the probability, and duration of the interference increases and with that FoReCo can reduce the trajectory error up to 94.4. It is worth mentioning that the experimental and simulation results can only directly be compared when involving the same number of robots. Figure 5.10(c) shows how FoReCo deviates more and more from the defined trajectory as the number of consecutive losses increase (in between second 23 and 23.2), since VAR builds its forecasts $\hat{c}_{i+1}$ using prior forecasted commands

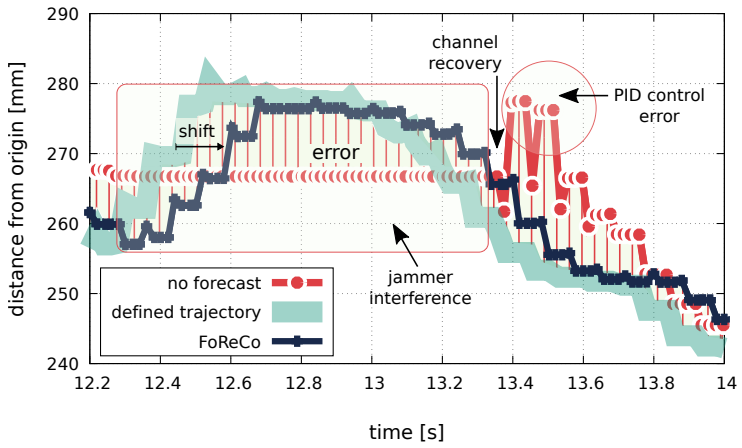


Figure 5.11. *Robot trajectory upon IEEE 802.11 jammer interference*

$\hat{c}_i, \hat{c}_{i-1}, \ldots, \hat{c}_{i-R}$. Thus, the prediction error propagates – see (5.5).

Figure 5.11 presents the trajectories followed by the robot arm when the wireless channel was interfered by the jammer. The results show how FoReCo reduces by more than 50% the trajectory error RMSE (from 18.91 mm to 8.72 mm) – compared to the bare Nyrio One solution without forecasting. In the interval between 12.4 and 12.6 seconds, FoReCo starts to deviate (shift) from the defined trajectory in an identical way to the controlled experiments described above. In addition, an interesting observation is how the robot trajectory behaves upon channel recovery. In the latter case, the Niryo One ROS MoveIt PID controller takes around 400 ms to stabilize the trajectory (from second 13.5 to 13.8), since the PID controller receives repeated commands $c_{i+1} = c_i, \forall i \in [12.4, 13.4]$ during more than a second, resulting in the highlighted error in Figure 5.11.

Since Niryo One is equipped with a Raspberry Pi3, which is limited in terms of computing power, we executed a set of experiments to measure both training and inference times of VAR whenever executed in the robot itself. While training takes around 5.99±0.06 min to train using the experienced human operator dataset, the inference (i.e., prediction of the commands) only takes 1.60±0.16 ms. These results show that the current hardware of the robot is sufficient to not only accommodate FoReCo within the constraints of our prototype but also to support the implementation of more stringent applications control loops. Note that training is only required when we need to build or update the model, which time profiling is presented in Table 5.2.

Table 5.2. *Time profiling of FoReCo training in Niryo One*

	Load Data (s)	**Down Sampling (s)**	**Check Quality (s)**	**Training Model (s)**
Raspberry Pi3 (Robot)	1.95 ± 0.02	0.26 ± 0.007	306.38 ± 3.15	50.98 ± 0.54

For comparison, Table 5.3 also presents training and inference times in different equipment: *i)* NVIDIA Jetson Nano which can be co-located with Niryo One; *ii)* a laptop equipped with a 2nd gen

Intel Core i7 and 6GB RAM representing the user equipment; and *iii)* a local server with two Intel(R) Xeon(R) CPU E5-2620 v4@2.10GHz and 64GB RAM representing the edge to offload training and inference tasks.

Table 5.3. *Training and inference times in different equipment*

	Training (min)	**Inference (ms)**
Raspberry Pi3 (Robot)	5.99 ± 0.06	1.60 ± 0.16
NVIDIA Jetson Nano (Robot)	1.31 ± 0.01	0.61 ± 0.28
Laptop (UE)	0.36 ± 0.01	0.22 ± 0.10
Local Server (Edge)	0.23 ± 0.007	0.0001 ± 0.00003

5.3.6. Discussion

In this section we discuss the results, analyzing how they can be interpreted from the perspective of real-time wireless networked control systems. In addition, possible future directions are identified.

Overall performance and applicability

In experimental scenarios (see Section 5.3.5.4) with a single robot, FoReCo reduces by a 34.35% the trajectory error upon the presence of up to 25 slots of interference. Moreover, the simulation results Section 5.3.5.3 show that FoReCo reduces the trajectory error up to a 94.4% with 25 robots sharing the IEEE 802.11 channel, and up to 100 consecutive interference slots. That is, FoReCo's benefits are more evident in worse wireless conditions, either due to simultaneous robots transmitting in the wireless medium or due to longer interference periods. The reason is that packet collision and delayed/lost commands increase with the number of robots in the channel, and the robotic arm remains still for long periods as the consecutive losses increase. Thus, FoReCo recovers more delayed/lost commands and prevents the robot from stopping.

Results show that FoReCo can achieve high precision and suggest that industrial manipulation applications (e.g., assembly, pick-and-place) can benefit from adopting the proposed method. Results also demonstrate that the prototype of FoReCo can be easily attached to robot manipulators without robot-specific modifications. Furthermore, results show how predictive control can improve the reliability of real-time remote control over the IEEE 802.11 wireless networks.

Coexistence with emerging wireless technologies

Existing industrial wireless technologies cannot meet all of the remote control requirements (such as 99.999% of reliability, 2-20ms latency, 100-200 Mbps data rate). Although emerging technologies such

as Wi-Fi 6E and 5G are expected to be a key enabler in this regard by increasing the levels of reliability in industrial wireless, the uncertain and time-varying wireless channel continues to be an issue for wireless technologies. The fact that FoReCo does not make any assumptions about the wireless channel and the network delays makes it applicable also to emerging wireless such as millimeter Wave or 5G. In addition, it is very likely that legacy systems will leverage previous technologies which will operate for years to come, and FoReCo can offer them the needed reliability for running remote control applications.

Real-time Path Tracking Predictions

Concerning the ML algorithm for repetitive reference trajectories, given the performance of VAR, our future work will consider other forecasting algorithms as exponential smoothing methods. We will also investigate other ML approaches, such as classification algorithms, to infer the type of command that comes next. It is worth noting that, although VAR performed well on the pick-and-place dataset, a deviation from the defined trajectory is witnessed when the number of consecutive commands losses increases. Future versions of FoReCo may mitigate large periods of consecutive losses by incorporating delayed commands that did not arrive on time, i.e., commands c_i with $\Delta(c_i) > \tau$ could be used instead of the predicted one $\hat{c}_i$, to infer commands after $n\Omega$ ms. In other words, based on (5.3) we could use $\hat{c}_n = f(\hat{c}_{n-1}, \ldots, c_i, \ldots)$.

Predictions at the edge of the network

In addition to improving the model accuracy and precision, an edge-based version of FoReCo can be considered where the VAR algorithm will always base its predictions on the real control commands $\hat{c}_i = f(\{c_j\}_{i-R}^{i-1})$. However, this approach is a bit more disruptive because an edge-based version of FoReCo indicates that the predictions will need to traverse the interfered wireless channel. Hence, FoReCo will need to piggyback the predictions together with real-time control commands. Piggybacking predictions require modifications of the robot drivers to use them whenever a control command does not arrive on time. We leave such an option for future work.

An extra resilience layer

As stated in Section 5.4, FoReCo targets a solution from the network perspective, in opposition to already existing related work. However, the two approaches are not mutually exclusive and, consequently, they can be used together to provide even higher levels of reliability and precision when performing remote control of robotic manipulators. In doing so, in a full-fledged solution, several recovery mechanisms can be envisioned as a resilience stack where all layers work together in a fail-over fashion.

Extension to other robotic systems

Although FoReCo explicitly targeted robotic manipulators, the solution described in this work can be easily extended to many other robotic systems that implement open- or close-loop between themselves and the remote controlling system. For example, remotely controlled AVGs within the manufacturing plant. Similar to the robotic manipulators, missing commands are predicted and injected into the AVG so that it can smoothly follow its trajectory. Still, FoReCo requires that performed tasks are somehow periodic so that the corresponding dataset can be created, and training and inference tasks executed with acceptable levels of precision.

5.4. Conclusions

This chapter discusses the role of AI in addressing some of the challenges in Industry 4.0, mainly related to networked robots. AI agents, with the help of ML algorithms, open the range of opportunities to enable optimizations in terms of reliability, robustness, and performance in the networked robots. This chapter starts by introducing the re-modeled concept for the networked robot, where cloud, edge, and fog computing are integrated with emerging networking technologies such as 5G and Wi-Fi 6E, and robots. It then identifies and analyzes exemplary AI agents for the networked robots, spanning from the application to the infrastructure level. FoReCo has been proposed to demonstrate the applicability of the Movement Prediction AI agent to predict the next movement(s) by using real data from a robotic arm. The FoReCo prototype uses VAR and ROS, and its performance has been assessed in a commercial research robotic arm remotely controlled over IEEE 802.11 wireless channels under the presence of interference. We also validate FoReCo through simulation and assess its performance in a real testbed. This chapter addressed **C7** and **C8** that were defined in the scope of this thesis by achieving the following results:

- We propose AI agent for networked robots, including possible ML algorithms to implement them. These agents will combine data from the robots, network, and computing infrastructure in order to optimize the robot operation.

- We develop FoReCo, a prototype of the Movement Prediction AI agent that was integrated into the 5TONIC testbed with a commercial research robotic arm.

- We achieved a trajectory error below 19.83 mm in simulation, and of 8.72 mm in experimentation.

- We reduced the trajectory error by more than 50% in both simulation and experimentation.

- We showed that FoReCo is lightweight enough to be deployed in the hardware already available in several solutions.

As follow-up work, we plan to *i)* integrate more ML forecasting algorithms; and *ii)* adapt solutions of wireless controlled AVGs [172][169] to remotely controlled robotic arms, and compare their performance against FoReCo in IEEE 802.11 wireless channels.

CHAPTER 6

Conclusions

This thesis aims to improve the evolution of robotic systems through the integration of modern ICTs - MEC, NFV, orchestration, federation, AI and 5G - into the existing and future robot systems. After providing an introduction of the enabling technologies, the thesis starts with **design** and **real-word prototype implementation** of an **end-to-end edge native robotic system**. The proposed system design distributes the robot functionalities at the edge of the network where location and radio network information services are available to optimize the performance of the robots. Additionally, the edge native robotic system envisions an orchestration system that is responsible for smart life-cycle management of the system. During this thesis we implemented and deployed two real-word testbeds with open-source software (e.g., ROS, Docker, LXD, Fog05, Hostapd, Python, and Linux) and research hardware (e.g., Turtlebot2 and Niryo One). The deployed testbeds are taken as starting point in this thesis to study the enhancements that the modern ICTs can offer to the robotic systems.

Leveraging on the proposed edge robotics system, this thesis experimentally evaluates in a Technology Readiness Level (TRL) 5-6 the enhancements that **edge computing** and **5G** can bring to robotic systems. In comparison with cloud computing, one of the key benefits that **edge computing** can offer to robotic systems is access to the local **context-information**. In this context, our research contributed by providing three exemplary context-aware algorithms for mobile robots and the results demonstrate: *i)* **smoother driving** of the mobile robot at high speeds by adapting the robot navigation to the Wi-Fi signal level, *ii)* **improved coordination** between mobile robots that use network-assisted D2D communication, and *iii)* **dynamic extension** of the wireless connectivity in indoor environment by migrating the access point functionality. In addition, to the context-aware enhancements, edge computing can offer **computation offloading** of mission-critical functionalities of the robot where

the experimental results showed that up to 16% of CPU and 34% of MEM savings can be achieved by virtualizing and offloading robot manipulator functions. For what concerns the combination of **5G** connectivity and **edge computing**, this thesis validates the low-latency KPI for remotely-controlled robots by achieving control loops of 20 ms and remote control synchronization of 97.66±0.67% while allowing the offloading of the complete robot stack in the edge.

Later, the thesis presents the feasibility of integrating innovative **orchestration solutions** in robotic systems. To do so, this thesis implements and integrates: **i) OKpi**, an NFV orchestration algorithm, and **ii) DLT-based federation** for multi-domain deployments in a real-world testbed that is composed of autonomously navigated mobile robot and diverse computational and network resources. The experimental results show how **OKpi** can meet the **latency KPI (15 ms)** of an autonomously controlled mobile robot by selecting the correct Point-of-Attachment. The target E2E latency of 15 ms was violated only by 7% of the time, while the SoA solution based on cloud robotics violated it for 74%. Regarding the **DLT-federation**, the results present a **federation time** of approximately 19 seconds for extending the wireless connectivity of a mobile robot in an external administrative domain.

Finally, the thesis focuses on the role of Artificial Intelligence in addressing some of the challenges in Industry 4.0, mainly related to networked robots. Having in mind the heterogeneity and diversity of the cloud-to-thing continuum, networked robots are facing a major challenge in their integration with existing network and computing infrastructure. In this context, our research contributed by proposing different exemplary **AI agents** based on **ML** algorithms spanning from **application** (movement prediction, task learning, risk reduction and predictive maintenance) to **infrastructure level** (dynamic scaling, intrusion detection and heterogeneously RAT selection). The last part of this thesis studies how to predict the next robot movement(s) of a robotic arm that is remotely controlled over IEEE 802.11 wireless channels under the presence of interference. To do so, this thesis studies and compare the accuracy of forecasting algorithms to predict the missing remote-control command using a real dataset of a human operator. The best forecasting algorithm is then implemented in both simulation and experimentation using a commercial research robot that performs pick-and-place tasks. Results show that upon interference FoReCo reduces the **trajectory error** by more than 34.35% in both simulation, and experimentation.

The advent of the new robotic era is yet to come and a set of powerful ICTs have the potential to lead the new generation of robotics services. The research of this thesis contributes towards the realization of such new robotic services by addressing some of the constrains faced in current cloud robots. The achieved results show how the **edge** of the network allows deployment of **intelligent**, **inexpensive**, **collaborative** and **context-aware** robots with low processing power and memory by leveraging on the elastic radio access and computation resources offered by the **edge** infrastructure. This opens a new set of innovative concepts such as **orchestration**, **federation** and **predictive control** in robotics that has the potential to lead towards future efficient, smart and automated industries.

References

[1] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26 754–26 773, 2017. DOI: 10.1109/ACCESS.2017.2773127.

[2] Boston Dynamics, *Spot*, [Online]. Available: https://www.bostondynamics.com/products/spot (accessed on Jul. 14, 2022).

[3] Boston Dynamics, *Atlas*, [Online]. Available: https://www.bostondynamics.com/atlas (accessed on Jul. 14, 2022).

[4] V. Kumar, D. Rus, and G. S. Sukhatme, "Networked robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 943–958. DOI: 10.1007/978-3-540-30301-5_42. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_42.

[5] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 1–1, Jan. 2016.

[6] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012. DOI: 10.1109/TASE.2012.2214676.

[7] myRobots, *myRobots*, [Online]. Available: https://myrobots.ca/ (accessed on Jul. 14, 2022).

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[9] F. Pires, A. Cachada, J. Barbosa, A. P. Moreira, and P. Leitão, "Digital twin in industry 4.0: Technologies, applications and challenges," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 721–726. DOI: 10.1109/INDIN41052.2019.8972134.

[10] ETSI, "Digital Enhanced Cordless Telecommunications (DECT); Study on URLLC use cases of vertical industries for DECT evolution and DECT-2020," European Telecommunications Standards Institute (ETSI), Technical Report (TR) 103 515 v1.1.1, Mar. 2018.

[11] 3GPP, "Service requirements for cyber-physical control applications in vertical domains," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.104 v17.4.0, Sep. 2020.

[12] 5G Alliance for Connected Industries and Automation (5G ACIA), *5G for Connected Industries and Automation, Second Edition*, Feb. 2019.

[13] NGMN, "5G E2E technology to support verticals URLLC requirements," Next Generation Mobile Networks (NGMN), Final Delivrable v2.5.4, Feb. 2020.

[14] ITU-R, "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," International Telecommunications Union Radiocommunication Sector, Report ITU-R M.2410-0, 2017, [Online]. Available: https://www.itu.int/pub/R-REP-M.2410-2017 (accessed on Jul. 14, 2022).

[15] ETSI, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 003 v2.1.1, Jan. 2019.

[16] 3GPP, "Study on Architecture for Next Generation System," 3rd Generation Partnership Project, TR 23.799, 2016.

[17] R. S. Montero and et al., "Extending the Cloud to the Network Edge," *Computer*, vol. 50, no. 4, pp. 91–95, 2017.

[18] ETSI, "Mobile Edge Computing (MEC); Radio Network Information API," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 012 v1.1.1, Jul. 2017.

[19] ETSI, "Multi-access Edge Computing (MEC); WLAN Information API," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 028 v2.0.1, Jul. 2018.

[20] ETSI, "Mobile Edge Computing (MEC); Location API," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 013 v1.1.1, Jul. 2017.

[21] ETSI, "MEC in 5G networks," European Telecommunications Standards Institute (ETSI), White Paper 28, Jun. 2018.

[22] I. I. C. W. Committees, *The industrial internet of things volume g1: Reference architecture*, https://www.iiconsortium.org/IIRA.htm, Accessed: 2022-4-20, 2019.

[23] ETSI, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Architectural Framework Specification," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 006 v2.1.1, Jan. 2021.

[24] M. Chiosi *et al.*, "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action," European Telecommunications Standards Institute (ETSI), Whitepaper, 2017.

[25] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint vnf placement and chaining," in *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, ser. CAN '16, Irvine, California, USA: Association for Computing Machinery, 2016, 19–24. DOI: 10.1145/3010079.3010083. [Online]. Available: https://doi.org/10.1145/3010079.3010083.

[26] F. Carpio, S. Dhahri, and A. Jukan, "Vnf placement with replication for loac balancing in nfv networks," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6. DOI: 10.1109/ICC.2017.7996515.

[27] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057036.

[28] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1346–1354. DOI: 10.1109/INFOCOM.2015.7218511.

[29] P. Zhao and G. Dán, "A benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1460–1472, 2018. DOI: 10.1109/TNSM.2018.2873178.

[30] F. Malandrino and C.-F. Chiasserini, "Getting the most out of your vnfs: Flexible assignment of service priorities in 5g," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–9. DOI: 10.1109/WoWMoM.2019.8792983.

[31] J. Martín Pérez, "Nfv orchestration in edge and fog scenarios," *https://doi. org/10.1007/s10922-020-09534-z*, 2021.

[32] C. Guimarães *et al.*, "DEEP: A Vertical-Oriented Intelligent and Automated Platform for the Edge and Fog," *IEEE Communications Magazine*, vol. 59, no. 6, pp. 66–72, 2021. DOI: 10.1109/MCOM.001.2000986.

[33] C. Guimarães *et al.*, "Public and non-public network integration for 5growth industry 4.0 use cases," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 108–114, 2021. DOI: 10.1109/MCOM.001.2000853.

[34] J. Baranda Hortiguela *et al.*, "Realizing the network service federation vision: Enabling automated multidomain orchestration of network services," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 48–57, 2020. DOI: 10.1109/MVT.2020.2979558.

[35] A. Francescon, G. Baggio, R. Fedrizzi, R. Ferrusy, I. G. Ben Yahiaz, and R. Riggio, "X–mano: Cross–domain management and orchestration of network services," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–5. DOI: 10.1109/NETSOFT.2017.8004223.

[36] GSMA, "Operator Platform Concept Whitepape," GSMA, White paper Version 1.0.3, Feb. 2020.

[37] A. Boubendir *et al.*, "Federation of cross-domain edge resources: A brokering architecture for network slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 415–423. DOI: 10.1109/NETSOFT.2018.8460114.

[38] X. Li et al., "Service orchestration and federation for verticals," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 260–265. DOI: 10.1109/WCNCW.2018.8369008.

[39] J. B. et al., "Composing services in 5g-transformer," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19, New York, NY, USA: ACM, 2019, 407–408. DOI: 10.1145/3323679.3326630.

[40] J. B. et al., "Nfv service federation: Enabling multi-provider ehealth emergency services," in *Proceedings of the International Conference on Computer Communications (INFOCOM'20)*, 2020.

[41] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in Springer Nature, Aug. 2017, pp. 85–113. DOI: 10.1007/978-3-319-38756-7_4.

[42] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017. DOI: 10.1109/ACCESS.2017.2756069.

[43] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5g cellular networks: Challenges, solutions, and future directions," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 86–92, 2014.

[44] EIA, "International Energy Outlook 2017," U.S. Energy Information Administration's (EIA), Report (R) IEO2017, Sep. 2017.

[45] A. Willig, K. Matheus, and A. Wolisz, "Wireless technology in industrial networks," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1130–1151, 2005. DOI: 10.1109/JPROC.2005.849717.

[46] J. Thomesse, "Fieldbus technology in industrial automation," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1073–1101, 2005. DOI: 10.1109/JPROC.2005.849724.

[47] J. Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005. DOI: 10.1109/JPROC.2005.849721.

[48] S. Vitturi, F. Tramarin, and L. Seno, "Industrial wireless networks: The significance of timeliness in communication systems," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 40–51, 2013. DOI: 10.1109/MIE.2013.2253837.

[49] M. Hermann, T. Pentek, and B. Otto, "Design Principles for Industrie 4.0 Scenarios," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 3928–3937. DOI: 10.1109/HICSS.2016.488.

[50] A. A. F. Saldivar, Y. Li, W. Chen, Z. Zhan, J. Zhang, and L. Y. Chen, "Industry 4.0 with cyber-physical integration: A design and manufacture perspective," in *2015 21st International Conference on Automation and Computing (ICAC)*, 2015, pp. 1–6. DOI: 10.1109/IConAC.2015.7313954.

[51] F. Tao and Q. Qi, "New it driven service-oriented smart manufacturing: Framework and characteristics," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, pp. 81–91, Jan. 2019. DOI: 10.1109/TSMC.2017.2723764.

[52] F. Tao, L. Zhang, Y. Liu, Y. Cheng, L. Wang, and X. Xu, "Manufacturing service management in cloud manufacturing: Overview and future research directions," *Journal of Manufacturing Science and Engineering*, vol. 137, Apr. 2015. DOI: 10.1115/1.4030510.

[53] D. Wu, M. J. Greer, D. W. Rosen, and D. Schaefer, "Cloud manufacturing: Strategic vision and state-of-the-art," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 564–579, 2013. DOI: https://doi.org/10.1016/j.jmsy.2013.04.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612513000411.

[54] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Design Automation Conference*, 2010, pp. 731–736. DOI: 10.1145/1837274.1837461.

[55] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, Feb. 2018. DOI: 10.1007/s00170-017-0233-1.

[56] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[57] A. M. Alba, J. H. G. Velásquez, and W. Kellerer, "An adaptive functional split in 5g networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 410–416. DOI: 10.1109/INFCOMW.2019.8845147.

[58] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, 2008, pp. 791–798.

[59] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, Jan. 2000.

[60] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, 2003, pp. 200–206.

[61] F. Fainelli, "The openwrt embedded development framework," in *Proceedings of the Free and Open Source Software Developers European Meeting*, sn, 2008, p. 106.

[62] J. Malinen, *Developers' documentation for wpa_supplicant/hostapd*, 0.7.x, 2009. [Online]. Available: http://w1.fi/wpa_supplicant/wpa_supplicant-devel.pdf.

[63] S. Bangolae, C. Bell, and E. Qi, "Performance study of fast bss transition using ieee 802.11r," in *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, New York, NY, USA: ACM, 2006, 737–742.

[64] P. K. Garimella, "It-ot integration challenges in utilities," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 199–204.

[65] F. Tao and Q. Qi, "New it driven service-oriented smart manufacturing: Framework and characteristics," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 81–91, 2019.

[66] A. Hahn, "Operational technology and information technology in industrial control systems," in *Cyber-security of SCADA and Other Industrial Control Systems*. Cham: Springer International Publishing, 2016, ch. Operational Technology and Information Technology in Industrial Control Systems, pp. 51–68.

[67] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[68] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[69] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, 2016.

[70] ETSI, "Mobile Edge Computing (MEC); Bandwidth Management API," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 015 v1.1.1, Oct. 2017.

[71] W. Dai, H. Nishi, V. Vyatkin, V. Huang, Y. Shi, and X. Guan, "Industrial edge computing: Enabling embedded intelligence," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 48–56, 2019. DOI: 10.1109/MIE.2019.2943283.

[72] P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 978–1013, 2018. DOI: 10.1109/COMST.2017.2780114.

[73] S.-Y. Lien, S.-L. Shieh, Y. Huang, B. Su, Y.-L. Hsu, and H.-Y. Wei, "5G New Radio: Waveform, Frame Structure, Multiple Access, and Initial Access," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 64–71, 2017. DOI: 10.1109/MCOM.2017.1601107.

[74] "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Redline," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) - Redline*, pp. 1–5229, 2012.

[75] S. Wang, "Cloud-dew architecure," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[76] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog," *IEEE Access*, vol. 7, pp. 150 936–150 948, 2019.

[77] N. Jangid, "Real time cloud computing," in *1st National Conference on Data Managment & Security (DaMS)*, 2011.

[78] A. Botta, L. Gallo, and G. Ventre, "Cloud, fog, and dew robotics: Architectures for next generation applications," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2019, pp. 16–23.

[79] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[80] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[81] O. C. A. W. Group, "Ieee approved draft standard for adoption of openfog reference architecture for fog computing," *IEEE P1934/D2.0*, pp. 1–175, 2018.

[82] "Ieee standard for adoption of openfog reference architecture for fog computing," *IEEE Std 1934-2018*, pp. 1–176, 2018. DOI: 10.1109/IEEESTD.2018.8423800.

[83] A. K. Tanwani, N. Mor, J. Kubiatowicz, J. E. Gonzalez, and K. Goldberg, *A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering*, 2019. arXiv: 1903.09589 [cs.RO].

[84] L. Qingqing, J. P. Queralta, T. N. Gia, H. Tenhunen, Z. Zou, and T. Westerlund, "Visual odometry offloading in internet of vehicles with compression at the edge of the network," in *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 2019, pp. 1–2.

[85] L. Qingqing *et al.*, "Edge computing for mobile robots: Multi-robot feature-based lidar odometry with fpgas," in *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 2019, pp. 1–2.

[86] K. Antevski, M. Groshev, L. Cominardi, C. J. Bernardos, A. Mourad, and R. Gazda, "Enhancing edge robotics through the use of context information," in *Proceedings of the Workshop on Experimentation and Measurements in 5G*, ACM, 2018, 7–12.

[87] A. Kattepur, H. Dohare, V. Mushunuri, H. K. Rath, and A. Simha, "Resource constrained offloading in fog computing," in *Proceedings of the 1st Workshop on Middleware for Edge Clouds amp; Cloudlets*, ACM, 2016, pp. 1–6.

[88] J. Zhu *et al.*, "An edge computing platform of guide-dog robot for visually impaired," in *2019 IEEE 14th International Symposium on Autonomous Decentralized System (ISADS)*, 2019, pp. 1–7.

[89] I. A. Tsokalo, H. Wu, G. T. Nguyen, H. Salah, and F. H.P. Fitzek, "Mobile edge cloud for robot control services in industry automation," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–2.

[90] D. Pakkala, J. Koivusaari, P. Pääkkönen, and J. Spohrer, "An experimental case study on edge computing based cyber-physical digital service provisioning with mobile robotics," in *Proceedings of the 53rd Hawaii International Conference on System Sciences*, Jan. 2020, pp. 1165–1174.

[91] B. V. Bhausaheb and P. S. Saikrishna, "Control algorithms for a mobile robot application in a fog computing environment," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, ACM, 2019, 30–36.

[92] M. Fu, S. Sun, K. Ni, and X. Hou, "Mobile robot object recognition in the internet of things based on fog computing," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1838–1842.

[93] S. Wan, Z. Gu, and Q. Ni, "Cognitive computing and wireless communications on the edge for healthcare service robots," *Computer Communications*, vol. 149, pp. 99 –106, 2020.

[94] J. P. Queralta, L. Qingqing, Z. Zou, and T. Westerlund, "Enhancing autonomy with blockchain and multi-access edge computing in distributed robotic systems," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 180–187.

[95] Q. Zhang, J. Liu, and G. Zhao, *Towards 5g enabled tactile robotic telesurgery*, 2018. arXiv: 1803.03586 [cs.NI].

[96] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in iot-based manufacturing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 103–109, 2018.

[97] L. Hu, Y. Miao, G. Wu, M. M. Hassan, and I. Humar, "Irobot-factory: An intelligent robot factory based on cognitive manufacturing and edge computing," *Future Generation Computer Systems*, vol. 90, pp. 569 –577, 2019.

[98] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86 769–86 777, 2019.

[99] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Utilizing fog computing for multi-robot systems," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 102–105.

[100] M. S. Shaik *et al.*, "Enabling fog-based industrial robotics systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 61–68.

[101] V. K. Sarker, J. Peña Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "Offloading slam for indoor mobile robots with edge-fog-cloud computing," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 2019, pp. 1–6.

[102] S. Dey and A. Mukherjee, "Robotic slam: A review from fog computing and mobile edge computing perspective," in *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, ACM, 2016, 153–158.

[103] O. Arsene, C. Postelnicu, W. Wang, E. Simona Lohan, and D. Iulian Nastac, "An architecture for indoor location-aided services based on collaborative industrial robotic platforms," in *2019 International Conference on Localization and GNSS (ICL-GNSS)*, 2019, pp. 1–6.

[104] J. Soldatos, S. Kyriazakos, P. Ziafati, and A. Mihovska, "Securing iot applications with smart objects: Framework and a socially assistive robots case study," *Wireless Personal Communications*, Feb. 2020.

[105] Q. Li, Y. Tian, Q. Wu, Q. Cao, H. Shen, and H. Long, "A cloud-fog-edge closed-loop feedback security risk prediction method," *IEEE Access*, vol. 8, pp. 29 004–29 020, 2020.

[106] D. He, Y. Qiao, S. Chan, and N. Guizani, "Flight security and safety of drones in airborne fog computing systems," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 66–71, 2018.

[107] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5g era," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 90–96, 2014.

[108] D. Rapone *et al.*, "An integrated, virtualized joint edge and fog computing system with multi-rat convergence," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2018, pp. 1–5.

[109] S. Nunna *et al.*, "Enabling real-time context-aware collaboration through 5g and mobile edge computing," in *2015 12th International Conference on Information Technology - New Generations*, USA: IEEE Computer Society, 2015, 601–605.

[110] V. Petrov *et al.*, "Achieving end-to-end reliability of mission-critical traffic in softwarized 5g networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 485–501, 2018.

[111] J. Shuja *et al.*, "Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers," *IEEE Systems Journal*, vol. 10, no. 2, pp. 507–519, 2016. DOI: 10.1109/JSYST.2014.2315823.

[112] S. Kim *et al.*, "Demo/poster abstract: Enabling time-critical applications over next-generation 802.11 networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 1–2.

[113] S. Bush and G. Mantelet, "Industrial Wireless Time-Sensitive Networking: RFC on the Path Forward," Avnu Alliance, White paper Version 1.0.3, Jan. 2018.

[114] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[115] S. Barré, C. Paasch, and O. Bonaventure, "Multipath tcp: From theory to practice," in *NETWORKING 2011*, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 444–457.

[116] N. Finn, P. Thubert, B. Varga, and J. Farkas, *Deterministic Networking Architecture*, RFC 8655, Oct. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8655.txt.

[117] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, 2018.

[118] P. Zhao and G. Dán, "A Benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds," *IEEE Transactions on Network and Service Management*, 2018.

[119] F. Malandrino and C.-F. Chiasserini, "Getting the Most Out of Your VNFs: Flexible Assignment of Service Priorities in 5G," in *IEEE WoWMoM*, 2019.

[120] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM*, 2017.

[121] J. Martín-Pérez, F. Malandrino, C. F. Chiasserini, and C. J. Bernardos, "OKpi: All-KPI network slicing through efficient resource allocation," in *IEEE INFOCOM*, 2020.

[122] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[123] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2018, pp. 1545–1550.

[124] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[125] K. Antevski and C. J. Bernardos, "Federation in dynamic environments: Can blockchain be the solution?" *IEEE Communications Magazine*, vol. 60, no. 2, pp. 32–38, 2022. DOI: 10.1109/MCOM.001.2100585.

[126] "IMT Vision: Framework and overall objectives of the future development of IMT for 2020 and beyond," *ITU Recommendation M.2083-0*, 2015.

[127] T. Lukovszki, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *Elsevier Journal of Parallel and Distributed Computing*, 2018.

[128] I. Cohen, G. Einziger, R. Friedman, and G. Scalosub, "Access Strategies for Network Caching," in *IEEE INFOCOM*, 2019.

[129] S. Dräxler, H. Karl, and Z. Á. Mann, "Jasper: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Transactions on Network and Service Management*, 2018.

[130] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks," in *IEEE INFOCOM*, 2019.

[131] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM*, 2018.

[132] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF Placement and Resource Allocation for the Support of Vertical Services in 5G Networks," *IEEE/ACM Transactions on Networking*, 2019.

[133] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, 2005, pp. 18–23.

[134] 3GPP, "Study on Communication for Automation in Vertical Domains," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.804, Dec. 2018, Version 16.2.0.

[135] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on automation science and engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[136] K. Henning, W. Wolfgang, and H. Johannes, *Securing the future of German manufacturing industry: Recommendations for implementing the strategic initiative INDUSTRIE 4.0 (Final report of the Industrie 4.0)*, 2013.

[137] P. Leitão, A. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, 2015. DOI: 10.1016/j.compind.2015.08.004.

[138] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *IEEE Network*, vol. 26, no. 3, pp. 28–34, 2012. DOI: 10.1109/MNET.2012.6201213.

[139] K. Xia *et al.*, "A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence," *Journal of Manufacturing Systems*, vol. 58, pp. 210–230, 2021. DOI: https://doi.org/10.1016/j.jmsy.2020.06.012.

[140] P. Bosch, S. Latré, and C. Blondia, "An analytical model for IEEE 802.11 with non-IEEE 802.11 interfering source," *Computer Networks*, vol. 172, p. 107 154, 2020.

[141] K. Borodulin, G. Radchenko, A. Shestakov, L. Sokolinsky, A. Tchernykh, and R. Prodan, "Towards Digital Twins Cloud Platform: Microservices and Computational Workflows to Rule a Smart Factory," in *Proceedings of The 10th International Conference on Utility and Cloud Computing*, New York, NY, USA, 2017. DOI: 10.1145/3147213.3149234.

[142] M. De Donno, A. Giaretta, N. Dragoni, A. Bucchiarone, and M. Mazzara, "Cyber-storms come from clouds: Security of cloud computing in the IoT era," *Future Internet*, vol. 11, no. 6, 2019.

[143] L. Girletti, M. Groshev, C. Guimarães, C. J. Bernardos, and A. de la Oliva, "An Intelligent Edge-based Digital Twin for Robotics," in *IEEE Globecom Workshop on Advanced Technology for 5G Plus*, Taipei, Taiwan, Dec. 2020.

[144] M. Z. Alom *et al.*, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, no. 3, 2019.

[145] ETSI, "Developing Software for Multi-Access Edge Computing," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 2nd edition, Feb. 2019.

[146] 5GPP and 5GIA, *The european 5G annual Journal/2021*, https://bscw.5g-ppp.eu/pub/bscw.cgi/d424095/5G%20European%20Annual%20Journal%202021.pdf, Online; accessed 28 October 2021, 2021.

[147] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network Intelligence in 6G: Challenges and Opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '21, New York, NY, USA: Association for Computing Machinery, 2021, 7–12. DOI: 10.1145/3477091.3482761. [Online]. Available: https://doi.org/10.1145/3477091.3482761.

[148] M. A. Uusitalo *et al.*, "Hexa-x the european 6g flagship project," in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2021, pp. 580–585. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482430.

[149] J. E. Solanes, A. Muñoz, L. Gracia, A. Martí, V. Girbés-Juan, and J. Tornero, "Teleoperation of industrial robot manipulators based on augmented reality," *The International Journal of Advanced Manufacturing Technology*, vol. 111, no. 3, pp. 1077–1097, 2020.

[150] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019. DOI: 10.1109/LRA.2019.2917707.

[151] T. Zhang *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," pp. 5628–5635, 2018. DOI: 10.1109/ICRA.2018.8461249.

[152] J. E. Solanes, A. Muñoz, L. Gracia, A. Martí, V. Girbés-Juan, and J. Tornero, "Teleoperation of industrial robot manipulators based on augmented reality," *The International Journal of Advanced Manufacturing Technology*, vol. 111, no. 3, pp. 1077–1097, 2020.

[153] X. Li *et al.*, "5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021. DOI: 10.1109/MCOM.001.2000730.

[154] B. H. Jafari and M. W. Spong, "Passivity-based switching control in teleoperation systems with time-varying communication delay," in *2017 American Control Conference (ACC)*, 2017, pp. 5469–5475. DOI: 10.23919/ACC.2017.7963805.

[155] D. Sun, F. Naghdy, and H. Du, "Neural network-based passivity control of teleoperation system under time-varying delays," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1666–1680, 2017. DOI: 10.1109/TCYB.2016.2554630.

[156] C. Ott, J. Artigas, and C. Preusche, "Subspace-oriented energy distribution for the time domain passivity approach," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 665–671. DOI: 10.1109/IROS.2011.6094697.

[157] M. Franken, B. Willaert, S. Misra, and S. Stramigioli, "Bilateral telemanipulation: Improving the complementarity of the frequency- and time-domain passivity approaches," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2104–2110. DOI: 10.1109/ICRA.2011.5979969.

[158] H. K. Khalil, *Nonlinear systems*. Prentice-Hall, 2002.

[159] E. Mujčić, A. Mujčić, and S. Pajazetović, "Internet-based teleoperation using wave variables and correction of position error," in *2016 International Conference on Smart Systems and Technologies (SST)*, 2016, pp. 219–224. DOI: 10.1109/SST.2016.7765663.

[160] D. Sun, F. Naghdy, and H. Du, "Transparent four-channel bilateral control architecture using modified wave variable controllers under time delays," *Robotica*, vol. -1, pp. 1–17, Jul. 2014. DOI: 10.1017/S026357471400191X.

[161] D. Sun, F. Naghdy, and H. Du, "Wave-variable-based passivity control of four-channel nonlinear bilateral teleoperation system under time delays," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 238–253, 2016. DOI: 10.1109/TMECH.2015.2442586.

[162] G. Niemeyer and J.-J. Slotine, "Stable adaptive teleoperation," *IEEE Journal of Oceanic Engineering*, vol. 16, no. 1, pp. 152–162, 1991. DOI: 10.1109/48.64895.

[163] D.-H. Zhai and Y. Xia, "Adaptive control for teleoperation system with varying time delays and input saturation constraints," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 6921–6929, 2016. DOI: 10.1109/TIE.2016.2583199.

[164] Y.-C. Liu and M.-H. Khong, "Adaptive control for nonlinear teleoperators with uncertain kinematics and dynamics," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2550–2562, 2015. DOI: 10.1109/TMECH.2015.2388555.

[165] P. M. Kebria, A. Khosravi, S. Nahavandi, P. Shi, and R. Alizadehsani, "Robust adaptive control scheme for teleoperation systems with delay and uncertainties," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3243–3253, 2020. DOI: 10.1109/TCYB.2019.2891656.

[166] Z. Chen, F. Huang, W. Sun, J. Gu, and B. Yao, "RBF-Neural-Network-Based Adaptive Robust Control for Nonlinear Bilateral Teleoperation Manipulators With Uncertainty and Time Delay," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 2, pp. 906–918, 2020. DOI: 10.1109/TMECH.2019.2962081.

[167] A Forouzantabar, H. Talebi, and A. Sedigh, "Adaptive neural network control of bilateral teleoperation with constant time delay," *Nonlinear Dynamics*, vol. 67, no. 2, pp. 1123–1134, 2012.

[168] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[169] P. M. de Sant Ana, N. Marchenko, P. Popovski, and B. Soret, "Wireless control of autonomous guided vehicle using reinforcement learning," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7. DOI: 10.1109/GLOBECOM42002.2020.9322156.

[170] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[171] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. Citeseer, 1994, vol. 37.

[172] C. Lozoya, P. Martí, M. Velasco, J. Fuertes, and E. Martín, "Simulation study of a remote wireless path tracking control with delay estimation for an autonomous guided vehicle," *The International Journal of Advanced Manufacturing Technology*, vol. 52, pp. 751–761, Feb. 2011. DOI: 10.1007/s00170-010-2736-x.

[173] M. Zorzi, R. Rao, and L. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 46, no. 2, pp. 445–455, 1997. DOI: 10.1109/25.580783.

[174] J. R. Jackson, "Networks of waiting lines," *Operations research*, vol. 5, no. 4, pp. 518–521, 1957.

[175] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[176] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, 2014, pp. 3104–3112.

[177] P. P. Pham, "Comprehensive analysis of the IEEE 802.11," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 691–703, 2005.

[178] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

[179] G. I. Palmer, V. A. Knight, P. R. Harper, and A. L. Hawa, "Ciw: An open-source discrete event simulation library," *Journal of Simulation*, vol. 13, no. 1, pp. 68–82, 2019. DOI: 10.1080/17477778.2018.1473909.

[180] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[181] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[182] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[183] W. H. Greene, *Econometric analysis*. Pearson Education India, 2003.

[184] N. Sugiura, "Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's," *Communications in Statistics-theory and Methods*, vol. 7, no. 1, pp. 13–26, 1978.

[185] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*. Springer Science & Business Media, 2009.

Appendix A: Box-Jenkins methodology

In this appendix we show how we applied the Box-Jenkings methodology [180] to decide that VAR with autoregression (AR) order of 10 was the most appropriate model to infer future robot commands $\hat{c}_{i+n}$. In the following, we apply the three stages of the Box-Jenkins methodology in our problem:

1) **Model identification**: the robot trajectory time series is i) stationary; ii) non-seasonal; iii) with causality; and iv) co-integration across the robot axis. Hence, VARMA is an adequate method to infer future commands – see [181]. Below we detail why properties i)-iv) are satisfied:

   i) the time series is stationary because the null-hypothesis (non-stationary) of the Augmented Dickey-Fuller test [182] is rejected with significance level $\alpha = 0.05$. Actually, the test yielded p-values in the range of $p \in [-3.71 \cdot 10^{-13}, -1.52 \cdot 10^{-26}]$ for every robot axis;

   ii) the time series is not seasonal, for the robot operator sometimes takes more time to repeat the pick and place tasks, and all the actions are not exactly the same, i.e, $\nexists T : c_i = c_{i+nT},\ \forall n \in \mathbb{N}$. Moreover, the Partial Autocorrelation Function (PACF) values – see Fig. 6.1 – drastically drop to zero, and do not fall outside the 95% confidence interval. Hence, there is no evidence of seasonality;

   iii) each robot axis Granger-causes [183] the others $c_i^l \sim c_i^{l'},\ \forall l, l'$ because the null-hypothesis (no Granger-causality) of the Granger-causality test was rejected with significance level $\alpha = 0.05$. Indeed, the test yielded p-values $p \leq 0.0144$ when it checked Granger-causality between every pair of axis;

   iv) the robot-axis time series are co-integrated, i.e.:

$$\forall l, \exists \{a_i^l\}_i^{T_l} : \sum_l \sum_i^{T_l} c_i^l a_i^l \sim I(m), \quad m \leq \min_i \{d : c_i^l \sim I(d)\} \tag{6.1}$$

   with $c_i^l \sim I(d)$ meaning that time series of axis l has differentiation order d. In other words, there is a linear combination of the robot axis time series with order of integration below the order of integration of each robot axis individually. In our experiments we used the co-integration Johanson's test [181], and proved that (6.1) holds because we sequentially rejected the null hypotheses of having $k = 0, k \leq 1, \ldots, k \leq 5$ co-integrating vectors with significance level $\alpha = 0.05$.

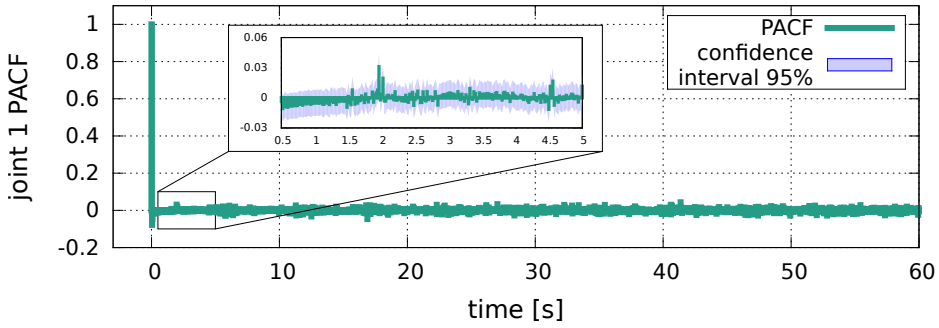


Figure 6.1. *PACF of joint 1 over time. Instants outside the confidence interval suggest possible seasonality.*

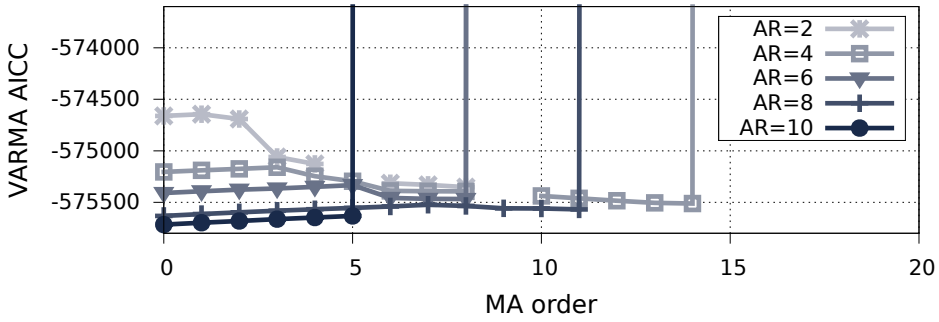


Figure 6.2. *VARMA accuracy using different AR and MA orders.*

2) **Parameter estimation**: after identifying VARMA as a suitable model, we have to estimate its parameters for our robot dataset. VARMA considers that future commands $\hat{c}_{i+n}$ are expressed using prior commands $c_{i-n},\ n \in \mathbb{N}$; and that regression residuals ϵ_{i+1} equal a moving average of prior residuals $\epsilon_{i-n},\ n \in \mathbb{N}$. Following Section 5.3.3.2 notation, VARMA looks as follows:

$$\hat{c}^k_{i+1} = f^k\left(\{\hat{c}_j\}^i_{i-R}, \vec{w}\right) = b^k + \sum_{l=1}^{d}\sum_{j=i-R}^{i} w^l_{i,j}\cdot\hat{c}^l_j + \epsilon^k_{i+1} + \sum_{l=1}^{d}\sum_{j=i-M}^{i} w'^l_{i,j}\cdot\epsilon^l_j, \quad k \le d \tag{6.2}$$

with $\vec{w} = \{\{w^l_{i,j}\}_{i,j,l}; \{w'^l_{i,j}\}_{i,j,l}\}$ the weights for the VARMA model, R the autoregression (AR) order, $w'^l_{i,j}$ the MA weights, and M the MA order.
To select the best parameters (R, M), we have trained different VARMA models with $1 \le R \le 10$, $0 \le M \le 10$; and compared their qualities using the Akaike Information Criterion Corrected [184], as authors of [185] do. Figure 6.2 illustrates the AICC achieved with different combinations of the AR and MA order. Results show that only the least accurate models benefit from increasing their MA order. Whilst the best models, with higher AR order, only get worse as the MA order increases.
From Fig. 6.2 we conclude that $(R = 10, M = 0)$ is the most accurate model, i.e., the VAR model that we use throughout the paper.

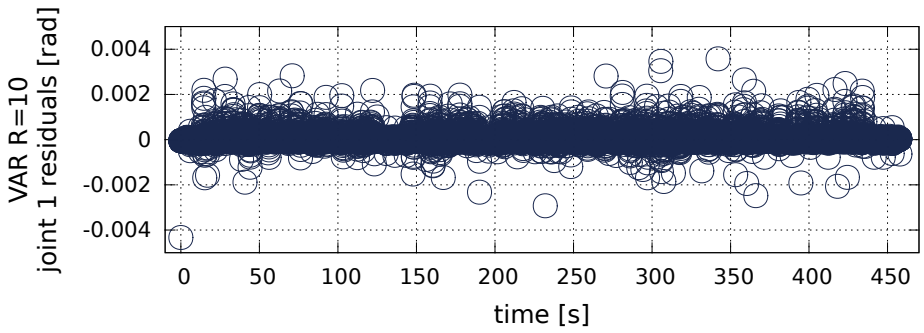


Figure 6.3. *Joint 1 residuals for VAR with AR order* $R = 10$.

3) **Statistical model checking**: to check the validity of VAR with $R = 10$ we run the Ljung-Box test [181], with null-hypothesis of no correlation across the residuals of the fitted VAR model. We could not rejected the null-hypothesis because the obtained p-values where $p > 0.99$ for every axis. Therefore, the residuals of VAR with $R = 10$ are not correlated. Moreover, despite some outliers, the residuals of all axis maintain a constant mean and variance over time – see[26] Figure 6.3. Hence, Box-Jenkins methodology [180] suggests that VAR with AR order $R = 10$ is an adequate estimator of future commands $\hat{c}_{i+n}$, $n \in \mathbb{N}$.

[26] For the other axis residuals have the same pattern

Appendix B: IEEE 802.11 analytical insights

In the following, we present some theoretical results about the expected delay of a command, and the causality assumption in the IEEE 802.11 scenario considered in this paper.

Lemma 1. *A control command c_i traversing a transport network, and an IEEE 802.11 wireless link under interference will experience an average delay satisfying*

$$\mathbb{E}\left[\Delta(c_i)\right] \leq D + \frac{1}{1-a_{m+2}} \sum_{j=0}^{m+1} a_j \cdot \mathbb{E}_j\left[\Delta_W(c_i)\right], \quad \forall c_i \tag{6.3}$$

with probability $1-a_{m+2}$*, and*

$$\mathbb{E}\left[\Delta(c_i)\right] = \infty, \quad \forall c_i \tag{6.4}$$

with probability a_{m+2}*.* $m+2$ *being the maximum number of allowed re-transmissions in IEEE 802.11 wireless links.*

Proof. In case a command is not lost in the IEEE 802.11 wireless link (less than $m+2$ re-transmissions), if we take the law of total probability and the analytical model in [140], the average wireless delay is

$$\mathbb{E}\left[\Delta_W(c_i)\right] = \sum_{j=0}^{m+1} a_j \cdot \mathbb{E}_j\left[\Delta_W(c_i)\right], \quad \forall c_i \tag{6.5}$$

Note that this happens with probability $1-a_{m+2}$. Therefore, if we foresee that a command is not lost, we have to rescale (6.5) by $\frac{1}{1-a_{m+2}}$, i.e., the probability that a command is not lost.

Since we know that $\Delta(c_i) = \Delta_T(c_i) + \Delta_W(c_i)$, if we take the expectation at both sides of the equatlity, use Assumption 1, and the rescaled version of (6.5); we obtain (6.3).

According to [140], (6.4) holds because a packet is lost $\Delta_W(c_i) = \infty$ in a IEEE 802.11 wireless link with probability a_{m+2}. □

Corollary 1. *A control command c_i traversing a transport network, and an IEEE 802.11 wireless link under interference, experiences an unbounded delay, that is*

$$\mathbb{P}\Big(\Delta(c_i) > K,\ \forall K \in \mathbb{R}\Big) > 0 \tag{6.6}$$

Proof. In particular, Lemma 1 says that $\mathbb{P}\Big(\Delta(c_i) > K,\ \forall K \in \mathbb{R}\Big) = a_{m+2}$. □

Lemma 2. *In IEEE 802.11 wireless links, the causality assumption*

$$|\Delta(c_{i+1}) - \Delta(c_i)| \leq |g(c_{i+1}) - g(c_i)|, \quad \forall c_{i+1}, c_i \tag{6.7}$$

only holds on average with probability $\sum_{j=0}^{m+1} a_j^2$.

Proof. We prove the lemma by cases, namely considering the different combinations of required re-transmissions of commands c_i and c_{i+1}.

If either command c_i or c_{i+1} is lost ($m+2$ re-transmissions), then we have with probability a_{m+2} that equation 6.7 does not hold, since either $\Delta_W(c_i) = \infty$ or $\Delta_W(c_{i+1}) = \infty$;

If c_{i+1} has j_2 RTX, and c_i has j_1 RTX (with $j_1 < j_2$), then

$$\Delta_W(c_{i+1}) - \Delta_W(c_i) = |\Delta_W(c_{i+1} - \Delta_W(c_i)| = \leq g(c_{i+1}) - g(c_i) \tag{6.8}$$

We can take the expectation on the left side hand and obtain:

$$\mathbb{E}[\Delta_W(c_{i+1})] - \mathbb{E}[\Delta_W(c_i)] = T_s + j_2 T_{col} + \tilde{\sigma} \sum_{k=0}^{j_2} \frac{W_k - 1}{2} - T_s - j_1 T_{col} - \tilde{\sigma} \sum_{k=0}^{j_1} \frac{W_k - 1}{2} =$$
$$(j_2 - j_1) T_{col} + \tilde{\sigma} \sum_{k=j_1+1}^{j_2} \frac{W_k - 1}{2} \tag{6.9}$$

with T_s the transmission time, T_{col} the collision time, $\tilde{\sigma}$ the average slot time, and W_k the k^{th} back-off window. Based on (6.9), the causality assumption does not hold, since

$$\exists \quad T_s, T_{col}, \tilde{\sigma}, W_k \quad : \quad (j_2 - j_1) T_{col} + \tilde{\sigma} \sum_{k=j_1+1}^{j_2} \frac{W_k - 1}{2} \quad > \quad g(c_{i+1}) - g(c_i) \tag{6.10}$$

The same reasoning applies in the case $j_1 > j_2$.

If command c_i required same re-transmissions as command c_{i+1} (i.e. $j_1 = j_2$), then (6.9) $\mathbb{E}[\Delta_W(c_{i+1})] = 0$, and the causality assumption (6.7) holds. This event occurs with probability $\sum_{j=0}^{m+1} a_j^2$. □

Corollary 2. *In IEEE 802.11 wireless links, the causality assumption*

$$|\Delta(c_{i+1}) - \Delta(c_i)| \leq |g(c_{i+1}) - g(c_i)|, \quad \forall c_{i+1}, c_i \tag{6.11}$$

does not hold.

Proof. The causality assumption does not hold with probability 1 (see Lemma), therefore, the causality assumption does not hold in IEEE 802.11 wireless links. □