UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RICARDO CARMINATI DE MELLO

A METHODOLOGY FOR CLOUD ROBOTICS IMPLEMENTATION BASED ON OPEN-SOURCE SOFTWARE: FROM HUMAN-ROBOT INTERACTION TO AUTONOMOUS APPLICATIONS

VITÓRIA
2020

Ricardo Carminati de Mello

A Methodology for Cloud Robotics Implementation Based on Open-Source Software: from Human-Robot Interaction to Autonomous Applications

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Anselmo Frizera Neto
Coorientador: Prof. Dr. Moisés Renato Nunes Ribeiro

Vitória
2020

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Mello, Ricardo Carminati de, 1990-
M527m A methodology for cloud robotics implementation based on open-source software : from human-robot interaction to autonomous applications / Ricardo Carminati de Mello. - 2020.
124 f. : il.

Orientador: Anselmo Frizera Neto.
Coorientador: Moisés Renato Nunes Ribeiro.
Tese (Doutorado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Engenharia elétrica. 2. Robótica. 3. Computação em nuvem. 4. Sistemas homem-máquina. 5. Sistemas de comunicação sem fio. I. Frizera Neto, Anselmo. II. Ribeiro, Moisés Renato Nunes. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

RICARDO CARMINATI DE MELLO

A METHODOLOGY FOR CLOUD ROBOTICS IMPLEMENTATION BASED ON OPEN-SOURCE SOFTWARE: FROM HUMAN-ROBOT INTERACTION TO AUTONOMOUS APPLICATIONS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Doutor em Engenharia Elétrica.

Aprovada em 26 de novembro de 2020.

COMISSÃO EXAMINADORA

Prof. Dr. Anselmo Frizera Neto
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Moisés Renato Nunes Ribeiro
Universidade Federal do Espírito Santo
Coorientador

Prof. Dr. Adriano Almeida Gonçalves Siqueira
Universidade de São Paulo
Membro Externo

Prof. Dr. Camilo Arturo Rodríguez Díaz
Universidade Federal do Espírito Santo
Membro Interno

Prof. Dr. Carlos Andres Cifuentes Garcia
Escuela Colombiana de Ingeniaría Julio Garavito
Membro Externo

Prof. Dr. Eduardo Rocon de Lima
Spanish National Research Council
Membro Externo

Dedico este trabalho aos meus pais, Dayse e Claudio.
Tudo que sou, devo a vocês.

Acknowledgments

First, I would like to acknowledge the funding that made it possible for me to dedicate myself to the research that led to this work. This study was financed in part by CAPES – Finance Code 001 – and by the European Commission H2020 program under the FUTEBOL grant agreement, as well from the Brazilian Ministry of Science, Technology, and Innovation. I thank CNPq and FAPES for supporting research projects related to this work. This work was also partially supported by the Colombian Ministry of Science, Technology, and Innovation Minciencias (Grant 801-2017).

I wish to express my deepest gratitude to my advisors and dear friends Professor Anselmo Frizera Neto and Professor Moises R. N. Ribeiro, without whom this work would not be. You have provided me with a safe space to learn and to question – and to be questioned. A safe space to focus on our current research while also dreaming about the future. Chance led me to both of you years ago and today I can not think of a better pair of advisors. I thank you with all my heart.

A substantial amount of the work presented in this thesis was carried out during research stays abroad and I want to thank the researchers that opened the doors of their research groups to me. I thank Professor Carlos Cifuentes and Professor Marcela Múnera for receiving me at the Colombian School of Engineering Julio Garavito and Professor Eduardo Rocon for receiving me at the Centre for Automation and Robotics - CSIC-UPM. Those were invaluable opportunities to develop our research and to learn from different points of view.

I thank all of my colleagues and friends for their support. A special thanks to Dr. Mario Jimenez and our CloudWalker team, Wandercleyson Scheidegger and Joelson Rocha, for all the work we put together. I am also grateful for everything the NERDS group did for me and I cannot express how much I've learned from all of you. In particular, I'd like to thank the other members of the Kamikaze team, my good friends Victor Garcia, Rafael Guimarães, and Pedro Hasse. I was also privileged enough to have as my personal networking consultants my friends João Henrique Corrêa and Diego Mafioletti, and I must thank you for all your patience while assisting and teaching me. I also thank those who welcomed me while in Colombia and in Spain, and an special acknowledgment is due to Sergio Sierra for all his help and companionship.

Finally, I wish to thank my parents, Claudio and Dayse, my brother, Leonardo, and my partner, Katherine, for everything they have done for me during all these years. You are also a fundamental part of this work.

[...] it's my opinion that anyone who can possibly introduce science to the nonscientist should do so. We don't want scientists to become a priesthood. We don't want society's technological thinkers to know something that nobody else knows, because such a situation would lead to public fear of science and scientists. And fear, as you know, can be dangerous.

Dr. Isaac Asimov, 1962.

Resumo

O advento do paradigma da robótica em nuvem tem o potencial de habilitar e impactar uma nova geração de dispositivos robóticos inteligentes. Este paradigma permite que robôs se comuniquem entre si e com plataformas remotas de computação para compartilhar experiências, dados e para delegar a execução de tarefas que demandam alto processamento. Este trabalho explora a robótica em nuvem direcionando um olhar sobre algumas das principais questões do paradigma para propor novas soluções que permitam a ampla adoção de funcionalidades de nuvem tanto na pesquisa em robótica quanto em robôs comerciais. Neste trabalho, investiga-se o paradigma da robótica em nuvem como um habilitador de uma série de aplicações e dispositivos. Também se questiona como a inserção de tecnologias de redes de comunicação e de computação em nuvem podem impactar a interação entre operadores humanos e robôs e quais são as limitações e requerimentos ao trabalhar neste paradigma. É proposta uma arquitetura de referência para robótica em nuvem acompanhada com uma solução integrada de redes a fim de fornecer conectividade ininterrupta para aplicações com robôs móveis. Ademais, é apresentado um *framework* aberto de comunicação para aproximar robôs e plataformas de nuvem, o qual atende às especificações de uma vasta gama de aplicações. Finalmente, é introduzida uma metodologia para implementação de robótica em nuvem baseada em software aberto e em equipamentos disponíveis comercialmente. Tal metodologia usa este *framework* de comunicação e pode ser utilizada como uma base comum para experimentação, possibilitando que diferentes trabalhos de robótica em nuvem sejam reproduzidos e comparados. A metodologia proposta e o *framework* de comunicação são validados considerando diferentes aplicações e variadas configurações de nuvem, demonstrando assim a aptidão desta abordagem e ampliando o conhecimento sobre o paradigma.

Palavras-chave: Robótica em nuvem, ROS, sistemas ciber-físicos, dispositivos assistivos, navegação autonoma, comunicação sem fio.

Abstract

The advent of the cloud robotics paradigm has the potential to unleash a whole new generation of smart robotic devices by allowing robots to communicate with each other and with remote computing platforms to share experiences, sensor data, and to offload heavy processing applications. This work explores cloud robotics by casting a light on key issues and proposing novel solutions to allow for the widespread adoption of cloud-based functionality for research and commercial robots. We investigate the cloud robotics paradigm as an enabler to a series of applications and devices and question how the insertion of network and cloud technologies into such solutions might affect the interaction between a robot and a human operating it and what are the limiting requirements for cloud-based solutions. We propose a reference architecture for cloud robotics paired with an integrated network solution to allow for uninterrupted connectivity in mobile robotics applications. Furthermore, we present an open communication framework to link robots and cloud platforms, and that suits a large range of applications. Finally, we introduce a methodology for cloud robotics implementation based on open-source software and commercial off-the-shelf devices. Such a methodology leverages our communication framework and provides a common standard that allows reproducing and benchmarking different cloud robotics works. Our methodology and communication framework are validated considering different applications and multiple cloud configurations, showing the suitability of our approach and providing insight for other researchers and practitioners.
Keywords: Cloud robotics, ROS, cyber-physical systems, assistive devices, autonomous navigation, wireless communication.

List of Acronyms

3GPP	3rd Generation Partnership Project
CPS	Cyber-Physical System
E2E	end-to-end
eNF	embedded Network Function
fSTA	fixed Client Station
HaaS	Healthcare as a Service
HREI	Human-Robot-Environment-Interaction
HRI	Human-Robot-Interaction
IoT	Internet of Things
KTE	Kinematic Tracking Error
LRF	Laser Rangefinder
mAP	mobile Access Point
MIT	Mobility Interruption Time
MOS	Mean Opinion Score
mSTA	mobile Client Station
NAT	Network Address Translation
NERDS	Software-Defined Networks Research Group
NTA	Assistive Technology Group
OS	Operating System
QoE	Quality of Experience
QoS	Quality of Service
R2I	Robot-to-Infrastructure
R2R	Robot-to-Robot
RAN	Radio Access Network
ROS	Robot Operating System
RTT	Round Trip Time
SDN	Software Defined Networking
SFC	Service Function Chainning
SLAM	Simultaneous Localization and Mapping
UFES	Universidade Federal do Espírito Santo
URLLC	Ultra-Reliable Low-Latency Communication
VM	Virtual Machine
VNF	Virtual Network Function

List of Figures

List of Tables

Contents

1 Introduction

This thesis explores the cloud robotics paradigm by casting a light on key issues of the subject and proposing novel solutions to allow for the widespread adoption of cloud-based functionality for research and commercial robots. We investigate cloud robotics as an enabler to a series of applications and devices and question what are the requirements of cloud robotics systems. We also look into how the insertion of network and cloud technologies into such systems might affect the interaction among robots and humans, and limit its use. Furthermore, we present a methodology for cloud robotics implementation capable of satisfying the requirements of multiple applications. Our methodology leverages a novel common framework based on open-source software and provides a common standard that can be used to allow reproducing and benchmarking different cloud robotics works. We perform a series of experiments considering multiple configurations to validate our methodology in different applications. In this chapter, we introduce the motivations behind this thesis, our goals, and detail the organization of the thesis.

1.1 Motivation

Cloud robotics emerged as a paradigm capable of revolutionizing the robotics field [1]. Envisioned as a key factor for the upcoming generation of service robots, cloud-enabled robots will not only co-exist within the Internet of Things (IoT), but also are expected to play an important role in areas such as eHealth and Industry 4.0 (also known as Industrial IoT) [2]. Several cloud robotics platforms are being developed by the scientific community [3, 4] whereas tech giants, such as Google and Amazon, are releasing their commercial platforms. Furthermore, cloud robotics concepts are rapidly spreading, as indicated by the success of Softbank's Pepper[1] and Anki's Vector[2], to cite a few of the cloud-enabled robots available in the market. Nevertheless, despite all the recent efforts, the cloud robotics field is yet to be fully explored.

The increasing complexity posed by robotic tasks faces challenges regarding resource constraints imposed by embedded hardware (i.e., processing, storage, learning, and information gathering) [1, 2]. Since each robot is limited by its hardware, the possibility of offloading processing tasks to and centralizing information on cloud platforms allows for more cost-effective robots operating – and cooperating – in unstructured environments [5]. Recent advancements in cloud computing and communication technologies have enhanced the capacity of commercial providers to meet the levels of QoS required by several classes

[1] Website: <https://www.softbankrobotics.com/us/pepper>. Accessed on December 1, 2020.
[2] Website: <https://www.digitaldreamlabs.com/pages/new-with-vector>. Accessed on December 1, 2020.

of applications. In this context, cloud robotics is being placed by researchers as a viable paradigm due to the possibility of seamlessly integrating all of its composing technologies [5].

As many of the robotics sub-fields may be impacted by cloud robotics concepts, cloud-enabled systems must be designed considering all sorts of implications to achieve some degree of robustness. Thus, it is important to draw clear requirements when designing this kind of system. To this end, it is necessary to understand how the QoS provided – considering Internet and cloud providers' point of view – might affect the execution of tasks [6, 7]. For instance, Human-Robot-Interaction (HRI) is crucial in areas such as social robotics and eHealth, especially when the interaction occurs in a physical level and the human can be considered to be inside the control loop [8, 9]. This is often called the human-in-the-loop factor since the interaction with the person must be taken into account when designing the control system [10, 11]. Nevertheless, the literature is yet to fully comprehend how network- and cloud-related issues may impact HRI in cloud-aided systems [7, 12, 13].

In general, critical applications present stringent requirements regarding deadlines for the execution of processing tasks, especially when real-time control is needed [14, 15]. Reliable and predictable communications are desirable factors and efforts are being made to push the next-generation technologies towards such standards [16]. For instance, the 3rd Generation Partnership Project (3GPP) defines for 5G networks a target latency of 10 ms and packet loss probability smaller than $10^{-5}\%$ for discrete automation in Ultra-Reliable Low-Latency Communication (URLLC) [17], which largely affects applications for eHealth and Industry 4.0. Nevertheless, it will take at least a few years until these novel technologies start to become commonplace. Meanwhile, current leading technologies such as WiFi are ubiquitous in residential, commercial, and even industrial contexts, and must be considered when designing and implementing cloud robotics systems [18, 19]. In this sense, several solutions have been presented to leverage the Software Defined Networking (SDN) paradigm over conventional networks to enhance QoS, which may be decisive for fostering the adoption of cloud robotics concepts even before the next generation of networks becomes a reality [20–22].

Considering the state-of-the-art in cloud robotics literature research is either focused on application-based approaches, which tend to leverage the cloud as a tool for improving performance, or on system-level and architectural approaches (refer to Fig. 1 for examples of cloud robotics systems and applications) [2]. In particular, system-level approaches help in paving the way for cloud robotics implementations by discussing the interaction among the multiple building blocks of cloud-enabled systems, as well as the technologies and equipment involved [2, 23]. In this sense, efforts for developing cloud robotics platforms, such as the ones presented by Mohanarajah *et al.* [24] and Sugiura *et al.* [25], may indicate common standards and good implementation practices. Moreover, cloud robotics testbeds for experimentation [14, 26] may accelerate research by providing specific solutions while

relieving researchers from implementing the whole cloud robotics stack.

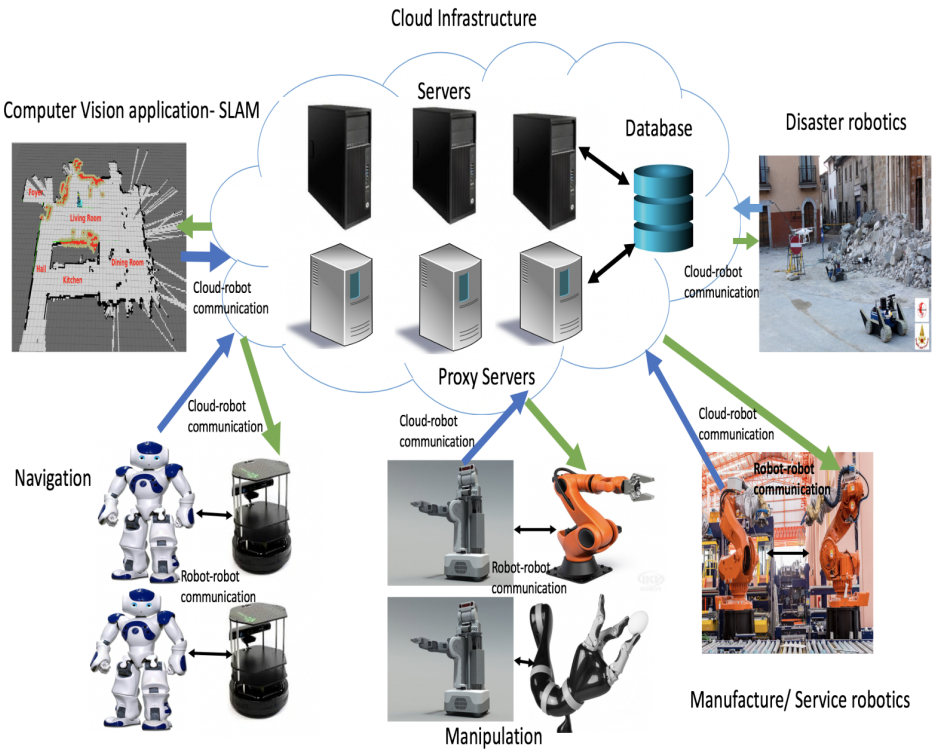


Figure 1 – High-level representation of some of the main cloud robotics system architecture characteristics and applications. Extracted from Saha & Dasgupta [2], licensed under CC BY 4.0.

Despite the promising capabilities and widely reported results, cloud robotics literature still lacks comprehensive analysis of implementation standards and network-related issues. For instance, in Saha & Dasgupta [2] – a recent literature review on the field – the vast majority of the assessed works are conference papers that do not show comprehensive analysis on implementation, experimentation, and obtained results. Thus, not only reproducibility decreases but also comparing different works becomes harder. Moreover, several implementation-focused works do not evaluate cloud robotics per se, but rather evaluate the performance of new algorithms on top of the cloud robotics paradigm (e.g., [27, 28]).

The adoption of common frameworks may address the lack of standardized practices in the field [2]. Many applications are being built on top of the ROS [29], which is becoming a standard for developing general robotics applications [30]. Nevertheless, despite its widespread use, ROS is designed for in-LAN communication and there is no consensus on how to achieve effective robot-cloud communication. Furthermore, works of tutorial nature are still scarce and the integration of the cloud robotics stack has not been thoroughly addressed either with or without ROS. Thus, efforts for establishing common practices are fundamental for the solidification of the field.

In this sense, this work addresses multiple aspects of cloud robotics with a special

focus on its fundamental aspects. Our approach first looks into the requirements of cloud robotics systems, particularly addressing the complexities of HRI and mobile robots. We then identify key limitations in the state-of-the-art to propose solutions to enable cloud robotics and satisfy its requirements. Finally, this thesis culminates into a reproducible methodology for cloud robotics implementations based on open-source software to enable multiple cloud-based applications.

1.2 Background

This work was started and carried out in association with the Assistive Technology Group (NTA) and the Software-Defined Networks Research Group (NERDS), both linked to the Graduate Program of Electrical Engineering at the UFES. The NTA has a large record of research on assistive and rehabilitative technologies, including smart walkers, mainly focused on HRI and Human-Robot-Environment-Interaction (HREI). The NERDS group develops research on cloud computing and programmable datacenter networks. Researchers at NERDS possess the expertise in managing cloud platforms and physical infrastructure resources, aiming at providing optimized levels of service for applications being executed on the cloud. From 2016 to 2019, the group participated in the FUTEBOL project, an EU-Brazil cooperation project under the European Commission H2020 program with a research focus on wireless-fiber integration for telecommunication. In particular, a major part of the UFES' role within FUTEBOL was related to cloud robotics experimentation for Industry 4.0. With the NTA, our group has also led research efforts in the CloudWalker project to develop a cloud-enabled smart walker. In some sense, the overall work developed by our groups also motivated this research on cloud-enabled cyber-physical systems.

In this context, this work expands our previous research efforts and explores existing gaps in the cloud robotics literature. To study the multiple aspects of cloud robotics and the integration of its subsystems, we must first define the scope of our research. We consider the use of cloud robotics in mobile robots that may interact with people and other robots to perform their tasks. We also consider the possibility of using either edge cloud platforms or remote cloud platforms – accessed via the public Internet – to process sensor data and to execute control functionality. Thus, we first look into the requirements of such systems to propose solutions to enable its implementation. Figure 2 illustrates a generic use case considering the scope of this work and discriminates the main components of a cloud robotics system and some of its features.

As illustrated in Fig. 2, the HRI is an aspect present in multiple robotics systems. When considering assistive robots, such as smart walkers (e.g., [31]), the HRI is of fundamental importance. Mobility assistance is a sensitive application in which an impaired individual is in close physical contact with the robot. This places an increased challenge when considering the use of the cloud in such a class of application since any cloud- or communication-related

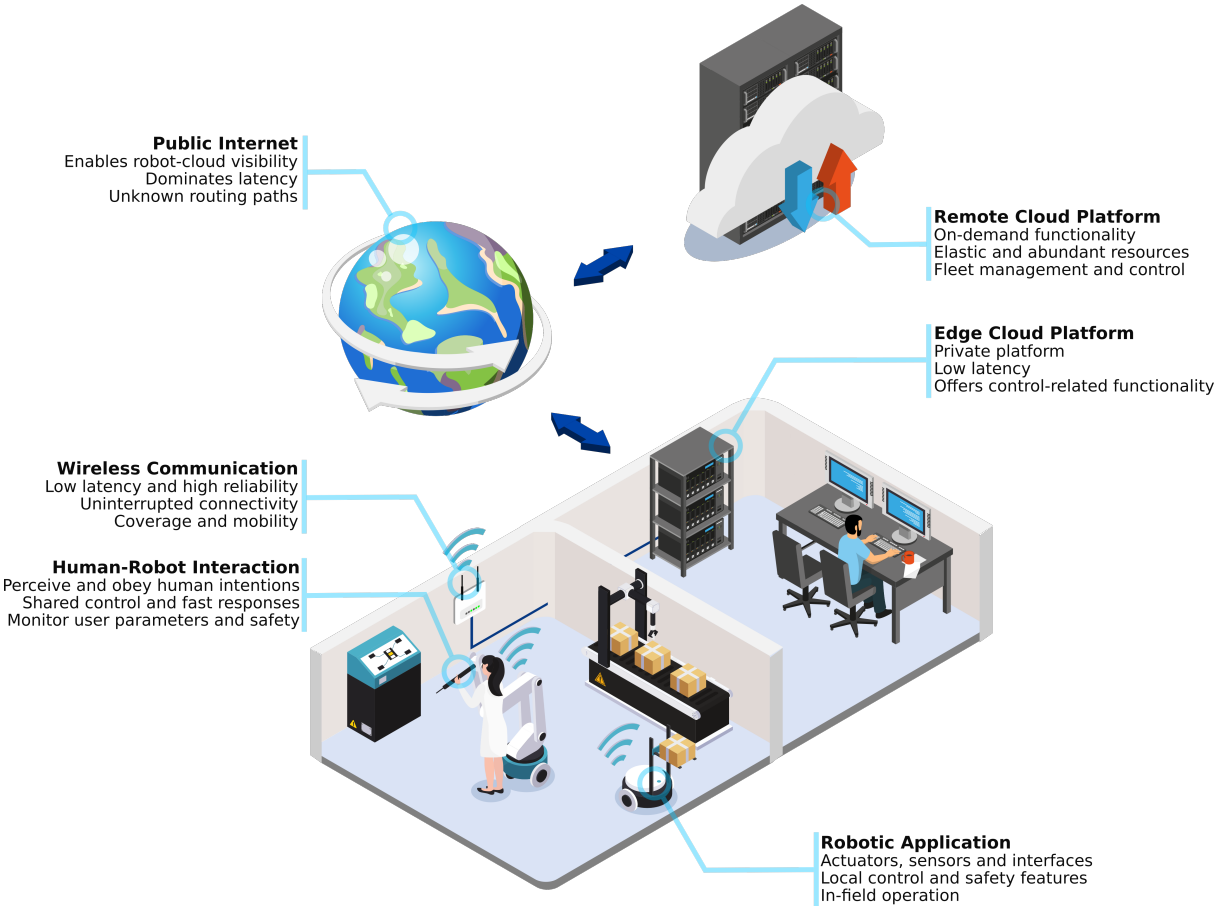


Figure 2 – Generic use case illustrating the main parts of a cloud robotics system: edge and remote cloud platforms as enablers of the operation of mobile robots in an industrial environment.

problem may result in unsafe interaction. In a previous work of ours, we have identified the need for assuring acceptable levels of QoS to meet the strict safety and the usability requirements of those systems [32]. The QoS is affected by all of the components linking the robot and the cloud (i.e., in Fig. 2, those are the wireless communication, the public Internet, and the cloud platform itself). Thus, a holistic view is necessary to further understand the requirements of these systems and the effects of network- and cloud-related issues upon the HRI and the human-in-the-loop factor.

Although latency is a major factor impacting every networked system, mobile robots working in indoor environments – a reasonable portion of the entire robotics world – are also particularly prone to lose wireless network connectivity. In such environments, connectivity loss is mostly caused by electromagnetic interference, direct signal obstruction, or access point migration. Whereas the first two reasons are likely associated with the use of long-range wireless technologies (e.g., LTE), the last one is a common issue in WiFi networks, which are currently ubiquitous in indoor locations. Since most commercial mobile robots integrate at least one of the WiFi standards by default, the WiFi seems particularly suitable for current cloud robotics applications [18, 19]. Moreover, given its

high throughput capabilities, the WiFi can handle the transfer of large amounts of data gathered by sensors such as video cameras [18]. The challenge is to use the WiFi in such a way to achieve the wireless communication features listed in Fig. 2, such as uninterrupted connectivity and high coverage area. Therefore, while we observe a need to consider the use of WiFi in indoor environments, there is also a need to develop solutions to use WiFi in cloud robotics by minimizing latency and momentary loss of connectivity during access point migration.

Finally, multiple design decisions must be made to implement a use case such as the one represented in Fig. 2. For instance, one may choose to construct an on-site cloud platform or to use a commercial cloud. In case the latter approach is chosen, vendor-specific services may be used, which hampers system portability. As a second example of system complexity, the software implementation itself largely depends on the development framework chosen. Besides proprietary solutions that are often close sourced, the implementation of cloud robotics systems in the research community also varies considerably. Particularly, we identified the absence of a standard framework for robot-cloud communication to be a major factor limiting factor. Thus, another key issue in the current cloud robotics state-of-the-art is the lack of standardization in experimental methodology and implementation practices. Since the reproducibility of experimental results is a main component of the scientific method, the lack of experimental replication in robotics research hampers research progress and further exploration of previous results by different researchers [33]. The IEEE currently considers reproducibility one of the priorities in robotics research and is pushing efforts towards reproducible experiments and benchmarking [33, 34]. Therefore, we work on the gap involving the lack of a communication framework and reference methodology to allow other researchers to replicate our experiments and methods, and to benchmark new systems and architectures.

1.3 Objectives

The general objective of this thesis is to develop a common methodology for cloud robotics experimentation to suit the requirements of different robotics applications. We intend to advance the knowledge on the cloud robotics paradigm and thus the specific objectives of this work can be listed as:

- To perform a critical analysis of the state-of-the-art in cloud robotics, identifying the requirements, applications, and open questions;
- To assess the requirements of cloud robotics applications involving mobile robots and human-robot interaction;
- To study the interplay of the cloud robotics paradigm and human-robot interaction by relating the system's quality of service to the user's perceived quality of experience;

- To design a system architecture for cloud robotics including the network as an active part of the system;
- To develop an open-source framework to enable robot-cloud communication;
- To implement cloud-based real-time services for different applications;
- To develop a reproducible methodology for cloud robotics implementation based on open-source software;
- To validate the developed communication framework and implementation methodology considering different robotic systems and configurations.

1.4 Contributions

This thesis investigates aspects of the cloud robotics paradigm mainly linked with three research topics, namely cloud-aided HRI, networking and wireless communication, and robot-cloud communication, to enable cloud-based robotics applications. Although related, each of these topics presents challenges that were not previously addressed in the literature or were not resolved by previous works.

One of the contributions of this thesis is to present a holistic view of how cloud robotics systems interacting with a person rely on acceptable QoS and to relate such performance with the user's perceived QoE. We perform a pilot study as the first step towards a better understanding of the interplay of QoS and QoE in cloud-aided human-in-the-loop systems. The outcomes of our pilot study allow us to establish proper latency requirements for suitable HRI, thus advancing the state-of-the-art in the field.

Although latency and other QoS metrics can be mitigated by design and implementation choices, uninterrupted wireless connectivity remains a key issue when considering mobile robots. Another contribution of this thesis is to leverage advanced networking techniques to construct a cloud robotics architecture encompassing all levels of the system to satisfy connectivity requirements. We expand previous work on WiFi and SDN to implement extended wireless coverage and uninterrupted connectivity in mobile robots. This allows for uninterrupted service from cloud-based applications on WiFi-enabled systems even during access point migration.

After advancing knowledge regarding some of the key cloud robotics' requirements and enablers, we deal with implementation aspects to present a reproducible solution to cloud robotics. Thus, the main contribution of this thesis is a novel methodology to integrate cloud robotics' composing subsystems based on open-source software and readily-available equipment. To this end, we address the lack of standard solutions for robot-cloud communication by presenting an open framework based on the ROS. Our communication framework should foster cloud robotics research by easing robot-cloud

communication upon a widespread open-source robotic middleware, allowing for a wide range of applications to benefit from easy-to-construct cloud-based functionality. Moreover, our methodology allows for clear and comparable implementations, in such a way as to not rely on black boxes and thus enable researchers and practitioners to validate the results of others.

1.5 Publications

During the realization of this thesis, the following papers were published and are directly linked to the work here presented:

- **Publication A: MELLO, R.**; JIMENEZ, M.; RIBEIRO, M. R. N. Ribeiro; GUIMARÃES, R. L.; FRIZERA-NETO, A. On Human-in-the-Loop CPS in Healthcare: A Cloud-Enabled Mobility Assistance Service. *Robotica*, v. 37, p. 1477–1493, 2019.

- **Publication B: MELLO, R.**; SIERRA, S. D.; MÚNERA, M.; CIFUENTES, C. A.; RIBEIRO, M. R. N.; FRIZERA-NETO, A. Cloud Robotics Experimentation Testbeds: a Cloud-Based Navigation Case Study. In: *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*, IEEE, Medelín, 2019.

- **Publication C: MELLO, R.**; JIMENEZ, M.; RIBEIRO, M. R. N.; FRIZERA-NETO, A. Towards a New Generation of Smart Devices for Mobility Assistance: CloudWalker, a Cloud-Enabled Cyber-Physical System. In: *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, IEEE, Enschede, 2018, pp. 439–444.

Moreover, this thesis also led to the submission of the following paper, currently under review:

- **Publication D: MELLO, R.**; SIERRA, S. D.; MÚNERA, M.; CIFUENTES, C. A.; RIBEIRO, M. R. N.; FRIZERA-NETO, A. The PoundCloud Framework for ROS-based Cloud Robotics: Case Studies on Autonomous Navigation and Human-Robot Interaction. *Robotics and Autonomous Systems*, 2020 *(submitted).*

Lastly, the author participated in other research works related to this thesis; the main publications resulting from such activities are listed below:

- **Publication E:** JIMENEZ, M.; **MELLO, R.**; BASTOS, T.; FRIZERA-NETO, A. Assistive Locomotion Device with Haptic Feedback For Guiding Visually Impaired People. *Medical Engineering & Physics*, v. 80, p. 18–25, 2020.

- **Publication F:** GUIMARAES, R.; MARTINEZ, V. G.; **MELLO, R.**; MAFIOLETTI, D.; MARTINELLO, M.; RIBEIRO, M. R. N. An SDN-NFV Orchestration for Reliable and Low Latency Mobility in Off-the-Shelf WiFi. In: *2020 IEEE International Conference on Communications (ICC)*, IEEE, Dublin, 2020.

- **Publication G:** ROCHA-JUNIOR, J. C.; **MELLO, R.**; BASTOS-FILHO, T.; FRIZERA-NETO, A. Development of Simulation Platform for Human-Robot-Environment Interface in the UFES CloudWalker. In: *2020 XXVII Congresso Brasileiro de Engenharia Biomédica (CBEB)*, SBEB, Vitória, 2020.

- **Publication H:** SCHEIDEGGER, W.; **MELLO, R.**; SIERRA, S. D.; MÚNERA, M.; CIFUENTES, C. A.; FRIZERA-NETO, A. A Novel Multimodal Cognitive Interaction for Walker-Assisted Rehabilitation Therapies. In: *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, IEEE, Toronto, 2019.

- **Publication I:** SCHEIDEGGER, W.; **MELLO, R.**; SIERRA, S. D.; MÚNERA, M.; CIFUENTES, C. A.; FRIZERA-NETO, A. A Novel Multimodal Human-Robot Interaction for Walker-Assisted Gait: Merging an Admittance Controller and a Deep Learning Approach. In: *Actas del X Congreso Iberoamericano de Tecnología de Apoyo a la Discapacidad*, Buenos Aires, 2019.

In Fig. 3, we map this thesis' chapters and our most relevant publications into their related research topics to position our work in the field. Note that some of this thesis' chapters are written taking as a basis the work from specific publications; Chapter 2 is related to Publication A and Chapters 4 and 5 are related to Publication D. We detail the thesis organization in the next section.

1.6 Thesis Overview

The remainder of this thesis is structured as follows. Chapter 2 addresses the human-in-the-loop effect in cloud-aided CPS with particular focus on assistive robots. We present a pilot study designed to investigate the interplay among QoS and QoE and discuss our findings to provide new insight into the theme.

Chapter 3 deals with the problem of uninterrupted indoor connectivity for mobile robots by proposing the use of SDN into cloud robotics architectures and exploring a networking technique to improve communication reliability.

Then, in Chapter 4 we present an open framework for robot-cloud communication based on the ROS. The operation of our framework is validated upon an emulated cloud robotics testbed to explore its capabilities and limitations.

Chapter 5 introduces and validates a methodology for cloud robotics experimentation based on open-source software and commercial off-the-shelf devices. Such a methodology

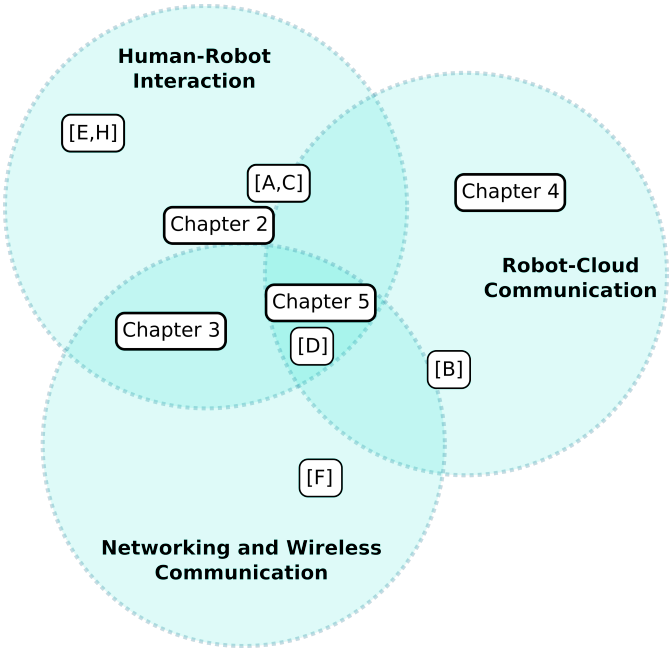


Figure 3 – Diagram connecting the main research topics related to our work with this thesis' chapters and our main publications.

includes the use of our framework for robot-cloud communication, which is also verified experimentally in a practical setup.

Finally, Chapter 6 summarizes our findings, presents our final remarks, and discuss future lines of research that arise from this thesis.

2 Requirements for Offloading Robots' Processing to the Cloud

The design of cloud robotics systems must carefully consider the requirements of network and cloud performance to maintain suitable levels of QoS for a given application. When offloading parts of the robot's processing to the cloud, system performance and safety become highly dependent on the capacity of network and cloud technologies to fulfill this set of requirements. In this chapter, we explore this issue within the eHealth field by considering cloud-enabled robots used for assisting human locomotion. This kind of system relies on close physical HRI and a change in paradigm towards cloud robotics faces multiple challenges.

Despite recent advancements in the field of cloud-enabled and human-in-the-loop CPS, there is still a need for systematically setting the requirements posed by the integration of these kinds of systems. This results in a lack of understanding of how infrastructure-related QoS issues inherent to cloud robotics affect user-perceived QoE during HRI. Thus, we present a case study in which a mobility assistive robot – a smart walker – guides a user throughout corridors. We conduct a pilot study using this cloud-enabled robot to investigate the relationship between QoS and QoE in such a system.

2.1 Cloud-Enabled Cyber-Physical Systems in eHealth

Robotics concepts are being increasingly applied to healthcare – encompassing a field that is often called eHealth – and future medical facilities are likely to be equipped with devices such as robotic caretakers and autonomous stretchers, devices which will be also sharing space with patients and health professionals. In this context, some recently developed assistive robots (e.g., smart walkers [31] and smart wheelchairs [35]) can be classified into CPS as they combine sensing, communication, and control/computing to interact with a physical entity [36]. Such a class of healthcare robots, apart from most robotic systems, usually interact at physical and cognitive levels with patients to provide functionalities where reliability and safety are essential [37].

The exploitation of the cloud robotics paradigm can expand the capabilities and features offered by assistive CPS [38,39]. Migrating a robot's processing capabilities to the cloud leads to lighter and less expensive robots capable to access increased memory and processing resources [1,23,40]. Mobility assistive devices used in eHealth can also benefit from cloud robotics concepts [41], as cooperation and information sharing among robots are enhanced, allowing for improved interaction and navigation [42].

In this context, cloud-enabled CPS for patient mobility assistance faces challenges related to E2E QoS of a complex system combining communication networks and cloud computing [7]. Healthcare CPS are expected to require from the network uninterrupted wireless connectivity, throughput guarantee, and stringent latency and jitter features [43]. Another intricate problem arises when objective QoS metrics need be translated into indexes of QoE to the end-user [44]. When interacting with a robotic system during mobility assistance, the user is often in-the-loop from the control algorithms' point of view. As shown by Pons [45], stability issues arise in such human-in-the-loop systems; latency inserted into the control loop greatly impacts the overall stability, affecting both physical and cognitive interaction between a human and a robot.

Nevertheless, most of the works conducted upon human-in-the-loop CPS do not consider direct physical interaction between human and CPS [46]. Lee *et al.* [47] point the challenges and directions for the recent generation of medical CPS, while Shah *et al.* [7] state that approaches to QoS in healthcare services are scarce in the literature as this is an emerging field. Hammer *et al.* [13] consider that QoS needs to be carefully controlled in CPS and its impact on QoE need to be taken into account. The challenges for the control of human-in-the-loop CPS are listed by Munir *et al.* [12], who states that the understanding of the complete spectrum of human-in-the-loop control is needed as more sophisticated applications appear. Thus, a better understanding into QoS influence over QoE is needed not only in CPS for healthcare, but also in other systems heavily influenced by the human-in-the-loop effect such as car driver assistance [48] and wearable exoskeletons [9].

When considering sensitive applications, such as patient mobility assistance, cloud services cannot be provided on a best-effort basis, thus QoE is likely to impose the QoS requirements for those future network/cloud infrastructures and not the other way round. However, the interaction of cloud technologies, healthcare robots, and patients are yet to be fully understood, especially in human-in-the-loop CPS that explores the physical interaction between human and robot.

To allow for the so-called Healthcare as a Service (HaaS) to arise, eHealth may leverage the systematic use of multiple heterogeneous devices performing different roles. In such a system, cloud-enabled robotic devices will provide services for patients and healthcare professionals. Currently, cloud computing has been reported to empower several healthcare CPS in applications such as patient localization, monitoring and status recording [43,49–51], and construction of databases and ontologies [52, 53]. Cloud-enabled robotic nurses have been discussed in Fosch-Villaronga *et al.* [54] and there are also initiatives in social robotics to design cloud-enabled companion and caregiver robots for the elderly, as in Fiorini *et al.* [55], leveraging the cloud capabilities to expand the current state-of-the-art. Regarding robotic devices employed for assistance and rehabilitation, cloud-enabled applications have been used to collect data from exoskeletons [56] and to allow on-line remote monitoring of therapy [57].

Assistive mobility devices are still beginning to benefit from cloud computing concepts. Some works have explored cloud-based services being provided to smart wheelchairs, as in Fu *et al.* [58] and Salhi *et al.* [59]. Regarding smart walkers, the ANG-MED robot [41] seems to be the first initiative in the literature to explore cloud-based services. The ANG-MED is a passive walker (i.e., without motors on the wheels) that uses cloud-based applications to provide authentication, database, and brake and motion control services for the caregiver; Tsardoulias *et al.* [41] state that the viability of such applications was verified during experimentation. Another initiative to push smart walkers' integration with the cloud can be found in our previous works with the CloudWalker [60, 61]. Our preliminary work verified the feasibility of exploring cloud robotics concepts to provide real-time control functionality to smart walkers.

To unleash a generation of truly cloud-enabled mobility assistive devices, some challenges and limitations must be overcome. Assistive robots in general can benefit in multiple ways from this new ecosystem composed of network infrastructure and cloud computing. For instance, navigation systems can largely benefit from cloud computing as they often require algorithms with high computation costs, such as laser or vision-based Simultaneous Localization and Mapping (SLAM) [62,63]. Applications on cloud platforms might leverage multiple robots' inputs to construct and update global maps dynamically, even pointing to more crowded areas to be avoided by path planning algorithms [42]. The cloud also has the potential to empower HRI by enabling improved human intent recognition and shared control. This could potentially lead to devices that can continuously cope with users as gait conditions change over the years.

Finally, the design of general networked systems for medical use must take into account various aspects such as the security of the system and controlled access to patient data. Privacy issues arise during the collection, transmission, processing, and storage of patient data and there are strict guidelines to be followed. The European Union regulates the use of patient data in its General Data Protection Regulation [64], whereas the United States' Food and Drug Administration regulates software designed for handheld medical devices [65]. As cloud robotics concepts become more common, one might expect the release of specific regulations in the near future.

2.2 Requirements, Current Limitations, and Challenges

Communication technologies play a fundamental role in future healthcare data transmission, storage, processing, and eventual feedback in real-time. Nevertheless, network and cloud-related constraints – and resulting QoS – have a direct impact on cloud-based applications. For instance, cloud services and the Internet itself are designed to work on a best-effort basis, leading to unpredictable behaviors unsuitable for cloud robotics [66]. Thus, as phrased by Li *et al.* [66], "*directly combining cloud services with robotic applications*

may result in unacceptable consequences."

In this thesis, we consider five general requirements in a cloud robotics system: high service availability; uninterrupted network connectivity; high throughput guarantee; low packet loss, and; low latency and jitter. Definitions regarding what characterizes high – or low – figures in each of these requirements are application-dependant and demand care to balance realistic performance targets and actual system requirements.

Applications making use of cloud-based services depend on both service availability and network availability, which also includes wireless connectivity in mobile robots. Given the current state of cloud computing technology and platform providers, uninterrupted cloud – and consequently service – availability is a requirement often fulfilled. Nevertheless, the life-cycle managing of robotic services and the real-time characteristic of this type of service is a challenge that must be addressed by cloud providers.

When looking into network availability, wireless technologies are a fundamental building block in mobile robotics' applications, and 5G technology will be a key enabler for providing connectivity in mobile real-time applications. For instance, the goals for critical applications in 5G are to provide latency figures as strict as 1 ms alongside 10^{-5} unavailability ratios [67], which would certainly meet the stringent requirements set by human-in-the-loop CPS. Nevertheless, 5G standards do not deepen the discussion regarding how such requirements were evaluated, and thus the requirements defined by 5G should be considered as general guidelines and not actual requirements for practical systems.

Moreover, current off-the-shelf technologies such as WiFi have been demonstrated to support connectivity in mobile applications [19]. While WiFi offers high throughput features, it also suffers from crucial issues as long delays for client association and high latency handover procedures [19]. During handover procedures (i.e., when a WiFi client migrates from one access point to another), connectivity is lost for a few seconds; Martinez *et al.* [19] illustrates this loss of connectivity in a simple experiment and we reproduce their results in Fig. 4. Considering indoor cloud robotics applications such as assisting mobility throughout healthcare facilities, the ubiquitous use of WiFi places it as a viable solution. Nevertheless, control for critical applications must be designed taking into account possible reliability issues in WiFi.

Current research points that to mitigate such issues, edge cloud architectures can help providing stringent latency and jitter [68] while SDN is expected to provide a uniform mechanism for E2E network programmability [21]. Concerns have been raised about SDN's ability in low-latency reconfiguration operations toward the core of the networks [69]. Nevertheless, the use of SDN may also address the lack of reliability in WiFi, as shown by Martinez *et al.* [19], enabling the use of such technology for cloud robotics. There is still a lack of studies addressing such technologies as enablers of human-in-the-loop CPS applications; we further address this topic in Chapter 3.

To guarantee user safety and comfort while using an eHealth CPS, latency should

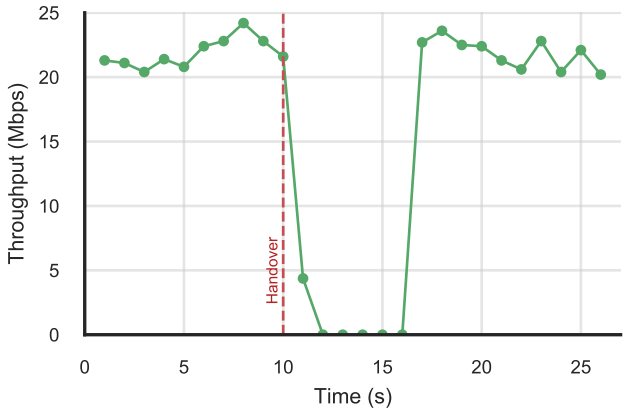


Figure 4 – Typical change in throughput as perceived by a WiFi client during a handover procedure. The procedure starts after 10 s from the beginning of the experiment and the throughput is only fully reestablished at 17 s. Adapted from Martinez *et al.* [19].

be kept as low as possible. For instance, testing the exchange of 70 B packets from our university in Brazil to a major cloud provider on both US' coasts, an average latency of about 700 *ms* was observed, decreasing to 50 *ms* with their edge-like service hosted in São Paulo. Besides latency, jitter also influences performance by inserting information disorder in the feedback loop. Short-term events, such as congestion and errors over wireless links, lead to individual packet loss and then throughput reduction. There are also long-term failures that are related to availability metrics that lead to massive successive packet loss events. In practice, however, actual dynamics of use might tolerate such real-world cloud performance.

The stringent requirements considered by current standards such as the 5G calls for advanced technological solutions; the lack of generality and available equipment may hamper paradigm migration towards cloud robotics. Nevertheless, if more relaxed latency and packet loss requirements can be verified for cloud-enabled CPS, the use of remote commercial clouds over the public Internet becomes viable, easing the deployment of this kind of system. This has the potential to foster rapid cloud robotics adoption and we address this issue in the case study presented in this chapter.

Despite efforts from vendors and platform providers, cloud robotics also lacks standard tools and development frameworks. The wide adoption of the open-source ROS framework favors common practices among developers. Nevertheless, ROS was not designed to include robot-cloud communication nor communication over the public Internet. As closed-source solutions are included within platform providers' services, such as the AWS Robo Maker, advancements in the field are restricted by the lack of programmability of proprietary solutions. ROS may be used as a basis for novel frameworks linking robots and clouds; we discuss this subject in Chapter 4.

Considering the implementation of cloud robotics systems in general, there must be

carefull consideration into which funcionality must be provided by the embedded hardware and which can be delegated to remote platforms. Ideally, low level controllers and safety-related functionality should be executed within the robot's embedded hardware; mid-level controllers, such as navigation or grasp planning, should be instantiated either locally or in edge clouds, and; high-level controllers, such as the general planning for executing a complex task, could be performed in a remote cloud platform. As for the current state of cloud-based eHealth systems, particularly considering devices used in rehabilitation and assistance, solutions tend to employ small degrees of cloud computing over control. Figure 5 illustrates some of the healthcare-related works referenced in this thesis on a scale of how much of their control depends on cloud computing. The cloud robotics trend points towards migration to the cloud, or "cloudification", of solutions, especially those involving complex algorithms. The case study that will be presented in the next section illustrates an extreme case of this control "cloudification", in which there is no embedded intelligence, and is placed on the rightmost end of Fig. 5 spectrum. In any case, there is a timely need for understanding how communication constraints might impact such applications from an end-user point of view.

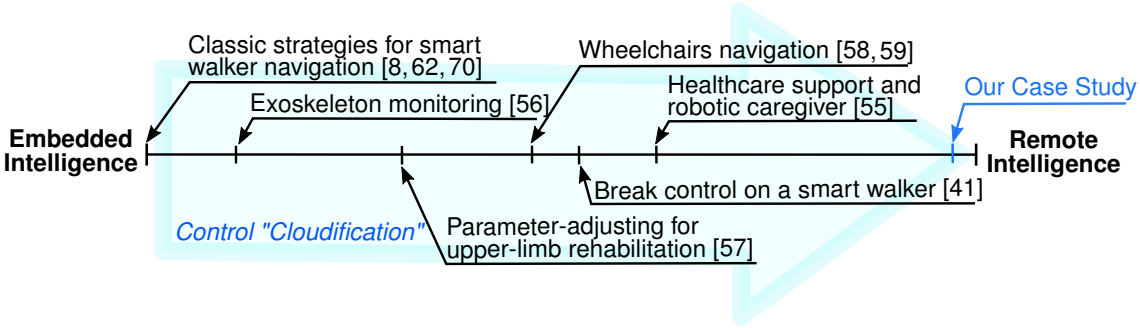


Figure 5 – Cloud-enabled CPS for healthcare: "cloudification" scale of current solutions.

Figure 6 considers three hypothetical cloud-based applications to illustrate the degree of importance of each of the five requirements previously listed. When analyzing how to make use of cloud-based functionality in robot-assisted surgery, most of the listed requirements are extremely important – the exception is the wireless connectivity since we consider that such robots would be connected via cable to the local network infrastructure. For instance, low latency features can enable the haptic interaction at the surgeon's joystick whereas a high throughput capacity can deal with the transmission of high-definition video from multiple cameras to assist the remotely-placed surgeon. Nevertheless, not all applications are as delicate as this one and a mobile telepresence robot used for patient triage at hospitals, for example, would tolerate poorer QoS levels. As a third example, we consider in Fig. 6 a mobility assistance application using a smart walker in the same configuration as discussed in the previous paragraph (i.e., device conceived with no local intelligence). In such an application, service availability and wireless connectivity become crucially important for the proper performance of the system, since all computation is

performed remotely. While throughput features are closely related to the sensor data that must be transmitted to perform the control of the device, latency and packet loss have a direct impact on the HRI and the user's perception of the device.

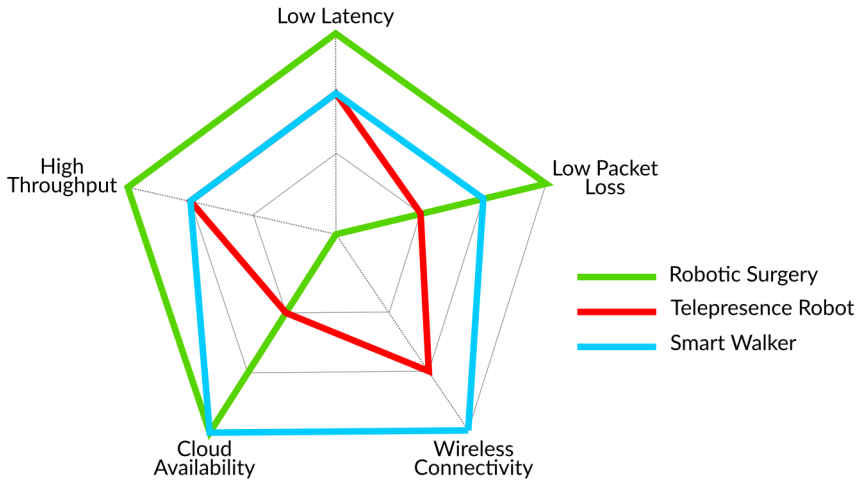


Figure 6 – Diagram illustrating the degree of importance of network and cloud requirements considering different classes of robots and applications.

The concept of QoE arises in the context of telecommunication applications, where both human senses and cognitive features are considered as part of the information processing [71]. Despite advances in QoE evaluation techniques, there are still challenges in properly weighing ease of use, comfort, and other subjective parameters into objective QoE metrics [72]. There are also challenges in mapping QoE into QoS requirements to the underlying network for every new application [44]. Although cloud-enabled CPS may use the previous QoS-QoE studies as step-stones, there are specific issues concerning healthcare applications; especially when the end-user is heavily embedded into the system dynamics.

Therefore, there might be trade-off solutions balancing network and cloud performance, CPS dynamics, and human physical and cognition features to define reasonable network requirements for cloud-enabled human-in-the-loop CPS. To investigate this topic, the next section presents an illustrative case study scenario in which a cloud-enabled CPS provides a mobility assistance service.

2.3 Case Study Scenario

We present a case study upon a scenario where hospitalized patients and nursing home inhabitants may benefit from the use of cloud-enabled assistive devices. Such a scenario is designed to illustrate a common mobility assistance service that may be provided by cloud-enabled assistive robots. Patients with impaired mobility or cognitive dysfunctions might need help during displacement across a clinic, therapy, and accommodations. Particularly in nursing home environments, assistive devices are required on a daily basis by many

permanently impaired residents and guiding features can further assist user displacement. Such a scenario is used as the basis in our pilot study, which will be later discussed.

Devices such as smart walkers can exploit emergent technologies to safely assist on navigation by guiding user displacement to the desired location. The device should be able to perform localization and to leverage mapping functionality to generate a path towards destination [31]. In cases where further assistance is necessary, such a path can be conceived as a virtual trail in which the user can walk along without being able to deviate from it, almost as if in a rail track. Figure 7 illustrates this scenario by displaying a person making use of a smart walker inside a facility with corridors and obstacles. The virtual trail established by the device marks the path to be followed and the walker slowly turns to keep on the trail as the user walks forward.

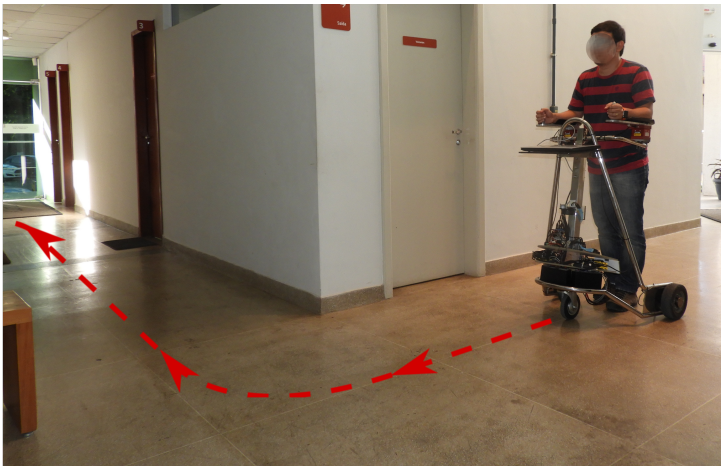


Figure 7 – Case study scenario: the smart walker guides user displacement inside a facility through automatically generated virtual trails. The interaction forces measured by the force sensors under the handlebars are used as inputs to control walker velocities.

This guiding feature can be used to assist the navigation of individuals with reduced independence and/or balance, as a consequence of vision and mobility impairments caused by cerebral diseases [73], cardiopulmonary and musculoskeletal diseases, or even stroke [74]. Furthermore, the haptic feedback indicates the trail to follow, a useful feature for those with orientation and localization problems. As interactions between patient and assistive CPS must be built on basis of navigation applications and coordination with other users, heavy control algorithms may be required, which can be instantiated in a centralized way over the cloud.

Considering the described scenario, in the following section, we present the smart walker used in our pilot study, the CloudWalker, and its architecture.

2.4 A Cloud-Enabled CPS for Mobility Assistance: the CloudWalker

The CloudWalker system architecture allows the use of smart walkers as cloud-enabled CPS, encompassing a system that involves physical and cognitive HRI aimed at mobility impaired individuals. In such a system, the user's motion intents are perceived by embedded sensors on a smart walker, which communicates with a cloud platform to provide information and also receive the appropriate control signals to physically command the robot.

The proposed architecture is described in Fig. 8. Physical interaction between the user's upper body and walker is measured by force sensors that capture the user's motion intent. Collected data is sent through the network to a remote cloud platform running the control applications. Data is processed by the control algorithms in the cloud, which sends back to the walker's embedded computer the generated control signals to command the actuators, resulting in the motion of the user-walker aggregate.

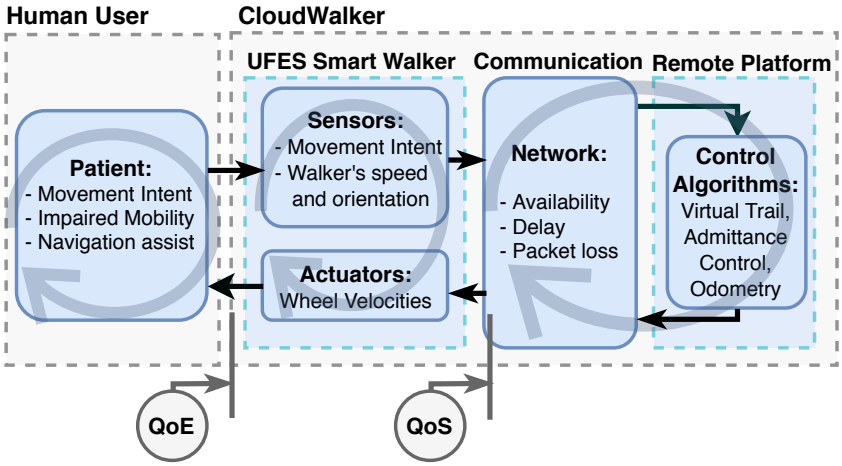


Figure 8 – CloudWalker architecture: main control loops and relation of perceived QoE and QoS.

Ideally, in this particular system, it is desirable to implement critical lower-level controllers on the device to maintain usability and stability even under situations where there is no network connectivity, delegating to the cloud only high-level computationally expensive tasks. As the main goal of this case study is to assess QoS impacts over the QoE on cloud-enabled human-in-the-loop CPS, the described architecture does not envision any local intelligence on the device. This inserts communication constraints directly into the control loop, an issue that has been thoroughly addressed in distributed networked control systems literature [75] and that, in our study, maximizes the impacts of QoS on system performance and user-perceived QoE. Therefore, safety measures are necessary during implementation to mitigate possible instability impacts. Furthermore, traditional networked control systems often consider latency when modeling the system to generate

robust controllers; modeling complex HRI to consider the insertion of latency into the system remains an open research field, further motivating the study presented in this chapter.

Our system is based on an in-house developed smart walker [8, 31]. The UFES Smart Walker (detailed on Fig. 9) is a robotic platform equipped with sensors and actuators designed to actively assist on patient's navigation throughout the environment. It can interact with the user in several ways, such as interpreting the physical interaction forces on the forearms supports and keeping track of the user's legs position to follow in front of the patient. It is also possible to track obstacles and even map the environment by deploying the frontal laser scanner. An embedded computer centralizes the acquisition of sensor data, communication with the network, and the forwarding of received control signals to wheel actuators. This application is integrated into the Matlab-Simulink Real-Time xPC Target architecture and for the case study, only the force sensors were used to detect the patient's intentions to move. For such an application, communication is based on the exchange of 40 bytes UDP/IP packets between the walker and the remote platform.

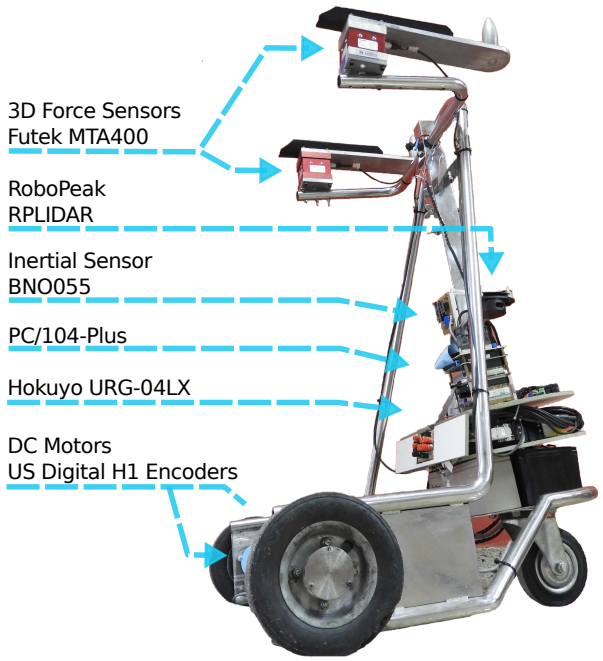


Figure 9 – CloudWalker system: the UFES Smart Walker in detail.

As depicted in Fig. 8, CloudWalker architecture encompasses three control loops:

- A local control loop, where motor dynamics must be taken into account during the system's response.
- A global control loop, with robot's sensors feeding remotely allocated controllers to command actuators accordingly to the patient's intents captured by the force sensor located on the support frame.

- A cognitive loop, where the patient reacts to the smart walker movements by instinctively changing forces applied to the support frame.

The local control loop, related to the walker dynamics, is the faster one, and a time constant is empirically obtained around 100 ms. Thus, the embedded electronics sampling time is set at 10 ms, 1/10 of the verified time constant of the system, as recommended by Dorf [76]. At every sample interval, data from sensors is sent through the network to the remote platform, which in turn sends back the processed control information containing the target velocities at that moment. Therefore, the round-trip time for network/cloud processing time should ideally be below our sampling window. Control information delayed beyond this sampling window directly impacts not only the global control loop but also the user's cognitive loop. This leads to control degradation - and even complete loss of controllability in extreme cases - and may mislead the human-CPS interaction at the risk of taking the whole system to an unstable state.

2.5 Pilot Study

To assess how network and cloud parameters may affect cloud-enabled human-in-the-loop CPS, we designed a pilot study to investigate the user's perceived QoE under variable QoS. Therefore, we test the CloudWalker system under different QoS conditions to subjectively evaluate its performance from the end-user point of view. We present a use case that, despite its deliberate simplicity, is strong enough to illustrate a mobility assistance service. The selection of controllers and the fact that there is no computation performed on the device itself is intended to push the system to a QoS-sensitive configuration. This pilot study expands a preliminary experiment performed by us, which was presented in Mello's Master's dissertation [61] and in Mello *et al.* [32]. Before presenting our pilot study, we first briefly discuss this preliminary experiment to address previous findings and thus contextualize our pilot study.

The preliminary experiment presented in the Master's dissertation of Mello [61] was designed as an exploratory approach aiming at understanding how users react to multiple QoS metrics. The same scenario and guiding service as described in Section 2.3 were employed and the CloudWalker system was tested by emulating the insertion of communication latency, poor availability and connectivity, and insertion of packet loss into the control loop. Twelve participants performed five tests each under different QoS conditions, evaluating their experience using ratings from 1 to 5. To illustrate some of the results from this preliminary experiment, we adapt Figure 10 from Mello [61]. In Fig. 10(a), we present odometry plots from tests performed by the same person; this participant better evaluated the test performed under lower round-trip delay although the test under higher delay presented fewer deviations from the reference path. Figure 10(b) the complexity of assessing QoS effects on user and system overall performance. Contrary

to the high RTT realization presented in Fig. 10(a), the realization represented by the green line in 10(b) – under the same 500 *ms* RTT – was not completed. The complexities of HRI under latency resulted in the system missing out the trail, leading to serious safety concerns. In contrast, Fig. 10(b) also presents a realization under poor connectivity and yet successfully performed. An in-depth discussion regarding these results can be found in Mello [61].

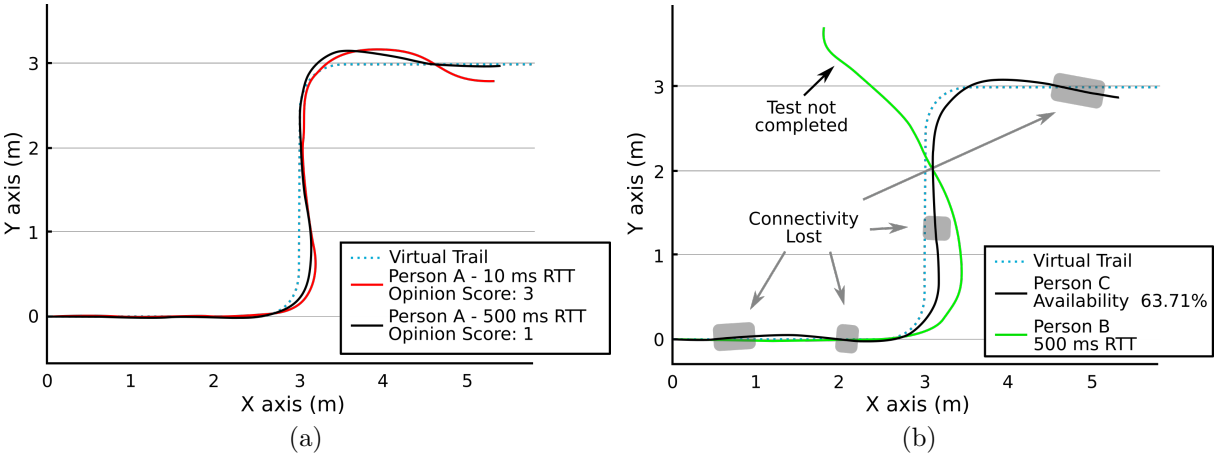
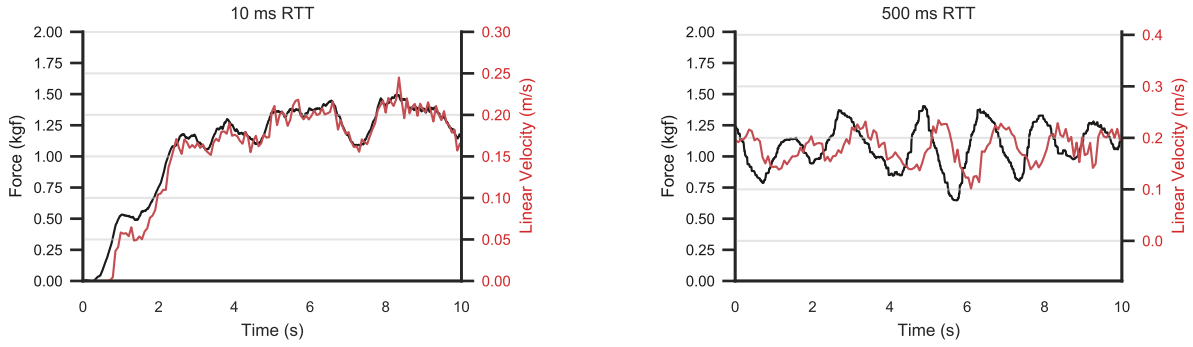


Figure 10 – Results of the preliminary experiments presented by Mello [61]: a) performed path and ratings under different RTT conditions, and; b) comparing two tests, one considering only high RTT (green line) and the other one under frequent connectivity loss (black line); grey areas mark periods of 2 s or more without availability. Adapted from Mello [61].

The preliminary experiment from Mello [61] provided us with valuable insight regarding the perception of the effects imposed by different QoS conditions into our system. When dealing with availability and connectivity, the implications are clear: high periods of an unresponsive state may compromise the execution of the task and put the user at risk. Thus, we chose to deal with the connectivity requirement separately from the QoE and further address it in Chapter 3. Furthermore, when considering packet loss as a QoS metric, no significant effects could be observed even under high loss probabilities (e.g., 30 %), likely due to the slow dynamics of a smart walker system.

The results of this preliminary experiment also indicated that latency is the major concern in human-in-the-loop systems due to its complex effects upon HRI. Figure 11 is extracted from Mello [61] to illustrate the effects of latency upon interaction. Figure 11(a) displays a sample of one test performed under 10 *ms* RTT, using dual-axis to overlay the interaction force exerted upon the walker and the resulting walker velocity; it is possible to see that the walker responds almost instantly to the variation in the interaction force. Contrastingly, Fig. 11(b) displays a sample of one test performed by the same person but under 500 *ms* RTT; the effects of latency are clear, as the walker takes some time to respond to the interaction force.



(a) (b)

Figure 11 – Results of the preliminary experiments presented by Mello [61], 10 s samples from the same participant using the walker under different RTT conditions. The double-axis in each figure indicate interaction forces between person and walker, as measured by the force sensors, and the resulting walker's linear velocity. Adapted from Mello [61].

However, the low number of participants and test realizations were factors limiting statistical analysis, which prevented us to draw more solid conclusions from our preliminary experiment [61]. Therefore, we designed the case study presented in this section to overcome the limitations of our previous experiment, aiming at reaching more solid data linking user-perceived QoE and variable QoS. To this end, we focus on the effects of latency as the main factor affecting both QoS and QoE. A total of 21 individuals participated in our pilot study and provided feedback about their perceptions while using the system. The following subsection details the implementation of the assistance service. The rest of this section describes the experiment, adopted protocols, and obtained results.

2.5.1 HaaS: Virtual Trails for Mobility Assistance

Here we focus on the pivotal role of wayfinding in healthcare facilities, an often-overlooked factor that can affect patient stress and is of special importance to mobility and cognitively impaired individuals [77]. We implemented an interaction-based guiding service that leads users in healthcare facilities through predetermined virtual trails. This service gives users the freedom to walk at their own pace and full control over the smart walker's speed. At the same time, the controller commands the turning speed to keep the user on a trail. This works on a proportional control strategy: the slower the person walks with the device, the slower the robot changes its orientation. The effect is of a virtual trail, where users can "push" the system along the virtual trail while preventing them to deviate from it.

Figure 12 provides a block diagram illustrating the three different control functionality that composes such mobility assistance service: an odometry algorithm, a path following

controller, and an admittance-based controller. Figure 12 also illustrates the control inputs (i.e., the interaction forces as measured by the force sensors) and how the – emulated – network constraints are inserted in the control loop in our implementation.

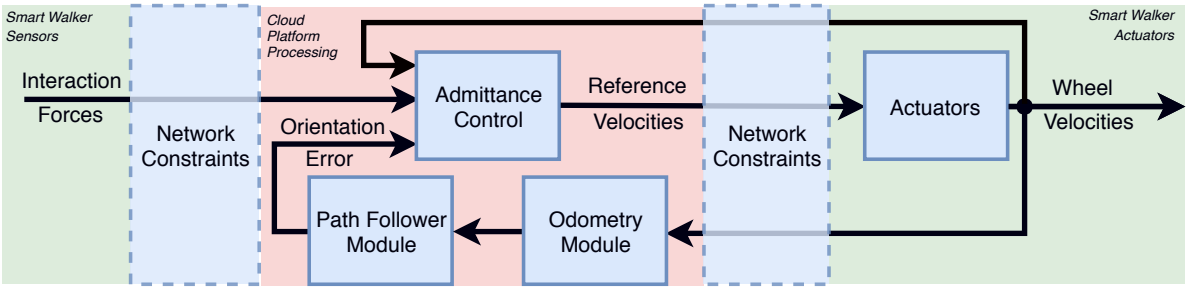


Figure 12 – Block diagram illustrating the control architecture of our cloud-based service for mobility assistance. The interaction forces and wheel velocities are sent from the walker to the cloud through the network and the processed control reference velocities are then sent back to the walker in response.

The main HRI channel is based on the haptic interface for the upper limbs. The forward force, $F(t)$, exerted by the user over the walker are measured on the forearms platforms and can be defined as:

$$F(t) = \frac{F_l(t) + F_r(t)}{2}, \tag{2.1}$$

where $F_l(t)$, $F_r(t)$ are the forward forces measured by the left and right arms force sensors. The forward force, $F(t)$, signals are filtered to remove high-frequency noise and this equation represents the detection of the user's motion intention.

The human intention feeds the *Admittance Control* block, which was proposed in [78] and is responsible for generating reference linear $v_c(t)$ and angular $\omega_c(t)$ velocities to command the walker. The reference linear velocity $v_c(t)$ can be defined as:

$$v_c(t) = \frac{F(t) - m_v\dot{v}(t)}{d_v}, \tag{2.2}$$

where m_v, d_v are mass and dumping parameters, respectively, used to tune the HRI dynamics, and $\dot{v}(t)$ is the linear acceleration of the smart walker. The reference angular velocity $\omega_c(t)$ can be defined as:

$$\omega_c(t) = \frac{\tau_v(t) - m_\omega\dot{\omega}(t)}{d_\omega}, \tag{2.3}$$

where m_ω, d_ω are mass and dumping parameters, respectively, $\dot{\omega}(t)$ is the angular acceleration of the smart walker, and $\tau_v(t)$ is the modified torque, which is the component responsible for the haptic guidance throughout the path. The modified torque $\tau_v(t)$ depends on the orientation error $\tilde{\theta}$ between the smart walker and the desired path. Such orientation

error is the output of the *Path Follower Module*, defined in [79]. Within the *Admittance Control*, the modified torque $\tau_v(t)$ is defined as:

$$\tau_v(t) = k_\tau v_c(t) d \tanh \tilde{\theta}, \tag{2.4}$$

where k_τ is a constant used to properly bind the modified torque to the forward forces, as to avoid disproportionality among the resultant linear and angular velocities, and d is the distance separating both force sensors. Thus, as the modified torque is generated to correct the orientation error, the resulting reference angular velocity maintains the user over the desired path. Finally, the odometry information feeds the *Path Follower Module* and closes the control loop, as seen in Fig. 12.

If no up-to-date control information is received at a given sample interval, the last valid control signal is repeated. As stability issues may arise not only from the inherent characteristics of a human-in-the-loop system, but also from communication constraints, we limited forward linear velocity and angular velocity to 0.25 m/s and 0.5 rad/s, which is compatible with the literature [31]. Since it is intuitive for smart walkers' users to pull the robot towards them to diminish velocity or during periods of gait instability, such reactions must be taken into account. Thus, in case negative values are generated for the linear velocity control signal, the controller saturates the linear velocity to zero, stopping the robot and inhibiting backward movements. This also restricts intense responses on the physical interaction with the human during low QoS settings, safeguarding the subjects of our experiment. Moreover, backward movements may lead to collisions involving the user and the robot and should be generally avoided.

In this pilot study, the same path is used in all tests with our virtual trail, consisting of straight lines interspersed by smooth curves to both sides (see Fig. 13). Such navigation path emulates common indoor navigation scenarios, as displacement through corridors and halls, which are likely to happen in hospitals and nursing homes.

2.5.2 Experiment Protocol

The only independent variable used is the total communication latency (i.e., RTT). As dependent variables, on which we wanted to measure the effect of the latency, we considered the users' QoE, the perception of control and the perception of safety when using the CloudWalker. The following RTT conditions are defined: 0 ms, 100 ms, 300 ms and 500 ms.

We recruited twenty-one participants to take part in the experiment (five women). All subjects are associated with our university and presented no disabilities. Only five participants had prior experience using a smart walker. Participants' ages ranged from 24 to 56 years old, their weights from 62 to 91 kg, and their heights from 1.63 to 1.88 m.

The volunteers received no compensation and we obtained the consent of all of them to collaborate with the study.

Two researchers conducted the experiment in our university for over 2 weeks. Each session lasted, on average, about 30 minutes and consisted of a briefing session, testing using the CloudWalker, and completion of post-experiment questionnaires after each test. No further instructions on how to interact with the device, designated path to follow, or details of the test conditions were provided. Participants were also encouraged to speak freely about their interaction with the CPS. We delineated that after using the system under a given RTT condition, each participant would answer 5-point Likert scale questions [80] for the dependent variables. The three questions composing the questionnaire are listed:

1. How do you rate the experience of using the smart walker?
2. How do you rate the feeling of control upon the smart walker?
3. How do you rate the feeling of safety when using the smart walker?

We perform a statistical analysis – presented in Appendix A – over the participants' responses to the questionnaire to evaluate the results of our pilot experiment. When evaluating the user-perceived QoE in the first question, the 5-point Likert scale is similar to the Mean Opinion Score (MOS) metric. The MOS is a widespread QoE metric that considers the average of scores given by each user regarding their experience using the system under a specific QoS condition. The concept of QoE attempts to combine user perception, experience, and expectation, and the MOS provides a simple evaluation method, in which the user ranks their experience based on a given scale [81]. Despite being one of the most popular metrics to measure QoE in applications such as media streaming, the MOS - and MOS-like metrics - have also been used to evaluate QoE in other human-in-the-loop applications such as robot teleoperation [82, 83].

Due to the number of participants, we chose a within-group design [80], in which each participant was exposed to all test conditions (i.e., use the CloudWalker under all the RTT conditions). This approach allows us to effectively isolate individual differences and reduce the noise level among participants; however, it is hard to control the learning effect and impact of fatigue. To minimize the learning effect, we randomly determined the order of the test conditions using a "Latin Square Design" and to mitigate the effect of fatigue, we kept the path relatively short with 9 m in total.

2.6 Results

Figure 13 illustrates four snapshots of a realization of the experiment. Although participants had no previous training, all of them figured out how to interact with the

walker and managed to successfully follow the unknown predetermined virtual trail. The comments made by the participants during the experiments are discussed in the next section.

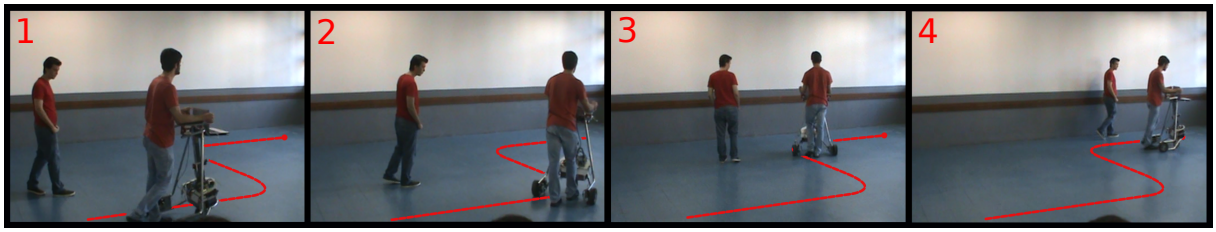


Figure 13 – Snapshots from the recorded video illustrating the virtual trail and following.

Figure 14 displays the distribution of questionnaire answers. We perform a statistical analysis to extract information from results – which is detailed in Appendix A – and report our findings in this section. In our statistical analysis, we hypothesize that there are no statistically significant differences in the answers reported under different RTT conditions and proceed to verify this.

Figure 14(a) summarizes participant's answers to the perceived QoE. The mean rating ranged from $\mu_{QoE\_RTT\_500ms} = 3.67$ in the higher RTT condition to $\mu_{QoE\_RTT\_0ms} = 4.29$ without latency insertion. There is a statistically significant difference between the rating distributions related to the RTT values of 0 ms and 500 ms, indicating that there are differences in the user's reported QoE under those two test conditions; such a result point to QoE degradation under the worst QoS condition. Furthermore, there are no statistically significant differences in any other pair of distributions.

Our results indicate the degradation of QoE with the increase of RTT, which is a reasonably expected conclusion. The insertion of latency in the control loop can lead most systems to an unstable state and it should not be different in human-in-the-loop systems given the insertion of sufficient latency. Nevertheless, it is not clear in the literature which RTT values may cause instability in such systems nor how much the user's QoE degrades even if the system is maintained in a stable state. Our results indicate that smooth QoS degradation leads to smooth QoE degradation, making it hard to pinpoint specifically when the user perception changes. Nevertheless, the absence of a statistically significant difference when considering 0 ms and 300 ms RTT conditions may be used as a guideline when setting the requirements of this kind of system.

Figures 14(b) and 14(c) display the rating distributions observed when evaluating users' perception of control and safety, respectively. The distributions are visually similar and the statistical analysis did not find significant differences in any comparison. Such results coupled with users' comments suggest that no degradation in safety and control was perceived by the users.

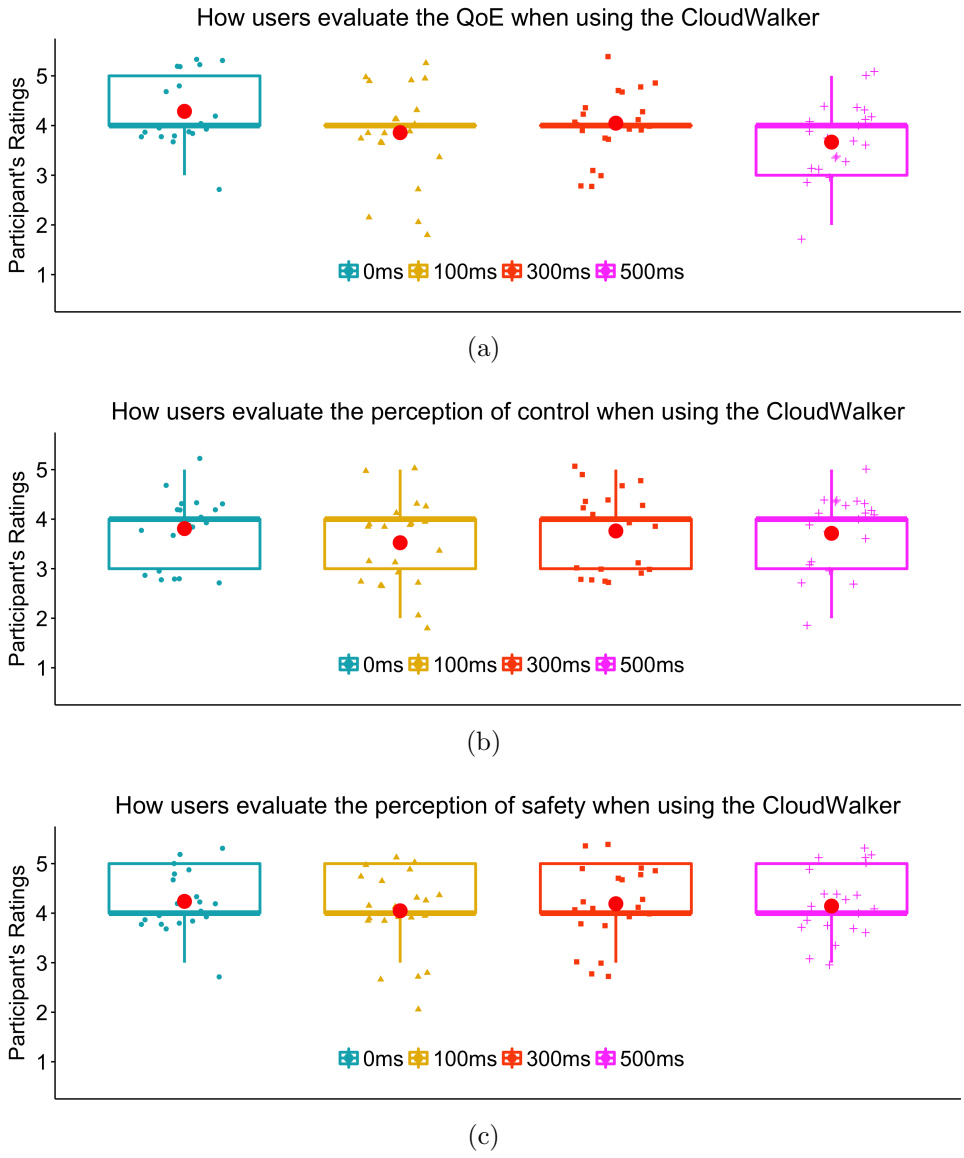


Figure 14 – Pilot study, boxplot diagram that summarizes the observations made. A random noise has been added to the data to make it easier to observe the distribution patterns. The red circles represent the mean values of the observed variables for the different RTT conditions: a) answers to Question 1, regarding the experience of using the system; b) answers to Question 2, regarding the feeling of control upon the system; c) answers to Question 3, regarding the feeling of safety when using the system.

2.7 Discussion

Most users followed the assigned virtual trail correctly with no prior information and under QoS conditions far worse than the ones envisioned by future 5G networks (i.e., latency smaller than 1 ms). Our outcomes unveil trends for relating QoS indexes and human-in-the-loop physical interactions with control loops affecting QoE degradation. The positive evaluations of the system even under reasonably high latency conditions indicate that it should possible to implement this kind of system communicating over

the Internet. When taking into account safety concerns, it is clear that our pilot study was based on an extreme example and practical implementations of a cloud-based system for mobility assistance should instantiate low level controllers in the robot's embedded hardware, delegating to the cloud higher level functions such as calculating the desired path, to give an example. Since such an implementation would diminish the effects of QoS over the user's perceived QoE, our results indicate the feasibility of employing cloud robotics concepts into assistive devices.

As our previous studies indicate [61], latency seems to be the major cause of lower QoE and also raised safety concerns as controllability might degrade to unstable conditions. Contrary to what we have previously observed, there were no occurrences of instability during our pilot experiment. This suggests that measures such as removing velocity restrictions and working with poorer QoS conditions might be necessary for future studies that intend to push such a system to extreme situations. Contrastingly, there is also a need for studies considering more complex – and realistic – QoS scenarios. In that case, we recommend longer-lasting realizations and extra instructions to participants due to potential safety concerns. Prior experience over benchmark conditions can also be tried for a comparative QoE evaluation.

In previous studies [61], we noticed that the analysis of MOS scores alone did not reflect entirely the user experience. For instance, one participant of the previous study said "this test is better than the previous one" and then gave the same score in both tests. From the feedback obtained in the experiments presented in this chapter, we corroborate the notion that comments like "this is the best one", "smooth", and "comfortable" are usually linked to the highest scores whereas negative comments are not necessarily followed by the lowest score. Positive comments could also be linked to higher experience scores in our pilot study. We also observed that user's comments reflected their scores regarding the sensation of safety and of being in control. Most users stated that the different QoS conditions did not impact on such perception and thus graded most of the test conditions similarly.

Here, we identified a statistically significant difference between the means of reported QoE for different RTT groups. As expected, among the RTT conditions we used, the 500 ms was considered the worst. Nevertheless, as even the tests under such high RTT conditions were completed, it may be realistic to assume that this kind of system would work even if the robot and the cloud are in distant countries. In Chapter 5, we present experiments considering cloud-based HRI and physically distant remote clouds. From the results of our pilot experiment, it is important to observe that as the latency of the network increases, the perception of control and the perception of safety does not seem to degrade as fast as the QoE.

Although these results indicate interesting findings, it is important to note that despite the effect of the independent variable (i.e., RTT) on the dependent variables, it

does not mean that we can generalize the conclusions to any RTT condition and any population. This is because we fixed beforehand what would be the RTT values used in the experiment [84]. The study also had some limitations. The group of participants can be considered homogeneous in terms of age, level of education, and physical condition. Therefore, one should keep in mind that our results should not be immediately generalized and the impact of other factors should be further investigated in future work.

Other assistive devices could be conceived as cloud-enabled CPS, each of them presenting particular characteristics and needs. Our study was performed over a guiding feature that can be employed by devices such as smart wheelchairs and canes, which are also affected by the human-in-the-loop effect. Efforts are needed in the direction of finding more suitable ways for the evaluation of cloud-enabled CPS healthcare applications, and also in establishing trustworthy means for relating QoS to QoE. Future studies should address some of the limitations found in our pilot study and eHealth providers will need to establish clear relationships between QoS and end-user acceptability.

User acceptance could be strongly affected whether the effects of QoS over QoE are not properly understood and mitigated in any human-in-the-loop CPS that rely on cloud-based services. As different service configurations are developed, each of them delegating part or even the totality of computational tasks to the cloud, more than addressing safety risks, there is a need to holistically evaluate the system including the end-users' point of view. The results from our pilot study corroborate this view, also pointing that user opinion alone may not reflect the experience. Despite our efforts to understand how safety and the feeling of control can be affected by QoS, there is still a need for deeper investigation in the direction of such a holistic evaluation.

It is especially difficult to compare our results to prior works regarding smart walkers or even other assistive devices due to the lack of standardization in methodology [85]. Works such as Wachaja *et al.* [62] and Werner *et al.* [86] presented objective and subjective metrics to evaluate guidance features in smart walkers, locally executing the control algorithms and focusing on the end-user point of view. Upcoming studies involving human-in-the-loop CPS could benefit from our results as stepping stones for new implementations.

2.8 Conclusions

In this chapter, we investigated the potential of emerging communication/computation technologies to unleash a new generation of healthcare assistive devices. To motivate discussions about the human-in-the-loop effect over a concrete case, we envisaged and implemented a pilot study for a healthcare service based on a virtual trail mobility assistance using the CloudWalker, a cloud-enabled smart walker. Upon this illustrative eHealth use case, challenges for migrating the control of a smart walker to cloud computing platforms could be discussed regarding QoS requirements, the human-in-the-loop effect,

and the perceived QoE.

Although the CPS system proved to be resilient to poor QoS, in our pilot study latency could be linked with lower QoE. Thus, despite the limitations of our pilot study, we conclude that the actual latency requirements for cloud-enabled human-in-the-loop CPS are far less restrictive than those envisioned by Industry 4.0 and 5G, to cite a few examples. Such requirements point to the feasibility of leveraging the cloud robotics paradigm in systems highly dependant on HRI. In the next chapter, we address another key requirement to enable the use of cloud-aided mobile robotics systems: uninterrupted network connectivity.

3 Indoor Wireless Connectivity for Mobile Robots

In this chapter, we resume the discussion regarding the networking requirements of cloud robotics with a particular focus on the connectivity problem. We discuss enabling technologies for cloud robotics and eHealth applications and how they can be used to fulfill the networking requirements of this kind of system. When considering the use of robots indoors, besides guaranteeing acceptable QoS, assuring continuous wireless connectivity is of fundamental importance for mobile robots. We present a novel cloud robotics architecture for mobile robots exploring the SDN paradigm to seamlessly integrate robots and network infrastructure. Then, we present a communication solution under such architecture. Our solution leverages common off-the-shelf devices and WiFi to eliminate loss of connectivity during wireless access point migration. To illustrate the use of the present architecture and communication solution, we discuss the re-design of the virtual trail service presented in Chapter 2 into an actual cloud service and perform a set of experiments to validate our architecture and communication solution. We end this chapter with a brief discussion regarding the proposed architecture and the implemented system.

3.1 Enabling Technologies for Cloud Robotics in eHealth

The adoption of cloud robotics in critical applications and verticals such as eHealth will certainly face communication challenges, related to uninterrupted connectivity and E2E QoS [6, 87]. Human-in-the-loop factors also play an important role as many healthcare devices must interact directly with humans, some of them even at the physical or biological level [9]. Moreover, the unordered nature of communication within clinics and hospitals implies that mobile robots are closely interacting with humans and other (intelligent or not) robots with heterogeneous priorities. The centralizing architecture of cloud computing may, on the other hand, facilitate pathfinding and decision-making processes, when intelligent devices are assisting patients navigating in such a complex environment.

Particularly for mobility assistive devices used for rehabilitation or locomotion empowerment, special attention is to be spent on safety and reliability aspects, as such devices often interact at a physical level with impaired and frail individuals [9]. Therefore, the underlying communication network must allow for uninterrupted connectivity. This opens new avenues for service providers to develop cloud-based applications, not only for the eHealth vertical, but also for Industry 4.0 [16] and robot-enhanced households [88].

We claim that, meanwhile monitoring the evolution of wireless technologies [89] and waiting for the 5G technology to be mature enough to be also economically viable for

eHealth applications, one can make use of existing technologies to cope with the currently known requirements of clinical healthcare services. The cloud robotics paradigm can help to achieve an early adoption of robotics in eHealth, thanks to its relatively lower costs of deployment [88]. Moreover, for indoor applications, robotic solutions can rely on off-the-shelf regular WiFi hardware for lowering capital expenditure (CAPEX) and operational expenditure (OPEX).

In a scenario in which multiple mobile devices with different degrees of autonomy move around corridors and halls of healthcare facilities, such as the one explored in Chapter 2, the cloud's global view can generate appropriate trajectories to the robots [42]. Thus, logistics can benefit from the centralization of knowledge in the cloud to improve efficiency, avoid crowded zones, prioritize tasks, and better cope with emergencies [42].

To cope with the high dynamism of the use cases involving cloud-enabled mobile robots in eHealth, a reconfigurable network topology seems to be the best solution as the underneath architecture for cloud-robotic services. Defining in a suitable way such network topology represents one of the key challenges that have not yet been properly addressed. For instance, new routing protocols and more effective management of the mobile nodes are needed to guarantee low latency, reliability, and security. These aspects are of special interest in R2R and R2I communications and have not yet been fully addressed in robotics, IoT, nor in vehicular networks [90–92].

The SDN paradigm is an excellent solution to global management and control of cloud robotics [93]. It allows for a separation of data and control planes, thus enabling controllers, in conjunction with cloud computing orchestrator, to manage the backhaul and the wireless infrastructure, as well as the mobile nodes and computing elements present in the system [94]. Nevertheless, SDN by itself is not enough to fulfill the cloud robotics requirements, as it cannot properly cope with uninterrupted connectivity and R2R or R2I communication.

The results observed in our previous works [60, 61] and the results presented in Chapter 2 point to the impact of moving smart walker's mobility assistance services to edge clouds. Those results refer to more relaxed values for QoS metrics than those suggested by the latest URLLC 5G specifications in the particular critical real-time application investigated [17]. For instance, network constraints such as latency and packet loss can be deeply attenuated by making use of private and on-site clouds, which reduce the size and complexity of the network while enhancing control over data security and privacy.

Finally, another key aspect of cloud-computing based systems is their demand for uninterrupted connectivity. In eHealth applications, uninterrupted connectivity implies agile handovers for mobile robots in healthcare facilities. Guaranteeing uninterrupted connectivity is an issue not only for the conventional WiFi technology, due to its slow client re-association procedures, but also for the backhaul network re-routing procedures [95]. In other words, the WiFi and the backhaul network both have difficulties in adapting to the

constant displacement of mobile robots. A prototype with initial results on using SDN over WiFi 802.11n and backhaul network has been presented in [19], with the final aim of providing ultra-reliable connectivity to mobile elements. Here, we expand such a solution to manage communication in a cloud robotics system.

3.2 A General Architecture for Cloud Robotics in Clinical Healthcare

We propose an SDN-based architecture for cloud robotics in clinical healthcare. Figure 15 displays an overview of the architecture, which is particularly suited for groups of mobile robots and is inspired by vehicular network architectures, such as the one presented by Contreras *et al.* [92], as they share many of the requirements of cloud-enabled mobile robots. Our multilayered architecture offers transparent integration between the elements of physical robotics, communication infrastructure, and services in the cloud. Furthermore, it tries to fulfill requirements for this type of solution through the multi-layer programmability achieved by SDN combined with cloud computing management. The functional blocks present in each layer of the Fig. 15 illustrate – in a non-extensive manner – common functionality of cloud robotics systems. The main architectural layers are described in the following:

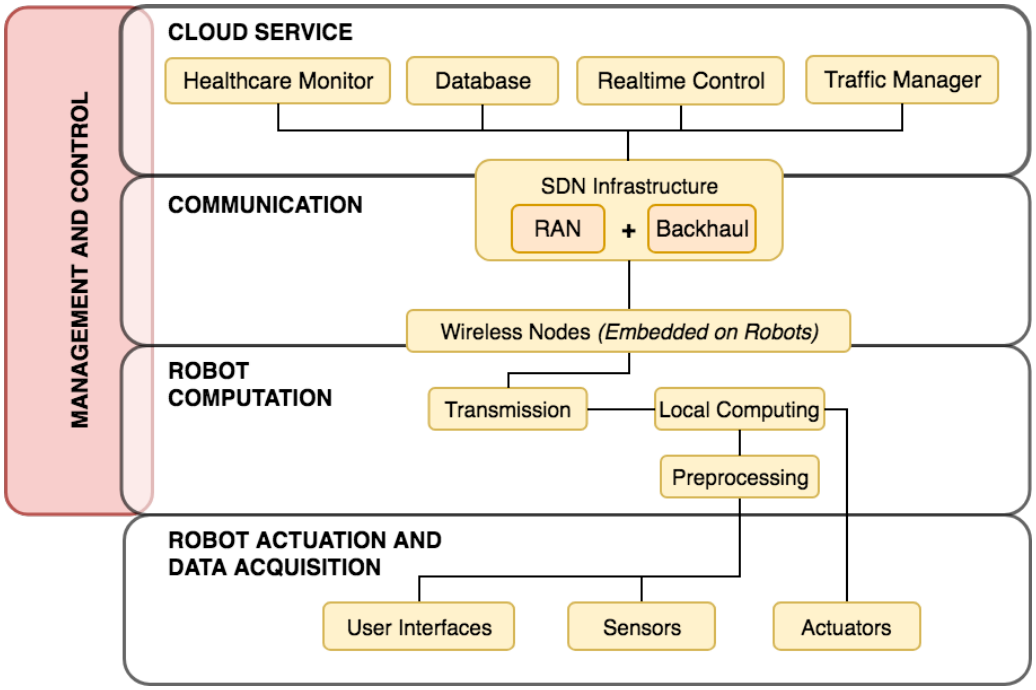


Figure 15 – A proposed reference architecture for cloud robotics in clinical healthcare.

- **Robot Actuation and Data Acquisition Layer:** the bottom layer comprises the elements that allow the robot to interact with humans and the surrounding environment.

The user interfaces enable the HRI by acquiring inputs and providing feedback. The sensory system is responsible for gathering information about the environment and the robot itself, while the actuators respond to the control algorithms commanding the robot's movements.

- **Robot Computation Layer:** this layer represents the mobile robot's embedded intelligence. Each robot should be able to pre-process data obtained from sensors; to manage and control its interfaces; to perform some degree of local computation; and to request which services are needed from the cloud and determine which information to exchange with it. The local computation should ensure efficient data exchange with the cloud and with other robots.

- **Communication Layer:** this layer leverages communication throughout an SDN-based infrastructure. The wireless nodes placed on the robots can exchange data among themselves (R2R) and with the Radio Access Network (RAN). The RAN is connected to the backhaul infrastructure, which is responsible for providing communication with the cloud.

- **Cloud Service Layer:** this layer is responsible for collecting the – pre-processed – information in each mobile node, to be coordinated and processed by the different robotic applications and to perform the management of all cloud robotics mobile elements. The use of distributed or centralized cloud environments allows more efficient processing of information as well as timely access to services requested by any element of the architecture.

- **Management and Control Layer:** this layer is in charge of providing resource and communication orchestration for all layers that it intersects. It is responsible for managing the quality of the signals, for establishing and checking appropriate control metrics, and for handling R2I and R2R routing mechanisms, with the final aim of achieving the most effective communication among the elements.

Considering the designed architecture, we propose a communication solution for indoor environments, such as hospitals and clinics, permeated by mobile robots performing critical tasks. In such a scenario, robots may cover large areas demanding migration between access points to maintain an uninterrupted application-level data exchange with the cloud.

Among wireless networks, the usage of WiFi is the first choice for indoor cloud-enabled robotics due to its almost ubiquitous presence. However, WiFi suffers from crucial issues as long delays for client association and high latency handover procedures [19]. During handover procedures, connectivity is lost for a few seconds, a characteristic that is unsuitable for cloud robotics applications (see Fig. 4). Nevertheless, the WiFi infrastructure needs no complex routing protocols like the ones used in Ad-Hoc networks, which is the most common solution for networked robots communicating among themselves. Moreover, the

WiFi infrastructure mode enables SDN integration, allowing the *Management and Control* layer of our architecture to keep track of the state of the system and to manage the mobile robots' connections through a SDN orchestrator. Thus, we choose to use a modified WiFi topology to ensure the communication – both R2I and R2R – of all elements in the architecture.

The communication solution proposed here employs the concept of split WiFi functions first presented by Martinez *et al.* [19]. This split of functions allows for multi-connectivity among the wireless nodes, which leads to an increased reliability in overall communication by leveraging a novel multi-association scheme for WiFi systems. Moreover, such multi-connectivity allows for the association between wireless nodes even before a handover procedure is necessary, thus enabling reliable handovers with zero Mobility Interruption Time (MIT), and adequate latency and throughput values.

The split of WiFi functions reverses WiFi traditional operation. Instead of placing an infrastructure's wireless access point connected with multiple clients (e.g., multiple computers and smartphones connected to a single office router), we center the wireless connectivity in the robot. To this end, we employ the concept of a mAP, which is installed in a robot's wireless node (see Fig. 15). As part of the RAN infrastructure, there are several fSTA that can associate with the mAPs, serving as access to the cloud applications. The association of several fSTAs to a mAP guarantees the multi-connectivity of robots' wireless nodes with infrastructure (i.e., R2I multi-connectivity), similar to macro-diversity solutions [96]. In such a solution, the handover procedures become a matter of switching the wireless path of data among already associated wireless nodes, thus drastically reducing interruption times. Data traffic from a robot configured as mAP can be switched from one fSTA to another by updating OpenFlow rules in OpenFlow-enabled switches, which are a vital part of the SDN backhaul infrastructure. Further, interface diversity through virtual wireless interfaces can be leveraged so that multi-association is possible between each fSTA and different mAPs. An overview of our SDN-wireless communication topology is shown in Fig. 16.

Considering new high-density access station deployments, several fSTAs would be present throughout the infrastructure to guarantee superior traffic density per area over small cells [97]. In such a network topology, the number of fSTAs must surpass the possible number of robots (or mAPs), thus decreasing the likelihood of two mAPs demanding association with the same group of fSTAs. If this condition is met, leveraging interface diversity would not be necessary.

This communication solution allows robots' wireless nodes to be configured as mSTAs capable of associating with another robot's wireless node bearing the mAP role, as illustrated in Fig. 16. Thus, the robot configured as mAP should be able to manage several robots (configured as mSTA) under its coverage area. In this way, the coverage area of the wireless infrastructure is expanded, guaranteeing connectivity to all the cloud-enabled

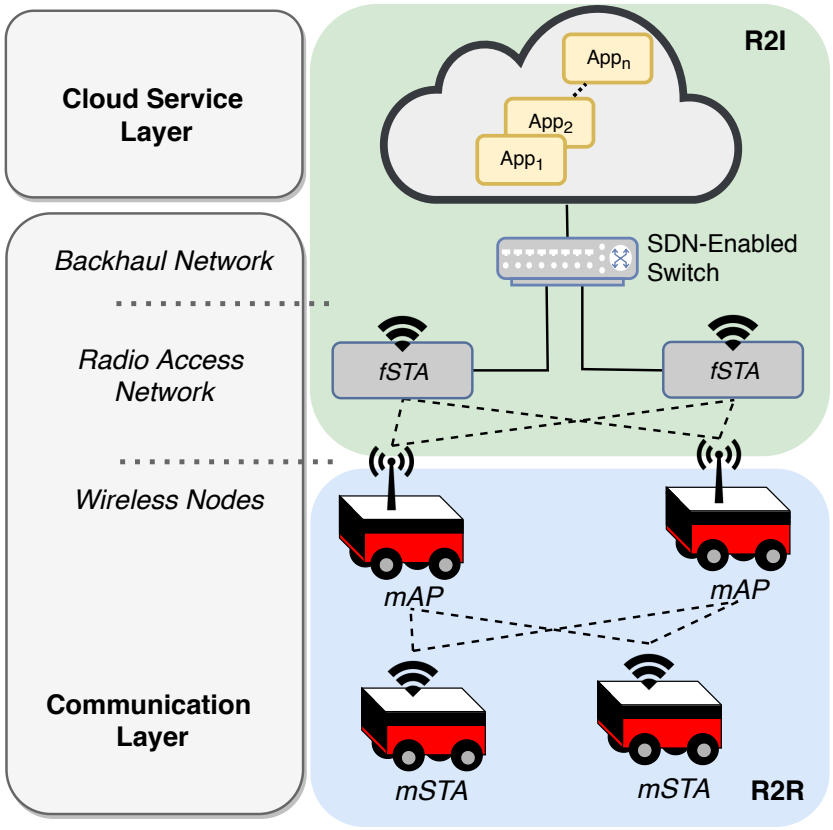


Figure 16 – Overall network topology: the split of WiFi functions enabling R2I and R2R; the dashed lines represent wireless paths stablished between the mAPs, mSTAs, and fSTAs.

robots in non-high-density deployments. This particular type of multi-connectivity among mSTA and mAP also enables direct R2R communication. Similarly, as it happens in ad-hoc vehicular networks, the communication happens with the nearest point between both elements, without the need to access the cloud infrastructure.

The presented reference architecture proposes the use of this SDN-based communication solution to fulfill the connectivity requirement of mobile robots. The particularities of this solution, such as the implementation of network orchestrators, monitoring agents, and decision-making mechanisms, are beyond the scope of this thesis. In Section 3.4, we present a use case based on the smart walker's guidance application; we describe a proof-of-concept implementation of the communication solution and perform a set of experiments to validate its use in cloud robotics.

3.3 Re-Designing the Virtual Trail Service

Having defined the overall architecture and communication solution, one can start designing innovative services for eHealth using mobility assistive devices that are based on that architecture. Here, we re-design the virtual trails service described in Section 2.5.1

aiming at the same use case (i.e., healthcare facilities where impaired patients might need help during their displacement across clinic).

To improve the previously described service, we also consider sensors capable of acquiring data regarding the surroundings. For instance, Laser Rangefinder (LRF) sensors can be used to feed obstacle detection and SLAM algorithms. Figure 17 illustrates the block diagram considering a classic implementation of this HRI strategy. An *Obstacle Detection* module responsible for providing information to a *Safety Manager*, which can scale down the velocities generated by the admittance controller or even stop the walker if there are obstacles too close to the walker. Moreover, the combination of a *SLAM* and a *Path Planning* modules allows for the automatic generation of the path to be followed.

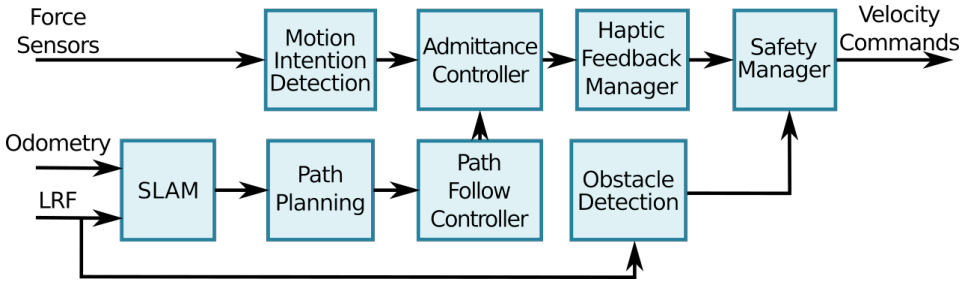


Figure 17 – Functional blocks composing a haptic guidance functionality for patients through dynamic wayfinding, considering an embedded implementation.

Figure 17 considers that every module is implemented on the walker's embedded computer. Thus, the information flow is direct and easy to follow, and the resulting HRI should guide the patient displacement towards a given destination. To transform this wayfinding strategy into a cloud-based service, safety must by assured against possible QoS and other network-related issues. Therefore, one might state two essential procedures: (i) to constantly monitor cloud and network QoS conditions, and; (ii) to handover control to the robot's embedded processing in case of poor QoS or overall failures.

Figure 18 illustrates the redesign of the presented HRI strategy into a cloud service for mobility aid. Figure 18(a) displays a minimum set of functions that must be executed on the smart walker, while also linking each block to our proposed architecture (see Fig. 15). Safety-critical functions are locally processed and a *Service Manager* (or communication manager) is responsible for walker-cloud communication. The *Service Manager* block is also responsible for monitoring the status of the communication and triggering the use of a local version of the admittance controller in case of poor QoS. Therefore, the user should maintain control over the walker even in case of connectivity loss.

Figure 18(b) illustrates the virtual trails service hosted in the cloud. An instance of the service is provided to each robot in the system and a *Service Management* function is responsible for organizing the data exchange, receiving data from sensors and interfaces, and routing it to related control blocks. It is also in charge of sending back to the smart

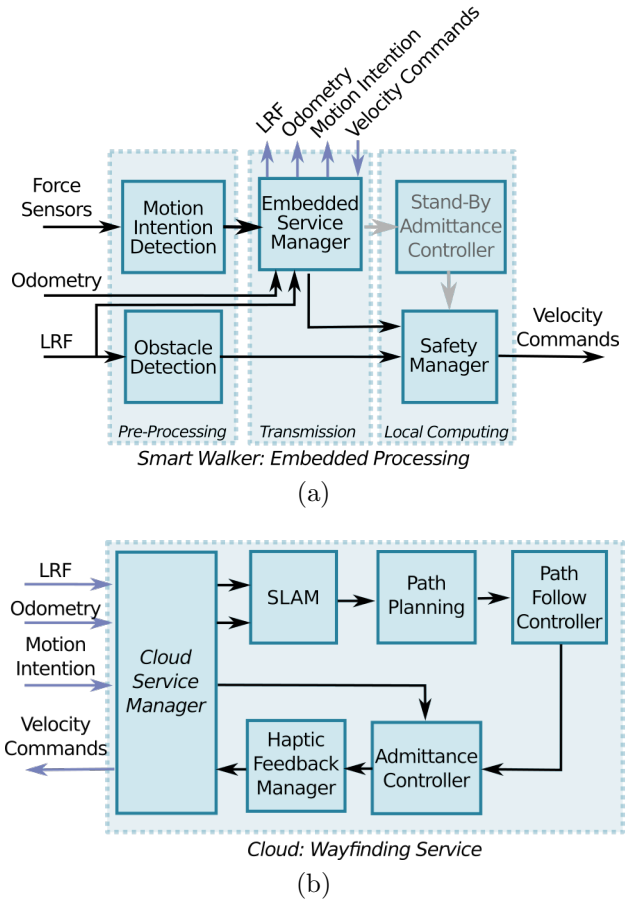


Figure 18 – Adapting the virtual trails control strategy into a cloud service for mobility aid: a) only essential blocks are computed in the smart walker; b) the blocks composing the wayfinding service deployed in the cloud.

walker the reference velocity signals that will ultimately generate the haptic feedback to the user.

3.4 A Demonstrative Use Case

The design discussed in the previous sections can now be implemented in a demonstrative use case. We chose to emulate the displacement along hospital corridors and thus created a fixed path composed of straight lines and curves to the left and right sides. By doing so, the *SLAM* and *Path Planning* modules displayed in Fig. 18 become unnecessary and are not implemented. The odometry module is used for localizing the robot and feeding the *Path Follow Control* module in our cloud service. These modules are implemented using the ROS framework[1] (Kinetic version), and each module corresponds to a computing process being executed in a cloud VM. To infer the users' intention of movement, we use data from the smart walker's force sensors and then remotely generate the haptic

[1] The overall ROS operation is detailed in Chapter 4. Here, it suffices to understand that ROS processes are related to data acquisition and processing, and generating control signals. The exchange of information among processes in the same machine is handled by the ROS middleware.

feedback to control it in the way described in Chapter 2.5.1. Once again, we employ the CloudWalker and the experiments are performed using the UFES Smart Walker robotic platform (see Fig. 9).

A Raspberry Pi 3 Model B (Raspbian Stretch Lite v4.9) is installed in the walker to act as a gateway linking the walker's embedded computer – in which there is no wireless interface – and the local network. The Raspberry Pi is connected to the smart walker's PC/104-Plus via an Ethernet cable and, at the same time, connected to the laboratory's network infrastructure via WiFi. The robot-cloud communication scheme is based on Mello *et al.* [60, 61]: on the Raspberry side, the communication software encodes sensor data into UDP/IP packets and waits for an answer containing the velocity commands that must be passed to the PC/104-Plus; on the cloud side, the software is used to decode the received data and to pass it to the controllers, from which the communication software receives velocity commands to be encoded into UDP/IP packets and sent back to the Raspberry Pi on the walker. Samples of odometry, velocity, and interaction forces (128 bytes packets) are sent to the cloud-based service every 10 ms. The Raspberry Pi also plays the role of the wireless node (see Fig. 16) for R2I and R2R communication.

The cloud service is implemented in an edge cloud based on OpenStack located at the NERDS' datacenter. Openstack is an open-source cloud management software capable to provide and manage the processing, storage, and network resources, as well as automatic instantiation of VM, and scaling of computing resources to meet QoS requirements. The cloud services are implemented in VMs running on the cloud platform. The number of VMs can be scaled whenever a new service is requested or a device is turned on or off. This provides flexibility to serve multiple devices that may not be in use all time. Here, a single ROS-enabled VM is responsible for processing sensor data incoming from the robot and generating control signals to command the movement of the device.

The SDN-enabled WiFi communication is implemented using fSTAs installed in PCs running the Ubuntu Server 14.04 with an Ethernet interface and a wireless card with the Atheros AR9380 chipset working in the 5 GHz band. The wireless nodes are installed in two Raspberry Pi 3B equipped with a wireless interface with a Ralink RT5572 chipset via USB, to allow operation in the same band. One of those Raspberry Pis is the one embedded in the smart walker, as previously stated. The second Raspberry Pi is used as the wireless node of a – emulated – second smart walker to enable testing the R2R communication feature of the proposed system. To provide mAP functionalities, the *hostapd daemon* and the OpenVSwitch are installed. A Ryu OpenFlow Controller centralizes orchestration and management of the architecture. We use Openflow 1.3 in mAP, mSTA, fSTA and in a backhaul switch that links the RAN infrastructure and the cloud platform.

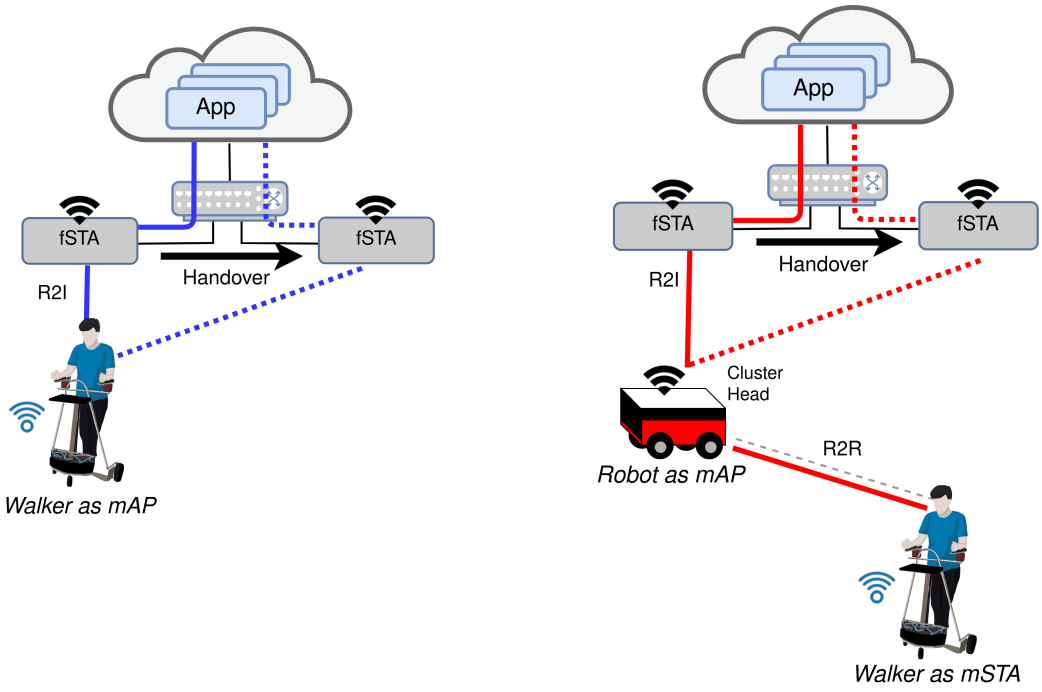
3.4.1 Experiment Scenarios

We consider a multi-robot environment in which the smart walkers possesses only the necessary intelligence to assure patient safety and must communicate with the cloud to receive the remotely generated control signals. Through patient-walker interaction, the cloud service is responsible for guiding patient displacement through a corridor with right and left turns.

Two scenarios are implemented to test our SDN-based solution, as described below and illustrated in Fig. 19:

- Single-Hop (R2I): the communication path involves mAP → fSTA → cloud service: the robot is configured as a mAP and must communicate with the cloud via the fSTAs of the network infrastructure.

- Multi-Hop (R2R+R2I): communication path involves mSTA → mAP → fSTA → cloud-based service: the walker is now configured as a mSTA, and thus associated with another robot that bears the mAP role.



(a) (b)

Figure 19 – Experiment overview: a) single-hop (R2I), and; b) multi-hop (R2R+R2I) communication.

The multi-hop scenario is characterized by data flowing from one robot (mSTA) to another (mAP), which is responsible for routing the packets to the network infrastructure through a fSTA. In both scenarios, a handover process is performed to change the path in

which data is flowing (i.e., mAP switches communication between associated fSTAs). The handover is automatically performed ten seconds after the beginning of each experiment.

In each of the described scenarios (i.e., single- and multi-hop), we perform two kinds of tests: regular and stress tests. The regular tests consider the common operation of smart walkers and aim at validating our proposed architecture and system. The regular tests are conducted exchanging sensor data and feedback data via the described walker-cloud communication.

The interaction is recorded by storing in the cloud sensor and feedback data. To evaluate the effects of the network on the system, we measure the packet loss ratio and the E2E latency. The packet loss ratio is estimated by comparing the number of incoming and outgoing packets in the robot's and the cloud's network interface cards. E2E latency is measured here as the elapsed time between sending data from the smart walker and receiving a response from the cloud-based service, including cloud processing time.

The stress tests consider extreme communication scenarios and are performed to evaluate the performance of our SDN-based communication solution. To push the limits of data exchange in our solution, the *iPerf*[2] tool is employed to generate artificial traffic. We perform throughput measurements for TCP flows using the *iPerf* while also measuring RTT latency between walker and cloud using the *ping* tool.

3.5 Results

The experiments were successfully performed considering both scenarios. During the regular tests, the user was able to interact with the walker and follow the assigned path in the single- and multi-hop scenarios. As the HRI aspect of the smart walker use is not the focus of this chapter, it suffices to state that the effects of the network/cloud insertion in the control loop were not perceptible during the experiments.

During each run of the regular tests, about 5000 UDP packets were generated and exchanged between the smart walker and the cloud. The average E2E latency is around 2.5 ms and 2.75 ms in scenarios R2I and R2R+R2I, respectively. These small latency values are due to the physical proximity of the cloud platform. Fig. 20 displays distribution of E2E latency in histograms for both scenarios. The observed packet loss rates were below 0.01 % in both cases. Such rates for packet loss can be regarded as low, especially when considering the handover process. There are no visible effects to the user of the walker during the handover process, neither larger values are observed for packet loss nor latency on those specific moments.

During the stress tests, we measured throughput values around 26 $Mbps$ and 15 $Mbps$ in the single- and multi-hop scenarios, respectively. As expected, the multi-hop wireless

[2] The *iPerf* is a tool used to measure the maximum achievable throughput on IP networks. It supports different protocols and parameters and reports statistics regarding throughput, packet loss, and other parameters.

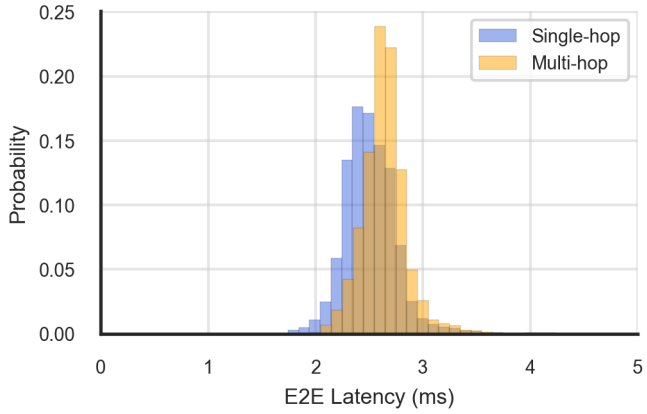


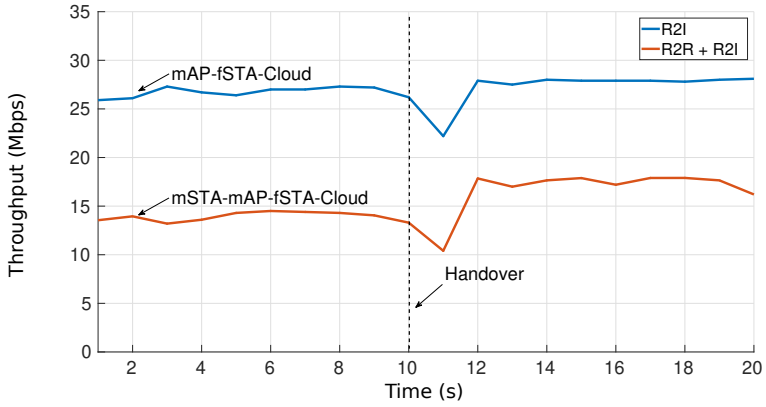
Figure 20 – E2E latency distribution during the regular tests considering the Single-Hop and Multi-Hop scenarios.

communication degrades the achievable throughput given the increased probability of error and retransmission in large TCP packets. The handover process occurs at 10 *s* (see Fig. 21(a)), when the mAP migrates from one fSTA to another. A minor degradation in throughput in both scenarios is observed but without connectivity interruptions, which is of fundamental importance. Considering the regular WiFI operation, as demonstrated by Martinez *et al.* [19], handover processes demand a re-association procedure which results in three to five seconds of lack of connectivity – as previously shown in Fig. 4. In our SDN-based solution, no re-association is needed during the handover process given the multi-connectivity property of our communication solution: as the fSTAs are already associated to the mAP, the handover is performed by switching the routing from one fSTA to another, resulting in the zero MIT observed.

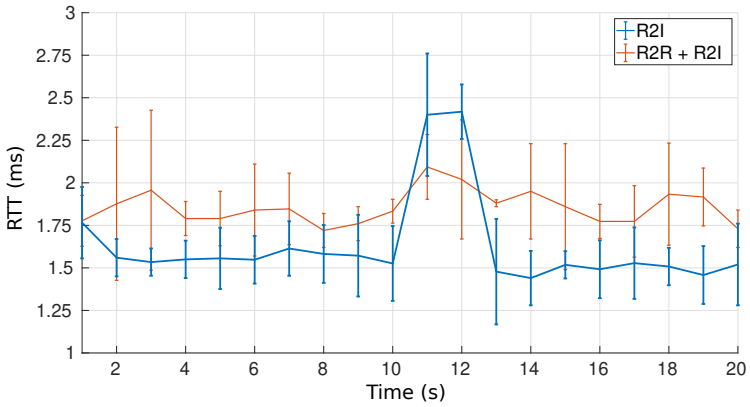
The RTT behavior in both scenarios is shown in Fig. 21(b). Fluctuations are kept under acceptable levels but is noteworthy that one-hop (i.e, R2I) connectivity is more sensitive to handover operations than the multi-hop one (i.e., R2R+R2I). This can be explained by the higher throughput loss ratio observed in the multi-hop scenario during handovers, which leads to longer queues and thus affecting the RTT measurements. Since queued packets associated with higher RTT times are more prone to be dropped and consequently not considered in the RTT measurement, the average RTT is less affected.

3.6 Conclusions

As discussed in Chapter 2, avoiding the loss of connectivity is a major requirement in cloud robotics and is also a challenge in WiFi networks. In this chapter, we presented a reference architecture for cloud robotics alongside a prototype built with off-the-shelf components to satisfy the wireless connectivity requirements of cloud robotics applications. Over a demonstrative use case, we discussed the design and implemented a cloud-based guiding service to assist impaired individuals during the use of a smart walker. The



(a)



(b)

Figure 21 – Results from the stress tests: a) average throughput measurements; b) RTT measurements.

SDN-based network infrastructure is also implemented and validated, in which the mobile nodes of the network can work either as clients or access points. The key contribution of our proposal is to leverage a solution based on multi-connectivity to perform seamless handovers in large facilities where the coverage area of a single access point would not suffice. Our experiments demonstrate our system's capacity to offer uninterrupted wireless connectivity, thus fulfilling this important requirement. By working with WiFi's 5 GHz band, we have an increased throughput capacity and a minor coverage area when comparing with the traditional 2.4 GHz band. Nevertheless, since our solution allows for easy migration from one fSTA to another, increasing the number of fSTAs in the environment to increase the coverage are is desirable as it also increases the number of possible simultaneous connections between an mAP and multiple fSTAs. Moreover, as our results show, the combined use of WiFi and edge clouds allow for high throughput capacity and low latency, satisfying other major requirements of cloud robotics systems.

Another important aspect of cloud robotics implementations is the adopted method to exchange information between the robot and the cloud. The way we dealt with robot-cloud communication in this chapter presents major limitations, being the lack of generality in

the encoding of sensor data its major problem. Our approach encodes all sensor data into a single flow, demanding further customization for each application or different configuration. The same is true the other way around, and there is a custom layer encoding the velocity commands that are sent back to the robot. This method is not scalable and adapting the communication software to respond to changes in the application is a time-consuming task. Chapter 4 addresses the lack of a standard tool and presents a novel framework to enable robot-cloud communication to fulfill another practical requirement of the cloud robotics paradigm.

4 A Novel Platform for ROS-based Robot-Cloud Communication

Despite the increasing popularity of the cloud robotics paradigm, the literature on the field still lacks a comprehensive analysis of several aspects of the technology. The adoption of common standards and frameworks is fundamental for the development of the field and to allow practical works to be reproduced and compared. In this Chapter, we present a ROS-based framework for robot-cloud communication, the PoundCloud, and discuss its integration within the cloud robotics ecosystem. The PoundCloud is open-source and freely available. To validate the PoundCloud functionality, we instantiate a virtualized cloud robotics testbed and conduct a series of experiments to verify its performance and suitability for practical implementations.

4.1 Background and Related Work

The term cloud computing is commonly employed as an abstraction to a set of distributed and reconfigurable computing resources that can be provided on-demand to multiple users. Thus, cloud computing services are usually provided in three models. The first one is Software as a Service (Saas), where the end-user has direct access to the application-level software running on the cloud. The second model is known as Platform as a Service (PaaS), in which the provided resources are in the form of application development platforms. Finally, the third model is Infrastructure as a Service (IaaS), where the user can dynamically provision, configure, and tailor computing resources to its own needs [98].

Robotics researchers have explored cloud computing to build virtual environments through Web services, where users can connect with the robots via dashboards [99]. In this sense, commercial cloud service providers (e.g., Google Cloud Platform and Microsoft Azure) offer a wide variety of products that have been used to empower robots' capabilities [2]. An example of such an approach is the work developed by Varrasi *et al.* [100], which used IBM Cloud Services' products – Watson Speech to Text and Watson Visual Recognition – to improve human-robot interaction.

Following the cloud robotics trend, variants of the term Robot as a Service (RaaS) have been proposed as cloud computing service models offered through the virtualization of robotic functions and applications [40,99,101]. In an early work, the DAvinCi platform [102] – presented in 2010 – implemented robotic services into a Hadoop cluster [103] using the ROS framework. Then, as part of the RoboEarth project [104], the Rapyuta cloud robotics platform was introduced to support outsourcing computing tasks [3]. It used WebSockets for communication and provided ROS-based cloud robotics services. Although other cloud

robotics platforms have also been presented in the literature (e.g., [4, 25, 105]), Rapyuta is an important case as it eventually evolved into the first commercial platform to provide cloud robotics services[1].

Regarding commercial initiatives, the AWS RoboMaker[2]. is a step towards Amazon's own cloud robotics platform. Currently, the RoboMaker provides SaaS via ROS-based packages capable of offloading certain processing tasks to the cloud. On the other hand, Google's Cloud Platform[3]. is a PaaS aimed at connecting robots and cloud applications via Kubernetes clusters and communication based on the Transport Layer Security protocol. Despite incipient, these initiatives may be of great importance to foster cloud robotics development. Nevertheless, when operating over the SaaS or PaaS paradigms, code portability and compatibility are hampered, tying the application deployment to the service provider. In this work, we present an open framework that can leverage any cloud platform capable of offering the IaaS model without the need for code redevelopment.

The lack of analysis on cloud robotics' implementation issues and standards hampers direct comparisons between different works. Although some of the problems can be tackled by cloud robotics platforms, which mainly solve implementation-related issues regarding the cloud, an alternative way is to explore the use of cloud robotics testbeds. Testbeds for experimentation may accelerate research by providing specific solutions while relieving researchers from implementing the whole cloud robotics stack. Examples of cloud robotics testbeds can be found in Ribeiro [14] and in Lu *et al.* [26]. In particular, the "Federated Union of Telecommunications Research Facilities for an EU-Brazil Open Laboratory" (FUTEBOL) project provides an experimentation testbed for real-time Industry 4.0 applications, including cloud robotics [14, 106]. Such a testbed can either be used as a semi-closed solution for experimentation or as an enabling infrastructure for hosting cloud-based services.

Several works have reported the execution of robotics tasks via the cloud robotics paradigm. Navigation systems are commonly found to be implemented in UGVs as they provide safe and effective environment interaction [107]. Furthermore, these systems are computationally expensive, as they implement several perception, cognition and actuation modules. In this sense, cloud-based navigation assistance services have been implemented by Limosani *et al.* [108], in which the remote platform provides relevant maps based on landmark detection, and by Gao & Cheng [109], which implemented a cloud-based mapping service using ROS. An analysis on offloading navigation-related computing tasks – also using ROS – is presented by Salmerón *et al.* [110], which concluded that, given some constraints, removing most of the robot's embedded computation might be a viable option. Whereas Cardarelli *et al.* [42] presented a navigation service for groups of robots that were able to provide global plans based on multiple sources of information, while each

[1] Website: <https://www.rapyuta-robotics.com/>. Accessed on December 1, 2020.
[2] Website: <https://aws.amazon.com/robomaker/>. Accessed on December 1, 2020.
[3] Website: <https://googlecloudrobotics.github.io/core/>. Accessed on December 1, 2020.

robot remained responsible for its local planing. Applications based on machine vision are also commonly found in industry, allowing for tasks related to classification, identification, and context acquisition [111]. Recent advances in deep neural networks and robot visual perception and control enable tasks performed over ever-changing environments [112, 113]. Often, such applications delegate the image processing tasks to remote services (e.g., Google Cloud Vision API) capable of reducing processing times [111, 114]. Human-robot interaction can also benefit from computer vision (e.g., [115, 116]) and may be enhanced by cloud robotics concepts [2].

4.1.1 Common Practices and Software: The Robot Operating System

In the search for common standards in robotics programming, several robotic-focused middleware have been developed, such as the ROS [117], YARP [118], and OROCOS [119], to cite a few. Due to its enormous adoption by developers, ROS is currently the most popular robotic middleware and is slowly becoming the standard in the field [30]. There are several reasons for ROS' popularity: open-source nature, thousands of packages and libraries available, integration with several robots and sensors, and inherent modularity. Besides developers and academics, ROS' popularity also increased with companies throughout the years (e.g., Clearpath Robotics and Fetch Robotics) and efforts towards the industry have been made by the ROS Industrial Consortium[4]. ROS' common practices (e.g., standard units of measure, reference frames, and package naming convention) are standardized in documents known as ROS Enhancement Proposals to better integrate and re-use software components.

In ROS systems, communication among ROS nodes happens in a pub/sub fashion via ROS topics or in the client/server paradigm via ROS services. When using topics, data is communicated in the form of messages, which are simple data structures formed by typed fields. ROS deals with the exchange of information among nodes through network sockets using transport layer protocols built on top of TCP and UDP – TCPROS and UDPROS, respectively. TCPROS is used by default on top of TCP sockets instantiated within the nodes.

A special node, the ROS master, is responsible for managing the system operation. The ROS master provides naming and registration services to the other nodes in the ROS system. It tracks publishers and subscribers to their associated topics to enable individual ROS nodes to locate one another. By communicating with the ROS master, nodes with a shared interest in the same topic can locate each other to start a peer-to-peer communication. This process is illustrated in Fig. 22: upon startup, nodes *NodeA* and *NodeB* advertise to the master that they will publish messages to topics named */topic1* and */topic2*, respectively; to be able subscribe to those topics, *NodeC* communicates with the master to register two subscribers, each associated with one topic. Then, as there are

[4] Website: <https://rosindustrial.org/>. Accessed on December 1, 2020.

both publishers and subscribers registered to each topic, the master sends instructions to the nodes to establish a connection and start communicating with one another.

Note that the ROS system represented in Fig. 22 is defined by a single master node and is comprised of two separate machines. The ROS framework allows for the integration of multiple robots and computers operating under the same ROS system - or, in other words, the same ROS master. Thus, each of the computation nodes communicate with the master to exchange information and to update forwarding rules.

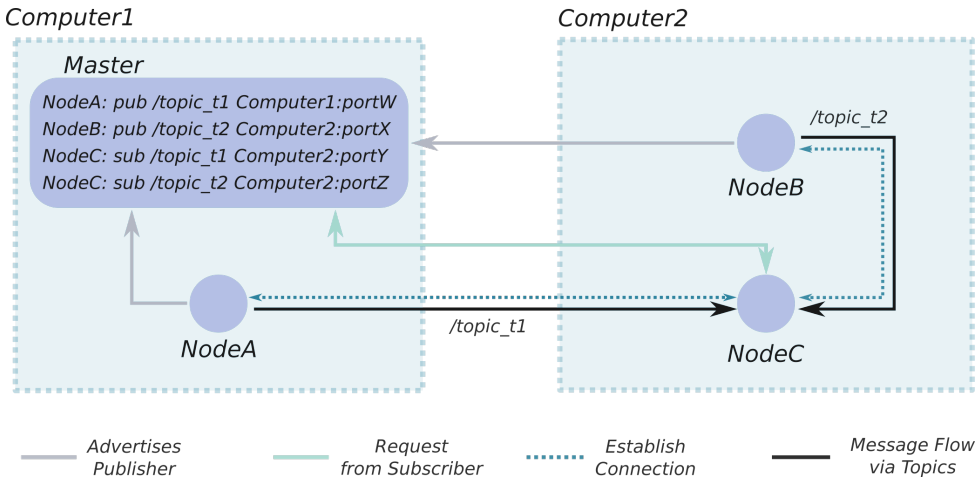


Figure 22 – ROS operation: the master registers and provide information for publishers and subscribers to establish connections and allow nodes to exchange information directly via topics.

As ROS is designed for communicating distributed systems within the same LAN, its distributed nature enables the use of multi-robot systems or even delegating processing tasks to another machine. Thus, the standard ROS communication protocols are not sufficient when communicating a robot with a system outside the local network, as it is necessary for cloud robotics; in these situations, there is no standard approach.

The package *ros_bridge* is the most common way of communicating ROS systems with non-ROS systems [2, 30]. It works by establishing a client/server connection among two systems and by converting ROS messages into JSON strings that can be received by non-ROS systems. When communicating over the Internet, the *ros_bridge* is mostly used to link the robot to a web page or dashboard accessed by a user. Despite not being *ros_bridge*'s primary purpose, it can be adapted to be used in multi-master ROS systems. Although this approach may be sufficient for the exchange of some kinds of messages (i.e., with small payload and low frequency), the encoding into JSON strings may not be efficient[5]. Thus, *ros_bridge* is unsuitable for several cloud robotics applications [30].

Koubaa *et al.* [30] proposed the ROSLink, a cloud-based approach in which the cloud acts both as a proxy linking robots and users, and also as a server capable of providing

[5] E.g., consider the decimal value 10 encoded as a single byte of type *uitn8* in the original ROS message; the JSON string conversion will turn it into the 3-byte string "10".

services to robots. Similarly to *ros_bridge*, ROSLink is also based on JSON, which limits its range of applications. Hajjaj & Sahari [120] proposed a port forwarding technique as a better alternative for robot-cloud communication. Although suitable for some use cases, port forwarding requires permission and privileges for proper configuration, which may be complex – or even impossible – to obtain in some scenarios. Sorrentino *et al.* [121] presented an approach based on reverse SSH tunneling as an alternative for port forwarding and VPN techniques. Such an approach leveraged a cloud-based server to allow for teleoperation via web pages. Other approaches in the literature vary; for instance, in a previous work of ours [60], specifically designed ROS nodes use sockets to act as gateways for robot-cloud communication, directly encoding custom ROS messages into specialized arrays to be transmitted using UDP/IP packets. This is the same approach used to perform the experiments presented in Chapter 3. Nevertheless, this approach is not scalable and can not be easily adapted for other applications since it demands further customization to transmit different kinds of messages.

Another common characteristic among several papers dealing with ROS-related cloud robotics is the lack of details regarding the communication scheme; it is often not clear how the authors implement robot-cloud communication or, when it is clear, the paper usually does not present experiments using real robots, usually performing simple demonstrative simulations. Moreover, there are other important issues to be addressed in ROS and, most specifically, cloud robotics, such as security and privacy of users, which are beyond the scope of this thesis.

Despite its popularity and benefits, using ROS does not come without problems. ROS was designed under the assumption of use on a single robot with reasonable computational resources on board and no real-time requirements. Moreover, ROS' design also considers excellent network connectivity and distributed use under local area networks. The second version of ROS, the ROS 2, aims at circumventing most of its predecessor's deficiencies. As ROS 2 is still in its early stages of development and adoption, it is not nearly as popular as ROS 1, and thus in this work we only consider ROS 1 (referred simply as ROS).

4.2 The PoundCloud Communication Framework

Our approach leverages independent ROS systems to decouple the operation of the robot and cloud-based service. In other words, we aim at instantiating one ROS master at the robot and another one at the service. Although ROS was not built to support multi-master approaches by default, several solutions have been presented in the literature (e.g., [122–125]). In general, multi-master approaches are used for multi-robot systems under the same LAN, whereas the subject is less explored when it comes to communicating over the public Internet.

The standard approach for configuring ROS multi-master systems is the Multimaster

FKIE[6], a software package that establishes and manages multi-master networks. In this approach, special nodes in each ROS system find each other and synchronize the information registered in each ROS master to make it available. Nevertheless, the Multimaster FKIE requires bi-directional visibility among all ROS systems, which make it suitable only when certain network configurations are used (e.g., when all robots are in the same LAN) [121].

Tardioli *et al.* [125] presented a novel solution for ROS multi-master communication in mobile multi-robot systems, the Pound. This solution creates a ROS node – to be referred to specifically as a Pound node – in each ROS system to manage the communication among them. Pound nodes communicate among themselves in an ad-hoc manner to exchange ROS messages relative to a given set of topics of interest and also to route the traffic of those messages towards the destination robot. Upon initialization in a given ROS system, the Pound node takes two actions: (i) it subscribes to a set of ROS topics to send the messages relative to those topics to other ROS systems, and; (ii) it advertises that it will publish on another set of topics the messages that it expects to receive from other ROS systems. Figure 23 illustrates the Pound operation. When a Pound node receives a message from one of the topics it subscribes, it will encode the message into UDP/IP packets and send it to another Pound Node – in another robot – which may be either the final destination of the packet or an intermediary hop that will route it to the destination[7]. When a Pound node receives a packet addressed to it, it decodes the ROS message and publishes the message to the adequate topic. Thus, the Pound acts as a middleman to allow for different ROS systems to communicate.

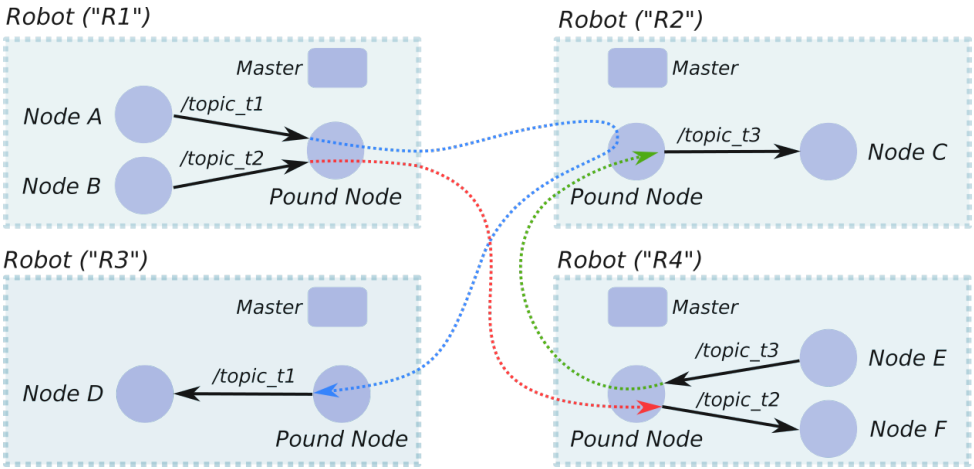


Figure 23 – The Pound operation on multi-master ROS systems: Pound nodes communicate messages published on a given set of topics; red and green dashed lines illustrate the direct exchange of messages among Pound nodes, whereas the blue dashed line indicates the use of an intermediary hop to route the traffic, which departs from *R1*, passes by *R2*, and finally reaches *R3*.

[6] Website: <http://wiki.ros.org/multimaster_fkie>. Accessed on December 1, 2020.

[7] This characteristic of using intermediary hops to route the message through a sequence of robots is inherent to the ad-hoc nature of the Pound.

The Pound also manages packet flow prioritization and packet fragmentation to meet pre-configured throughput and jitter requirements [125]. This mechanism aims at solving a problem in which, when sending data, flows of information with small payloads and higher priority may be delayed if there are concurrent flows with larger payloads. Tardioli *et al.* [125] details the Pound implementation and compares it – obtaining positive results – against other ROS multi-master solutions. The positive aspects of the Pound place it as an interesting starting point for the development of a ROS-based framework for cloud robotics.

In this thesis, we present a novel robot-cloud communication scheme based on ROS and built on top of the Pound, the PoundCloud[8]. While the Pound targets multi-robot communication based on an ad-hoc local network, the PoundCloud works under the assumption that the cloud platform is likely to be reachable only through the public Internet. Thus, our solution generalizes the Pound to allow for communicating robots and cloud platforms via the Internet.

With the PoundCloud, the robot may be connected to the Internet using a Network Address Translation (NAT) service whereas the cloud may be either within a local network infrastructure (e.g., edge cloud) or even on a remote platform reachable via a public IP. No external network configuration (e.g., IP forwarding, VPN, or specific routing schemes) is needed. Network-related configuration on the PoundCloud requires only the cloud IP address and the network ports it will be using[9]. To the best of our knowledge, there are no other tools with similar features targeting ROS. Table 1 summarizes the main differences between the original Pound and the PoundCloud operation.

When using the PoundCloud, robot-cloud communication is comprised of two stages. In the first stage, the robot contacts the cloud-based service in a client/server approach. Thus, upon startup, the robot's PoundCloud instance sends a service request to the cloud's PoundCloud to establish a connection. The cloud registers the robot's IP address and answers the request, thus establishing the connection.

Then, after communication is established, a PoundCloud ROS node is initiated in both robot and cloud to interface with their ROS systems. These nodes subscribe to the relevant topics and use the PoundCloud connection to send the published messages to each other. The PoundCloud incorporates the standard Pound operation with respect to the management of packet flows transmission, taking advantage of its improved performance.

Figure 24 illustrates the operation of the system. Topic *namespaces* are used to identify the source of the information. Thus, data that should be sent to the cloud will be published on topics under the namespace that identifies the robot. The same is observed on the

[8] Publicly available at https://github.com/ricardocmello/ros_pound_cloud (a tutorial can be found in the branch *example*).

[9] Some attention must be paid to the configuration of the network ports that the PoundCloud will use to communicate, as some ports may be blocked by firewalls that interface with the public Internet either on the cloud side or on the robot side.

Table 1 – Main differences between the regular Pound and the PoundCloud, our version for cloud robotics.

Characteristic	**Pound**	**PoundCloud**
Main purpose	Multi-robot communication	Robot-cloud communication
Wireless communication	Ad-hoc mode	Infrastructure mode
Robot-robot communication	Peer-to-peer	Peer-to-peer
Robot-cloud communication	Not possible	Service-oriented (client-server approach)
Information routing	Pre-configured peer-to-peer routing	Robot routes to a gateway (e.g., a wireless AP)
LAN requirement	Robots must be on the same LAN	Not a requirement, but the system can work if robot and cloud platform are in the same LAN (e.g., edge cloud platform)
Operation over the Internet	Not possible	The cloud must have a public IP that can be reached by the robot
IP restriction	Robots must have sequential private IPs (e.g., 192.168.1.1, 192.168.1.2, etc.)	Cloud must be reachable from robot; robot may be behind a NAT

other way around: data flowing from a given cloud-based service will be published under the service's namespace. This separates the data that is being exchanged between robot and cloud from the general ROS system; relay nodes are used to map the topics of interest, bridging the PoundCloud and the rest of the ROS system.

It is worth noting that the ROS does not provide any security measures when transmitting data by default; the Pound nor the PoundCloud add any security features on top of the transmitted ROS messages. Although research applications are often able to tolerate insecure communications, commercial and industrial applications cannot fail to consider security aspects. Although it is beyond the scope of this thesis to implement security features into the PoundCloud, future works should address this issue.

In the following section, we present a functional validation of the PoundCloud using a controlled setup composed of a simulated robot and an emulated cloud platform.

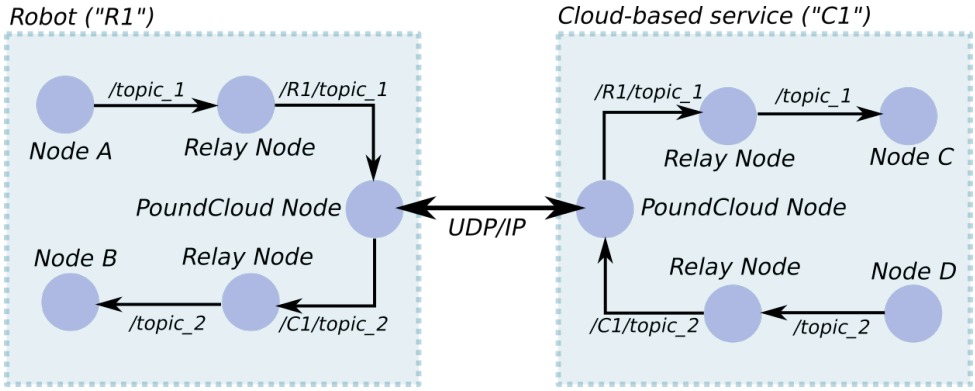


Figure 24 – The PoundCloud approach: a distributed ROS multi-master system for cloud robotics.

4.3 PoundCloud: Functional Validation

In this section, we demonstrate PoundCloud and validate its operation. Here, we verify the feasibility of using the PoundCloud in a distributed multi-master environment. To do this, we propose a virtualized cloud robotics testbed to allow us to tune and control most aspects of the system. We then design and perform a set of experiments on this testbed to validate the PoundCloud and better understand its effects on multi-master ROS systems.

Our virtualized testbed is composed by three logical machines representing the main components of a cloud robotics system: the robot, the communication network, and the cloud. We use the term logical machines to indicate that they can either be separate computers or VMs running on the same host computer. The logical machine that represents the robot is comprised of a robotics simulator, a ROS system running the components which would be executed within the robot's embedded computer, and the PoundCloud framework, which interfaces the robot's ROS system with the cloud. The communication network is represented by a logical machine that deals with all communication traffic. Thus, robot and cloud must route their network traffic through this machine to be able to reach each other. To emulate network effects on the communication, this machine inserts network-related constraints (e.g., packet loss, latency) in the traffic. Finally, the logical machine that represents the cloud executes the ROS system with a set of controllers and also the PoundCloud to interface with the robot's ROS system.

The virtualized natured of the proposed testbed is flexible enough to allow for multiple ways of implementing it. For instance, all of the logical machines can be instantiated in a separate physical machine or even in the same machine. As the integration of the conceptual blocks mimics an actual cloud robotics system, not all components need to be virtualized. Nevertheless, the more complex the implementation, the less control one has over the testbed. For example, if an actual robot is used instead of a machine running a simulator, the communication network will no longer be composed only by a logical machine, but it will also encompass the actual network components that are used to

connect the robot. In this case, even though one might estimate and even model the behavior of those real components inserted in the system, there is little control over the constraints set by those devices. In case one needs to focus on the study of cloud computing techniques (e.g., parallelization, scalability), an actual cloud platform can be used, whereas robots and the overall communication network can be virtualized within the cloud.

Here, we choose to implement our virtualized testbed in a single physical machine to increase our control over its components. The communication network and the cloud are instantiated in separate VMs whereas the robot logical machine is instantiated on top of the host computer's Operating System (OS). A detailed representation of this implementation is presented in Fig. 25. This figure can be used to guide the reader through Sections 4.3.1 to 4.3.3, in which we discuss the implementation of each of those components.

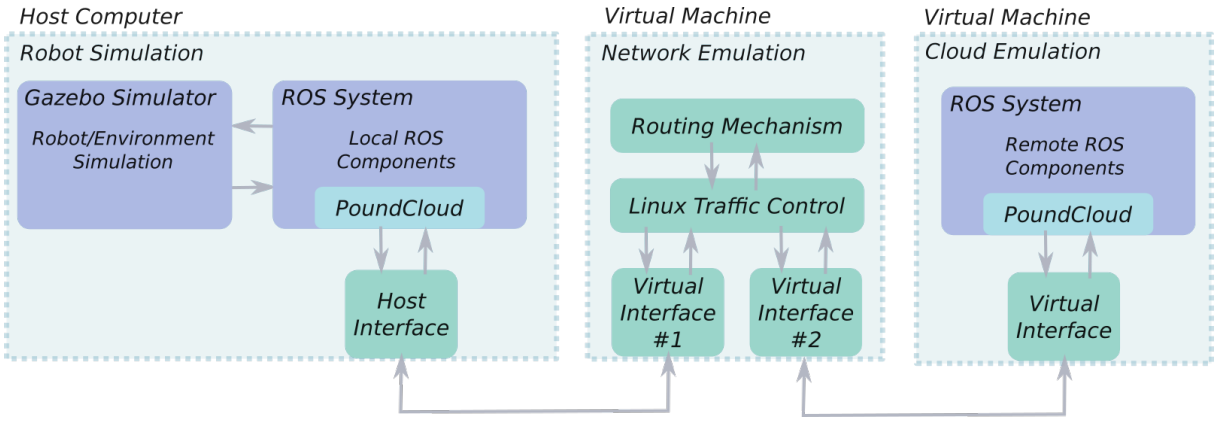


Figure 25 – Functional blocks of the virtualized testbed for cloud robotics experimentation used to validate the PoundCloud operation.

4.3.1 Robot Simulation

We use the Gazebo[10] platform to simulate the robots and the surrounding environment. The Gazebo is an open-source simulator that enables experimentation with simulated groups of robots in complex indoor and outdoor environments. It integrates a robust physics engine, graphic rendering, and programmatic interfaces for sensor simulation and actuator control. Moreover, Gazebo has built-in integration with ROS and can subscribe and publish to ROS topics. In fact, the Gazebo software can be started directly from the ROS environment using *roslaunch*, a tool for automatic initialization of ROS systems, to initialize Gazebo as a ROS node.

The Gazebo offers a wide variety of models of robots, sensors, actuators, and environments out-of-the-box to allow for fast prototyping. One may also choose to develop and use their own models to accurately depict a target environment (e.g., an specific laboratory or building) or to include custom robots and components. This allows for customizing the experimentation scenario to mimic real-world settings and situations.

[10] Website: <http://gazebosim.org/>. Accessed on December 1, 2020.

As displayed in Fig. 25, while Gazebo is responsible for simulating the robot's actions, perception, and interaction with the environment, it also communicates with a local ROS system. This communication is performed via ROS topics and the ROS system execution is decoupled from the simulation environment. In other words, considering input and output topics of the local ROS system (e.g., sensor readings as input, velocity commands as output), the same system can be used to interface with either a simulator, as we do here, or with an actual robot.

The Gazebo and the local ROS system are the two main modules of our logical machine representing the robot. This logical machine is instantiated on top of the OS of a host computer. As shown in Fig. 25, this computer also hosts the two virtual machines composing the remainder of the testbed, described in subsections 4.3.2 and 4.3.3.

A PoundCloud instance is executed in the host computer to communicate with the cloud. Thus, a PoundCloud node is started within the local ROS system to subscribe and publish to a given set of topics. The PoundCloud opens its communication sockets on a set of UDP ports using the IP address of one of the host's network interfaces. As the outgoing traffic from this PoundCloud instance is directed to the cloud VM, the host computer is configured to forward this traffic towards the VM emulating the network.

4.3.2 Network Emulation

We use a Linux VM instantiated in the host computer to emulate the communication network. Two virtual network interfaces are used to represent the network endpoints (i.e., one is linked to the robot and the other one to the cloud). Linux kernel's routing tables are used to implement the policy routing responsible for forwarding the packets arriving at both interfaces to the desired recipients. This can be configured by the user through the ip command from the *iproute2* package.

We take advantage of the network traffic control offered by the Linux kernel's network stack. The traffic control can act on queues of the outgoing traffic to control and shape how data is sent. A queueing discipline, or *qdisc*, is the scheduling code that is attached to a network interface to drop, forward, queue, delay or re-order packets. The user can configure and manage traffic control *qdiscs* via the command line using the *tc* command from the *iproute2* package.

We use the *NetEm*[11] (Network Emulator) *qdisc* to add delay and packet loss to outgoing packets in the network interfaces. *NetEm* allow us to insert delay according to a given distribution, which can be one of the built-in distributions (i.e., *normal*, *pareto*, or *paretonormal*) or even a custom distribution created using real data. Once the distribution is set, it is possible to tune its mean value, variance, and even add correlation to the delay of sequential packets. *NetEm* also allow us to set a random probability for packet loss

[11] Website: <https://man7.org/linux/man-pages/man8/tc-netem.8.html>. Accessed on December 1, 2020.

on the outgoing queue (i.e., packets leaving the network interface are dropped at random following a given probability).

Using *NetEm* is a simple yet powerful way to provide flexibility to the experimentation so that we can consider different situations and architectures. For example, by examining the characteristics of a communication flow among two locations over the Internet, one may use measures of delay and loss to cast a simplified model for that specific scenario. It is also possible to achieve more complex emulation models by using other features of *NetEm*, such as packet corruption, disorder, and bandwidth limit, or even by also using other queue disciplines at the same time. Nevertheless, considering that the main objective of this testbed implementation is to validate the PoundCloud operation, we consider the use of *NetEm*'s delay and loss features to be sufficient.

4.3.3 Cloud Emulation

We instantiate a VM to play the role of the cloud. This emulates a single VM running in the cloud, which is a rough simplification of a cloud operation and is incapable of approximating most features of the cloud computing paradigm. Consequently, one is not able to massively parallelize processes nor to explore distributed architectures in this approach. Nevertheless, this is sufficient if the experiment to be performed accommodates two assumptions: cloud computing techniques is not the main subject of the experiment, and; a single VM in the cloud can perform all the processing offload required by the application in question.

The remote ROS system and PoundCloud are instantiated in the VM. The remote PoundCloud node mirrors the local one to publish and subscribe to the topics of interest to the remote ROS system, completing the PoundCloud communication link and thus connecting local and remote ROS systems. As it can be seen in Fig. 25, all the outgoing traffic from this VM is directed to the emulate robot in the host computer via the emulated network.

4.3.4 Experimental Protocol

We use our testbed to perform two different sets of experiments to validate the capability of the PoundCloud of communicating two independent ROS systems. First, we perform a set of baseline measurements to verify the PoundCloud's response to multiple transmission requirements. Then, we build an autonomous navigation experiment in which the controllers are instantiated in the cloud to verify the PoundCloud's ability to serve a common robotic application.

Baseline Measurements: We design this experiment to assess the PoundCloud's capability of transmitting information amongst two ROS systems and to further understand its limitations. The focus here is on the practical aspects of the PoundCloud operation

and its implications for robotic applications. We generate artificial traffic of ROS messages to a given set of topics in the local ROS system under different conditions and measure the loss of messages at the remote ROS system. This is done by comparing the number of messages transmitted by the local ROS system and the number of messages received by the remote ROS system as recorded by the *rosbag* tool. We instantiate our testbed without any network constraints to evaluate the PoundCloud under ideal conditions.

In this experiment, we present two scenarios in which the local PoundCloud node subscribes to a set of topics and transmits to the remote PoundCloud node the messages published to these topics. To contemplate different flow conditions, in the first scenario we publish a single topic, and thus a single flow is transmitted, whereas in the second scenario we publish to 5 topics to evaluate flow concurrency. Multiple traffic conditions are evaluated by varying the size of the messages and their publish rate.

In both scenarios, all of the published messages are of type *std_msgs/Float64MultiArray*, which is composed of C-type *Float 64* arrays; we vary the payload sizes of such messages from 1 kB to 16 kB. We also vary the overall message transmission rate from 10 Hz to 200 Hz[12]. We consider seven values for message size and also seven values for message frequency, leading to a total of 49 $(size, rate)$ measurement points. This range encompasses transmission bandwidths from $1[kB].10[Hz] = 80kbps$ to $16[kB].200[Hz] = 26,6Mbps$ [13]. In each scenario, we observe each of these $(size, rate)$ operating points during 30 s and measure the total message loss. Two additional observations are made as we also measure the message loss considering a single flow of 64 kB messages published at 15 and 30 Hz to emulate the transmission of larger pieces of information such as video frames.

Autonomous Navigation Experiment: We design a experiment in which the robot must navigate inside a building to reach a series of goal positions to evaluate the PoundCloud operation on a common robotic task. To further explore the effects of the network into the autonomous navigation control loop, we repeat the experiment considering different RTT conditions.

We upload to Gazebo a simplified model of a building from the Colombian School of Engineering Julio Garavito based on a map previously acquired using the ROS navigation stack. Inside this building model, we determine a sequence of nine goal-positions forming a closed path with approximately 41 m of length. In our simulations, we use a model of the Clearpath Robotics' Jackal with a simulated SICK LMS111 LRF placed on top of it. The Jackal is a small differential wheeled robot and its model advertises the robot's

[12] I.e., when considering the overall transmission rate of 200 Hz in the first scenario, messages are published to the single topic at 200 Hz; in the second scenario, messages for each topic are published at $200/5 = 40$ Hz to achieve the overall 200 Hz transmission rate. This fixes the transmission bandwidth in each of the measured points.

[13] This range of overall bandwidth, message size, and message frequency encompasses the operation requirements of most sensors and common robotic messages exchanged within ROS. For instance, LRF messages usually range from 2 kB to 5 kB, depending on configuration and angle resolution, and are often published at 10 or 25 Hz, whereas odometry and velocity messages are more frequent but considerably smaller.

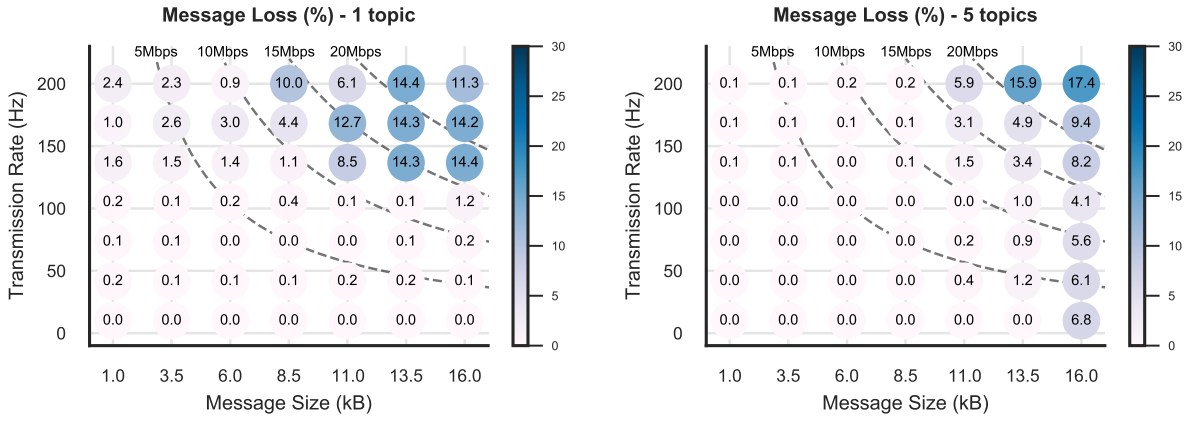
odometry based on the fusion of simulated encoders and IMUs.

We use the *move_base* package from the ROS navigation stack to command the navigation. The *move_base* node is executed in the cloud and uses coordinate frame data (i.e., ROS /tf topic), odometry data, and laser data to evaluate the surroundings and generate velocity commands to allow the robot to reach its objectives. The velocity commands are generated at 30 Hz while odometry and laser scans are generated at 50 Hz. The /tf topic fluctuates near a 100 Hz rate. The PoundCloud is used to manage robot-cloud communication and is configured accordingly.

We establish five different network conditions for RTT, being those a baseline condition without latency and other four conditions with RTT of 50 ms, 100 ms, 150 ms, and 200 ms. We perform a total of seven trials considering each network condition and during each trial we measure the total time for task completion and the total message loss.

4.3.5 Results

Baseline Measurements: Considering the two scenarios, we measured message loss as a percentage of the total transmitted messages under 49 different transmission conditions, as described in the previous section. Figure 26 displays the average loss values in each of the $(size, rate)$ pairs measured in both scenarios. In each measurement point, we display the numerical loss value on top of a marker colored according to a color map to ease the visualization of the results. We also plot bandwidth contour lines to provide insight of the actual transmission requirements[14].



(a) (b)

Figure 26 – Measured message loss percentage with varying message size and transmission rates: a) first scenario, considering a single topic being transmitted, and; b) second scenario, when five topics are transmitted simultaneously. The dashed contour lines indicate corresponding bandwidths.

[14] Note that, as it is customary, we use the *Byte* (e.g., B, kB) as payload unit and the *bit per second* (e.g., Mbps, Gbps) when representing bandwidth.

The loss remains negligible in both scenarios for a large part of our measurement range, which indicates that the PoundCloud is suitable for transmitting data generated from a large range of sensors and algorithms. At first glance, the results presented in Fig. 26 may seem unintuitive as the message loss decreases from the first to the second scenario despite the increase in the number of concurrent flows. Nevertheless, different factors must be considered when analyzing the data, such as bandwidth and individual flow rate.

In the first scenario, message loss is negligible when considering transmission rates up to 100 Hz despite any increases in message size. Our additional measurements of single flow transmission of 64 kB messages at 30 and 60 Hz yielded losses of 0.1 and 0.3 %, respectively, indicating that message size alone does not lead to the loss of messages under ideal communication conditions. This indicates that the PoundCloud can be used to transmit larger messages such as video frames and depth data.

When considering a single flow, message loss increases with bandwidth, remaining low up to 10 $Mbps$ and rapidly increasing after that. This is likely due to the underlying Pound operation and its associated limitations. For instance, Tardioli *et al.* [125] presents bandwidth measurements for the original Pound and observes losses of approximately 6 % when transmitting 6.5 $Mbps$ considering messages varying from 1 kB to 64 kB. In our measurements we do not observe a similar behavior, but rather considerably smaller losses when transmitting at bandwidth values up to 10 $Mbps$[15]. As an additional remark, message loss also increases with the flow rate even for low bandwidths. Nevertheless, in practice, very few applications demand such high rate transmissions and thus the bandwidth is the preponderant factor for such applications.

In the second scenario, as shown in Fig. 26(b), losses are near zero when considering messages smaller than 16 kB and a transmission bandwidth of 10 $Mbps$. Despite the concurrency of five flows, the overall transmission rate has a little impact over the loss, especially for smaller messages, given the lower rate of each flow. Thus, message loss also increases with bandwidth in the second scenario. When analyzing the measurements over the lowest rate considered, 10 Hz, there is a sudden peak in loss when increasing message size from 13.5 kB to 16 kB despite the low bandwidth of $\approx$1.3 $Mbps$. This seems to indicate that the performance of the Pound's packet scheduler decreases when concurrent flows require larger payloads, despite low bandwidth. Although not a problem for most applications, this result indicates that one must be careful when transmitting large messages to multiple topics (e.g., sending to the cloud data from several cameras).

Autonomous Navigation Experiment: All 35 trials were conducted successfully regardless of the RTT considered. Once again, we measure message loss by comparing the number of messages received and published on both PoundCloud instances in each trial.

[15] It must be noted that the measurements presented by Tardioli *et al.* [125] considered a non-ideal scenario including communication over WiFi, which increases the likelihood of message loss. Nevertheless, Tardioli *et al.* [125] observed an increase in message loss for a fixed bandwidth as transmission rate increases. Such an effect can also be observed over the 5 $Mbps$ contour line in Fig. 26(a).

Figure 27 displays the loss distribution observed considering the transmitted topics and the different network scenarios evaluated.

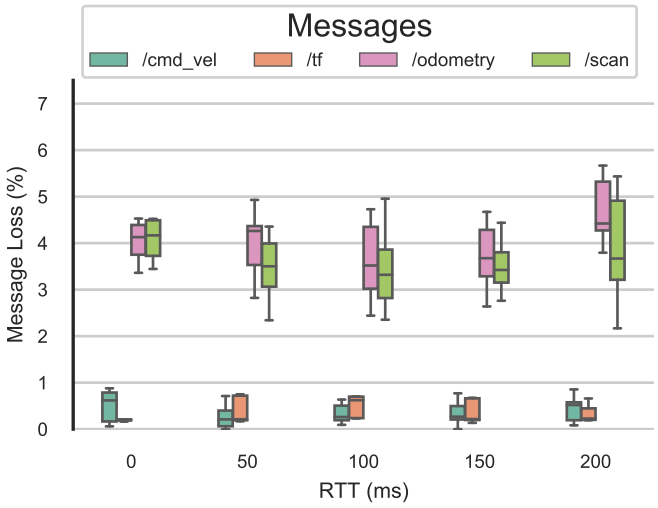


Figure 27 – Distribution of message loss rate for each transmitted topic.

As seen in Fig. 27, the message loss rates are similar during all of the experiments. This is to be expected as we only inserted latency, and no packet loss, into our simulated network. The velocity commands and /tf messages present a constant $sub - 1$ % loss which is compatible with the findings from our previous experiment regarding baseline measurements for the PoundCloud. The velocity commands messages present a payload of 132 B whereas /tf messages' size fluctuates from 132 B to 524 B. Despite the differences in the sizes of the odometry and laser scan messages – 797 B and 5920 B, respectively – their loss rates are similar. Although these results may differ from what we observed in our baseline measurements, it likely approximates the inherent losses of the PoundCloud to be observed in practical implementations.

Regarding the task completion times, we measure the time passed from the moment the robot starts to move until it reaches the last goal. The average completion time was 112.54 s considering all trials and increased RTT had no apparent effect on such a metric. We perform a Student's T-test to evaluate the completion time distribution measured for each network condition considering a 0.05 significance level. The results point that there are no statistically significant differences on such distributions, suggesting that the increase of RTT did not affect the overall time for task completion on our experiment. Although this result may seem unintuitive at a first glance, the worst RTT condition (i.e., 200 ms) places the information arriving at the controller only a few samples behind current sensor measurements. Given the static nature of the simulated environment and the Jackal's low maximum velocity (i.e., 0.5 m/s), the controller is still able to plan the movement with reasonably up-to-date information. This result also suggests that such a practical experiment may be possible even if the robot and cloud are in somewhat distant countries.

4.4 Conclusions

In this chapter, we placed the ROS as a viable standard middleware for robotics and identified a software package for communicating distributed ROS systems, the Pound. Then, we presented the PoundCloud framework, which we constructed on top of the Pound to achieve robot-cloud communication over the public Internet, and discussed its implications within the cloud robotics ecosystem. Finally, we validated the PoundCloud on an emulated testbed to verify its operation and potential limitations. Our experiments' results indicate that the PoundCloud can be used to extend systems based on the ROS middleware to incorporate cloud robotics concepts. The preliminary measurements of the experimental performance show that the PoundCloud is suitable for a large range of robotics applications. sIn Chapter 5, we present a methodology for cloud robotics experimentation including the PoundCloud as the communication framework, and validate it on a set of practical experiments.

5 A Common Methodology for Cloud Robotics Experimentation

This chapter presents a methodology for cloud robotics implementation based on open-source software and commercial off-the-shelf devices. To validate our methodology, we present two use cases representing a set of common robotics tasks. In such use cases, most of the computation is carried out remotely and we perform a series of experiments to demonstrate our technique. In general, our results point to the feasibility of the presented approach in different classes of applications even under non-ideal network and cloud settings. This indicates that our methodology and framework can be adopted by other researchers and practitioners to replicate our results and to guide general cloud robotics experimentation.

5.1 Towards a Open Cloud Robotics Methodology

We propose an open cloud robotics approach to foster the adoption of the cloud robotics paradigm and to motivate discussions on implementation details and experimentation. We envision the possibility of implementing cloud robotics systems based on open-source software and commercial-off-the-shelf devices. On the research side, such systems should lead to common practices and reproducible works. The development ecosystem may also benefit from the use of open and accessible tools, enabling more actors to get involved in the subject.

Figure 28 illustrates a generic reference architecture for cloud robotics. The robot is coordinated by multiple self-interacting agents. A *Task Manager* is in charge of overseeing the task execution and deciding if it is necessary to offload part of the task to the cloud. The offloading decision weighs the costs associated with processing and transmission to conclude whether to conduct processing tasks locally or at the cloud [15, 126]. It is also possible that part of the computational tasks cannot be conducted locally, thus demanding the use of a cloud-based service. In any case, if the cloud is necessary, a *Communication Manager* should start communication with a cloud platform requesting the service. After the service is started, the robot is directly served by the service application on the cloud.

In Fig. 28, remote and edge cloud platforms are displayed to exemplify how such platforms can be used in cloud robotics. We consider that an edge cloud platform (from now on referred to simply as edge) is physically closer to the robots and is capable of offering lower communication latency – which is important for time-sensitive applications such as real-time control – although usually with limited resources when compared to a remote – or core – cloud platform [32]. As Chen *et al.* [127] show, edge platforms are

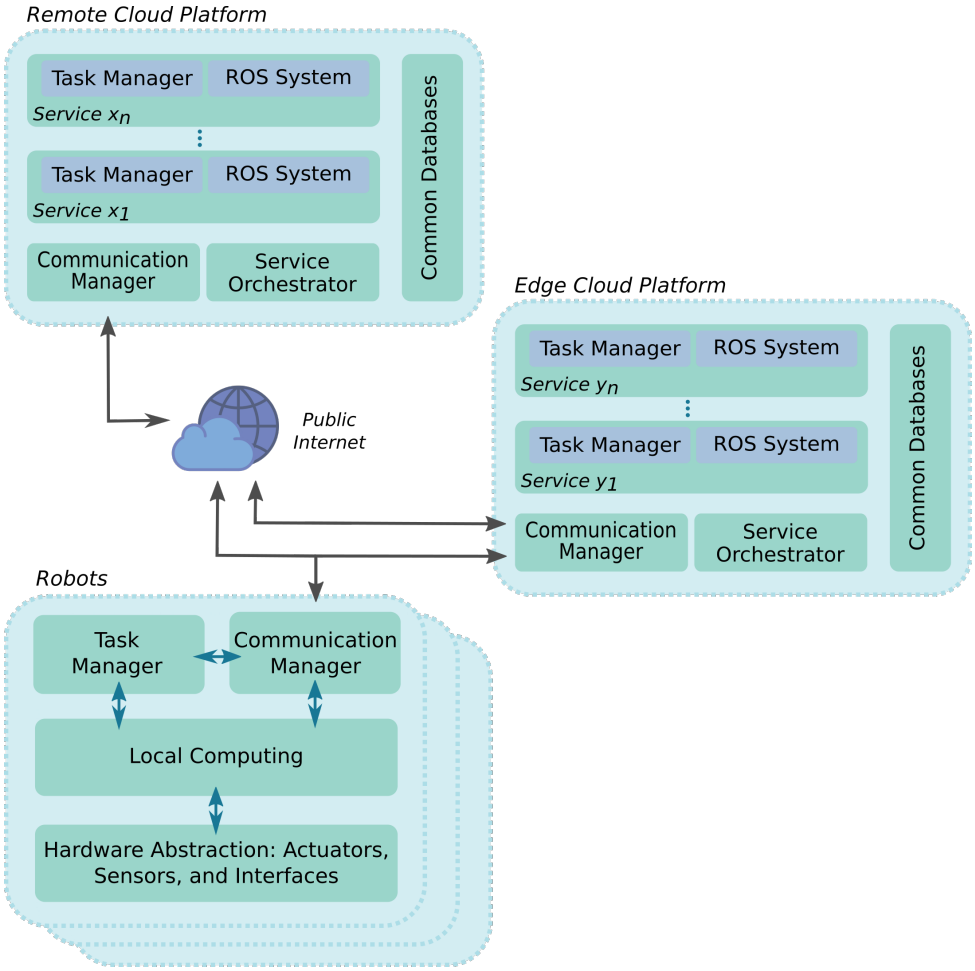


Figure 28 – Generic reference model for cloud robotics architectures.

locally implemented and can be reached without the need of passing by the public Internet. A remote cloud platform (to be referred to as cloud), on the other hand, is likely to present increased latency and more compute and storage resources, making it suitable for higher-level applications such as planning, mapping, and context extraction. Cloud and edge may work together by delegating specific services for each of them (e.g., the edge manages the robot navigation while the cloud is responsible for the path planning, or the edge pre-process sensor data and sends it to the cloud to decrease network load) [32, 127].

In Fig. 28, when a service request reaches either the edge or the cloud, the *service orchestrator* is responsible for instantiating and managing the services. By representing robotic applications as services, Fig. 28 suggests the use of container technology in the cloud although structures based on VMs can also be employed. Containerization techniques are commonly employed as it suits service-oriented architectures, which are widespread in edge implementations [128]. Thus, technologies such as Docker and Kubernetes are commonly used for service orchestration.

According to Andiappan & Wan [129], a methodology can be defined as the general strategy to solve a problem. Thus, a methodology guides the practitioner within a set of boundaries. In other words, once a methodology is defined, suitable methods can be

chosen to develop a system. In this thesis, we focus on a common methodology to enable the use of cloud-based services to enhance robots' capabilities. Our methodology is based on:

- The use of commercial off-the-shelf devices. Using readily-available devices allows for the direct reproduction of a given implementation and also for clear comparisons between different works;
- The use of open-source software, preferably based on ROS. Due to its open nature, open-source software is readily available to anyone, allowing for direct reproduction of pieces of software. Furthermore, it provides a starting point for anyone who wants to build functionality on top of existing software. Using ROS allows for a wide range of practitioners to incorporate such pieces of software into their systems;
- Cloud platforms providing the IaaS. Commercial SaaS and PaaS providers may offer proprietary solutions that cannot be easily migrated to other platforms. Since we aim at designing cloud robotics services that can be platform agnostic, the use of IaaS allows for easy setup replication.

In the remainder of this chapter, we implement a cloud robotics system according to our methodology and validate it considering two experimental use cases.

5.2 Robotic Platform for Experimentation

Wheeled robots and autonomous unmanned ground vehicles (UGV) are of widespread use and we decided to use a popular robotic platform within this category. Thus, the Pioneer LX research platform (Omrom Adept Technologies, U.S.A.) is the UGV used in this work (see Fig. 29). The robot's kinematic structure consists of a differential drive configuration with two front caster wheels and two rear caster wheels. Sensory and actuation systems constitute the platform main capabilities:

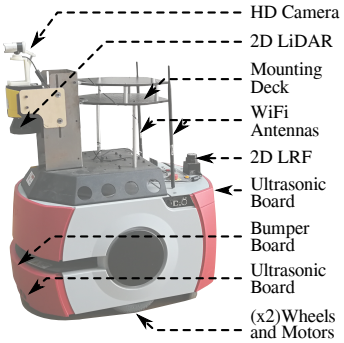


Figure 29 – The robotic platform: sensors and actuation system of the Pioneer LX robotic platform used in the study.

- *Odometry estimation.* Encoders at each motorized wheel and an internal Inertial Measurement Unit (IMU) are used to estimate UGV's position.
- *Environment sensing and obstacle detection.* Internal safetu systems of the platform are fed by a set of range sensors. A front 2D Light Detection and Ranging sensor (LiDAR) and an LRF are used for environment sensing and obstacle presence estimation. Two ultrasonic boards are used for low-rise obstacle detection. An HD web camera is integrated to identify people in the environment and for remote teleoperation purposes.
- *CPU.* An on-board computer (Intel Dual Core 1.8 GHz Atom, 2 GB RAM) running Ubuntu/Linux provides support for software development. Moreover, ROS Indigo is installed on it.

The UVG is located at a university campus (Escuela Colombiana de Ingeniaría Julio Garavito, Colombia) connected to the Internet via a commercial ISP. Although several techniques have been proposed to deal with wireless communication problems in mobile robots (e.g., SDN for reliable access point handover [19]), we chose to deploy our system using standard WiFi. Therefore, we create a local network using a 2.4 GHz 802.11n access point (AP). Moreover, the presence of multiple wireless devices on campus results in inevitable interference – especially when working in the 2.4 GHz band. Thus, we evaluate our system in a non-ideal and easy-to-reproduce scenario. Nevertheless, there are two assumptions: i) the AP coverage area is large enough to cover the robot's entire workspace; and ii) the robot is already pre-configured to automatically connect to the AP. An Archer T2U (Tp-link, China) USB adapter working on the 2.4 GHz band is used to provide wireless connectivity to the UGV.

5.3 Cloud Platforms for Experimentation

Different kinds of cloud platforms may be desirable in distinct situations or applications. In general, the physical distance separating robot and cloud plays an important role both in communication latency and network complexity and must be taken into account beforehand. Thus, even though commercial cloud providers operate in multiple locations, one must consider multiple cloud platforms when evaluating the impacts of employing cloud-based services. Our system is integrated into three types of cloud platforms. We consider a commercial cloud computing provider, using the developing tools provided by the company; a private cloud managed by our laboratory's team; and an emulated edge cloud platform. Similar setups are provisioned in each of the cloud platforms, which are described as follows:

Commercial Cloud. The Google Cloud Platform is chosen as our commercial cloud provider. Using the Compute Engine, we instantiate a VM (8 vCPUs, 16 GB RAM) on a

datacenter located on the United States' west coast. We installed the Ubuntu Server 16.04 OS with ROS Kinetic and instantiated our cloud services. The VM possesses its public IP and the estimated straight-line distance separating it from the UVG's location is above 5500 km.

Private Cloud. We used the cloud platform provided by the FUTEBOL's testbed for experimentation on real-time Industry 4.0 applications [14]. Such a testbed is located at the UFES and managed by UFES' NERDS. This cloud computing platform is implemented on a datacenter using the OpenStack multi-node version Pike. One of the major features of such a cloud platform is its ability to automatically perform live vertical scaling (i.e., increasing or decreasing the use of resources on a VM without the need for restarting it), which, by the time of the writing of this thesis, is a feature often unavailable in commercial cloud platforms such as Google's or Amazon's. This is of special importance in long-term cloud robotics tasks requiring a variable amount of computing resources over time [130]. We instantiate a VM (8 vCPUs, 16 GB RAM) and install the Ubuntu Server 16.04 glsos with ROS Kinetic and our cloud services. The VM is reached via the laboratory's public IP and the estimated straight-line distance separating it from the UVG's location is above 4600 km. In the remainder of this work, our private cloud setup will be referred to as the FUTEBOL cloud.

Edge Cloud. Due to the envisioned benefits of fog and edge computing for cloud robotics, we use a laptop (Intel i7-7700HQ, 8 threads, 16 GB RAM) with Ubuntu Desktop 16.04 glsos installed to emulate a VM on an edge cloud setup. As the laptop's available resources match the other instantiated cloud VMs, we chose to not virtualize resources and instead use the laptop itself as an emulated VM in a edge cloud platform, resulting in similar specifications throughout all three computing setups.[1] The laptop is in the same LAN as the UGV connected via cable to a router, which greatly mitigates network-related issues. Considering that the computational resources are available in the LAN, the standard ROS-Pound approach would suffice. Nevertheless, the system is configured in the same way in all our cloud platforms. For the sake of simplicity, from now on we refer to the laptop as a VM in an edge cloud setup.

5.4 Cloud-Based Services for Robotic Tasks

Ideally, cloud services for robots should be – at least – modular and independent both from the cloud platform and the robot. By leveraging ROS and the PoundCloud, such requirements can be easily satisfied, as a ROS-based service would take as input messages published to a given set of topics and answer by publishing messages into another set of

[1] In other words, the robotic applications are executed directly on top of the laptop's OS, without the overheads introduced by virtualized environments.

topics. To demonstrate our system, we implement cloud-based services considering two important use cases for mobile robots: HRI and autonomous navigation.

Despite decades of research, HRI is still a growing research field facing several challenges. As safety is a major concern, several works have been conducted over simulations and augmented-reality [131]. Another layer of complexity is imposed when the human is inserted within the control loop of the robot, in tasks where the proper interaction is of critical importance [32, 132]. This complexity is escalated when a network – or the Internet itself – is also inserted in the control loop. Thus, we present in Section 5.5 the implementation of a follow-in-front service, in which a robot and a human must work together to accomplish a task.

Autonomous navigation is another field of increasing relevance as mobile robots are being ubiquitously employed in practical contexts such as factory and logistics automation [42, 133]. Moreover, navigation in unstructured environments remains a challenge and it is often addressed by leveraging multiple sensors and complex algorithms [134]. Thus, the centralization of information at the cloud may allow for improved planning systems for single robots and also for better coordination among groups of robots [2, 42]. In Section 5.6, we present the implementation of an autonomous navigation service to remotely control a robot towards a series of goal positions.

Given a task and the overall architecture of the system, to implement a cloud-based service one must first decide which processing tasks ought be kept in the robot and which can be delegated to the cloud. To cope with the objectives of this work, as a design decision we chose to maximize the amount of processing delegated to the cloud, thus keeping at the robot only the essential and leaving to the cloud the generation of the control commands. In Sections 5.5.1 and 5.6.1 we describe the general operation of the distributed controllers and our approach at designing such services.

5.5 Case Study 1: Human-Robot Interaction

In this section, we describe the implementation of an HRI service and perform a set of experiments on top of it to evaluate the impacts and potential benefits of migrating to the cloud processing components responsible for managing the interaction amongst a human and a robot. In the following subsections, we detail the service and the experiment, and present the obtained results.

5.5.1 Follow-in-front Service

The development of techniques for natural interaction among humans and robots is still an open research field. For instance, certain tasks require a mobile robot to cooperate with a human operator by following the operator's lead [135]. The difficulty level of this

kind of task is escalated when the robot must follow in front of the person and the robot must not only track the movements of the human but also anticipate them to not deviate from the desired trajectory. Most of the works dealing with this problem implement some way of predicting changes in movement direction by tracking the human torso's natural inclination, as in Cifuentes *et al.* [8] and Hu, Wang, & Ho [135]. Nevertheless, such approaches may be error-prone in unstructured environments and, in some situations, it may be useful to provide some level of control to the human operator.

Scheidegger *et al.* [115] proposed a follow-in-front controller that maintains the robot ahead of the human while allowing control of the steering via face commands. We'll now briefly explain the operation of the controller before describing the proposed architecture and implementation of the service. An in-depth explanation of the follow-in-front controller can be found in Scheidegger *et al.* [115] whereas here we describe the controller in general terms. In summary, there are two communication channels: a visual channel, leveraging video acquired by a camera pointing towards the face of the operator; and an active ranging channel, based on data measured by an LRF sensor directed towards the operator's lower limbs.

The robot tracks the operator via a leg detection system that uses an LRF and a clustering algorithm to identify the operator's legs. At the same time, the video stream is processed to estimate the orientation of the user's face with respect to the robot. A face detection system identifies standard landmarks on the operator's face and then estimates the pose of the face. The follow-in-front controller inputs are (i) the human position with respect to the robot – as gathered by the leg detection system – and (ii) the human face horizontal orientation. The robot's linear and angular velocities are generated using a formation controller, to keep the robot ahead of the user, and a steering controller, which minimizes the horizontal angle provided by the face detection system. Thus, the human can use his/her face to command the steering of the robot, which stays ahead of the person by a given distance[2].

Figure 30(a) illustrates the general blocks that are implemented at the robot. Data gathered by the sensors that compose the interaction channels are directly sent to the cloud without any pre-processing. The incoming data (i.e., the remotely generated control signal) feed the *Velocity Supervisor*, which monitors the consistency of those values before transmitting them to the *Actuation System*. Moreover, a *Safety System* monitors the ranging sensors, such as the LRF and sonars, to detect nearby obstacles and flag the *Velocity Supervisor* to stop the actuators. Figure 30(b) illustrates the general blocks that compose the cloud service. Data gathered by the robot's sensors are processed in the leg and face detection systems to feed the follow-in-front controller. The generated control

[2] As shown by Scheidegger *et al.* [115], the steering controllability (i.e., the responsiveness of the controller to steering commands) is enhanced as the distance between the person and the robot decreases. Given a unicycle robot, the person should ideally be at the center of rotation of the robot, which is impractical considering the robotic platform used in this work.

signals are then sent to the robot to close the distributed control loop.

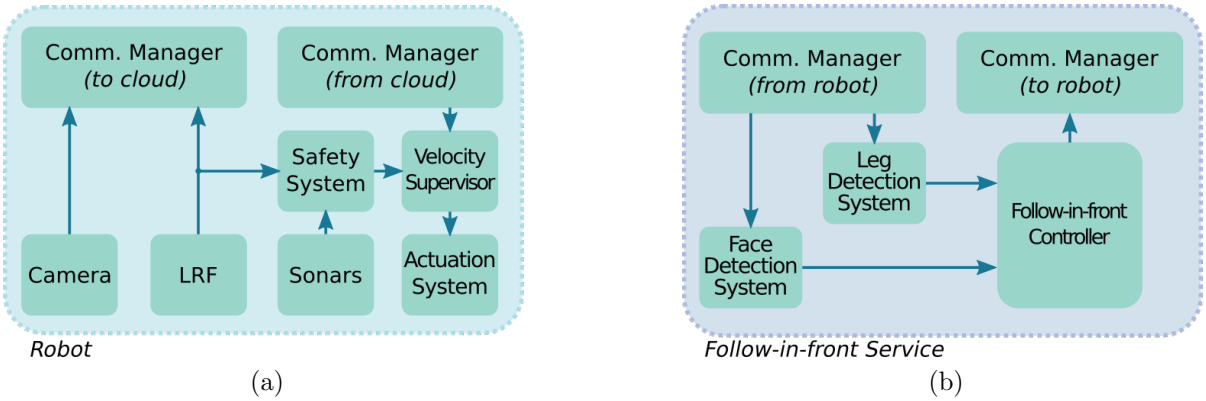


Figure 30 – The building blocks of the follow-in-front service: a) components and processing tasks performed at the robot; b) controller and it's processing modules executed by the cloud service.

5.5.2 Experimental Protocol

The follow-in-front service generates control signals to maintain the robot ahead of the human operator based on the information gathered by the robot's interaction channels. Thus, the robotic task is defined as following the human along a predefined path marked on the floor – and completely unknown to the robot. The marks on the floor are used as a standard guide for the operator throughout the whole experiment. The path is formed by a Lemniscate of Bernoulli defined by a parameter $a = 2.5\ m$. This kind of path is commonly used in the validation of path-following controllers given its symmetry and its challenging curvatures. Moreover, the lemniscate prevents sudden changes in direction of movement and demands displacement in different horizontal and vertical orientations [136].

The experiment is composed of three parts and a total of fifteen trials, five trials per part. In the first part, we design a baseline scenario and all computation is performed at the UGV's embedded hardware to extract the baseline performance. The other two parts are performed instantiating the service at our edge cloud and at the commercial cloud platform, respectively. A single individual is previously trained to operate the robot and is responsible for participating in all trials.

From prior experimentation, we know that the *Follow-in-front Service* demands at least the processing of five video frames per second (FPS) in the *Face Detection System* for proper command extraction and HRI. Nevertheless, the Pioneer LX embedded hardware was unable to process more than one FPS in our preliminary trials. Thus, to execute the first part of the experiment, we temporarily upgrade the Pioneer LX by embedding a Raspberry Pi 3 model B3 (Quad Core 1.2 GHz, 1 GB RAM) into the system. The rationale is that the Raspberry Pi is low cost yet powerful enough to satisfy the processing

requirements without distorting the system's processing capacity when compared with other commercial UGV's hardware. Only the *Face Detection System* is executed in the Raspberry Pi, which can maintain a stable processing rate of 5 FPS. Finally, the frequency of the *Face Detection System* is set to 5 Hz (i.e., to process 5 FPS) during the first part of the experiment and set to 15 Hz during the following two parts.

We are interested in observing: i) if the follow-in-front task can be properly accomplished, and; ii) how the system performance is affected by the remote processing of the interaction. The task is considered accomplished if the robot stays ahead of the operator and completes the path. Large deviations or inconsistencies in control responses might lead to the interruption of the task. To verify how the distributed control affects performance, we store metrics related to hardware usage, to network parameters, and to the robot's actions. The collected metrics are listed:

- *Time to completion.* Total task time, in seconds.
- *Average speed.* Robot's average linear speed, in meters per second.
- *Kinematic Tracking Error (KTE).* Given by Equation 5.1, the output value increases with the performed path average error, when compared with the reference path, and its variance; the smaller the value, the better the tracking.
- *CPU and memory usage.* Total hardware load, in percentage, observed every second during the task.
- *Network latency.* Average RTT, in ms, of packets from the robot to the cloud, using the standard *ping* tool.
- *Throughput.* Uplink and downlink bandwidth usage, in bits per second, measured at the robot's wireless interface and the cloud's VM logical network interface.
- *Message loss rate.* Calculated by comparing the number of messages (e.g., data from the last odometry or laser scan reading) that leave a given node (i.e., the robot or the cloud) and arrive at another node; message loss rate is used instead of the standard packet loss rate as the former is more closely related to the application.
- *Message jitter.* Given the period a message is generated (e.g., the UGV's pose is estimated, and broadcasted, every 100 ms), the jitter distribution is obtained observing the period of arrival of these messages at the destination node.

The KTE is an objective tracking measurement in which the desired path is compared to the performed one [137]. The KTE outputs a number that increases with the the mean error and its variance and can be obtained using Equation 5.1:

$$KTE = \sqrt{\left|\bar{\varepsilon}\right|^2 + \sigma^2}, \tag{5.1}$$

in which $|\bar{\varepsilon}|$ is the absolute mean error between the discrete points that compose the performed and desired path, and σ^2 is the variance of the error distribution. Thus, the KTE takes into account not only stationary errors on the path following, but also its oscillations, directly reflecting the struggle in keeping the robot on track.

Table 2 summarizes the types of messages exchanged among the robot and cloud platform highlighting the associated ROS topics. The */tf* topic, managed by the library *tf2*, transmits data regarding the reference coordinate frames and their translations into one another and is set at the highest priority in ROS Pound as these messages are important in most applications. The next two messages with the highest priority are the ones actually used by the cloud-based service: the compressed image and the scan readings from the LRF pointing to the operator's legs (i.e., */scan* topic). The cloud service responds with the velocity commands encapsulated within the */cmd_vel* topic. Finally, the topics associated with the *RosAria* namespace contains data from the Pioneer's onboard controller and other sensors and are sent to the cloud for logging purposes.

Table 2 – Case study 1: the distinct types of data exchanged during robot-cloud communication.

ROS Topic	Source	Message Size (Bytes)	Period (ms)	ROS Pound Priority	Description
/tf	UGV	125–220	≈ 10	1	Coordinate frame data
/usb_cam/image_raw/compressed	UGV	47,611	66	2	Data from camera
/scan	UGV	2,222	100	3	Rear LRF scan data
/RosAria/pose	UGV	745	100	4	Pose data
/RosAria/laserscan	UGV	2,284	120	5	Front LRF scan data
/RosAria/sonar	UGV	269	100	6	Data from sonars
/cmd_vel	Cloud	80	100	1	Velocity commands

5.5.3 Results

All trials were considered successful and the robot was capable of following ahead of the human operator while staying on the path. The mean time for task completion was 60.12 ± 4.0 s during the first part of the experiment, decreasing to 51.22 ± 1.3 s and to 53.91 ± 5.7 s during the second and third part, respectively. The observed mean linear velocity was similar during the first and third scenarios, 0.33 ± 0.74 m/s and 0.33 ± 0.81 m/s, respectively, and increased significantly during the edge cloud scenario, 0.37 ± 0.76 m/s. Moreover, a KTE value of 0.1217 ± 0.0007 was obtained during the first scenario, whereas 0.15404 ± 0.0029 and 0.16804 ± 0.0007 values were obtained during the second and third scenario, respectively. Smaller KTE values were obtained during the longer tests, while the KTE values increased in scenarios where the information was externally processed.

Table 3 displays the mean CPU and memory use in each device throughout the trials. The offloading of the computation tasks to the cloud reduces the UGV's CPU usage in

over 55 % without affecting much the memory load. This is a direct result of the controller nature, which mainly requires processing power. Offloading also helps in preventing CPU overloads that could lead to increased reaction times on safety and critical control loops. It is also interesting to note that the cloud VM is over-dimensioned: it could easily be resized to half of its CPU resources and a quarter of its memory resources, if not even more. Despite the lack of a standard to be followed, reasonably dimensioning VM resources can deeply impact operational costs of cloud robotics implementation over commercial platforms. The CPU usage observed in the edge can be explained by our implementation of the edge, as the laptop emulating the edge's VM also runs Ubuntu Desktop's graphical interface and computing resources are shared with other processes besides our service.

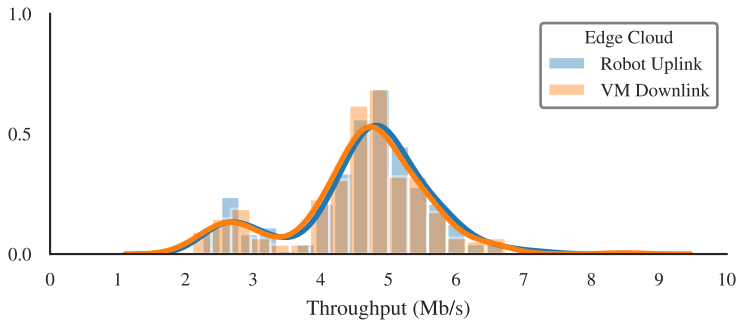
Table 3 – Case study 1: resource usage measured in each device in per scenario.

Scenario	Device	Mean CPU Usage (%)	Mean Memory Usage (MB)
Embedded Processing	UGV	85.6 ± 6.2	588 ± 12
	Raspberry	52.9 ± 4.7	284 ± 28
Edge Cloud Processing	UGV	37.2 ± 3.8	544 ± 8
	VM	40.3 ± 4.9	$2,064 \pm 48$
Commercial Cloud Processing	UGV	38.6 ± 4.7	534 ± 2
	VM	13.2 ± 2.6	960 ± 384

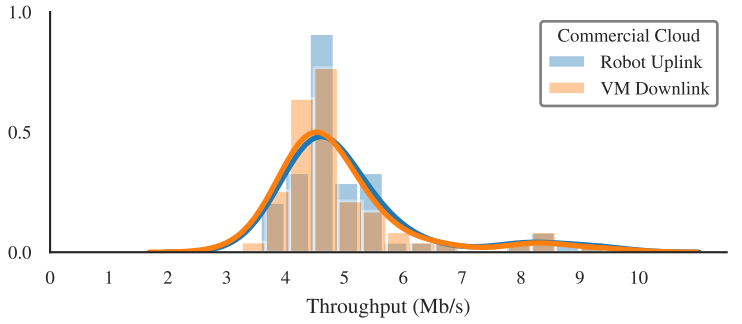
During the second part of the experiment, we performed a total of 2198 *ping* measurements from the UGV to the cloud platform, which averaged a RTT 9.9 ± 10.1 ms with a 0.81 % packet loss rate. The minimum and maximum measurements were of 1.2 ms and 284.5 ms. Though sparse and uncommon, such high RTT spikes could render unsuitable robotics-related network packets in some classes of applications [96]. In the third part of the experiment, there was a total of 2625 *ping* measurements, averaging a RTT of 124.1 ± 14.4 ms and a 0.23 % packet loss rate. The minimum and maximum RTT measurements were of 108.6 ms and $1,094.9$ ms.

In general, the robot-to-cloud traffic consumes about 150 times more bandwidth than the cloud-to-robot traffic. This discrepancy is expected as the robot must upload to the cloud the video stream and laser scans. One interesting way of looking into the uplink/downlink bandwidth during the last two parts of the experiment is by plotting the distribution of the traffic per second, as shown in Fig. 31. In both Fig. 31(a) and Fig. 31(b), the way that the distributions are shifted in the bandwidth axis indicates the amount of information loss in E2E communication. For instance, during the second part of the experiment, the observed median of the robot's uplink was of 4.64 Mb/s, while the median of the edge's downlink was of 4.54 Mb/s, representing two percent on general information loss. This value and distribution can be compared with the results shown in [15], which displayed more than 50 % loss on sensor data traffic when using WiFi and the standard ROS communication system. Regarding the third part of the experiment,

the robot's uplink was of 5.07 Mb/s, while the median of the cloud's downlink was of 4.95 Mb/s.



(a)



(b)

Figure 31 – Case study 1, throughput distribution: robot uplink and VM downlink observed considering the (a) edge and (b) commercial cloud scenarios.

Analyzing the bandwidth usage does not exactly reflect the loss of relevant information. Instead, we look directly into the application to evaluate how many ROS messages of each type are generated in a source machine and how many of those messages are arriving at the destination machine. Figure 32 shows the message loss rate observed during the experiment. Given the simplicity of the network structure linking the robot to our edge, the critical part of the network is the wireless communication channel and most of the losses are likely to be associated to it. As it would be expected, the loss rate of most messages increases during the third part of the experiment given the physical distance to the cloud and the network complexity associated with it. It is also interesting to note that despite the small payload associated with the */tf* messages, its losses were steadily higher than other messages'. This is caused by the higher frequency of these messages and their interaction with the queuing system implemented by the Pound tool.

The non-deterministic latency observed in real networks affects the period of arrival of the messages at the destination. Figure 33 displays the jitter distribution of the arriving messages that are relevant from a control point of view. The */tf* is not considered given that information related to different frames is published at different rates[3]. Data from

[3] The */tf* period information displayed in Table 2 is a rough approximation considering the total number

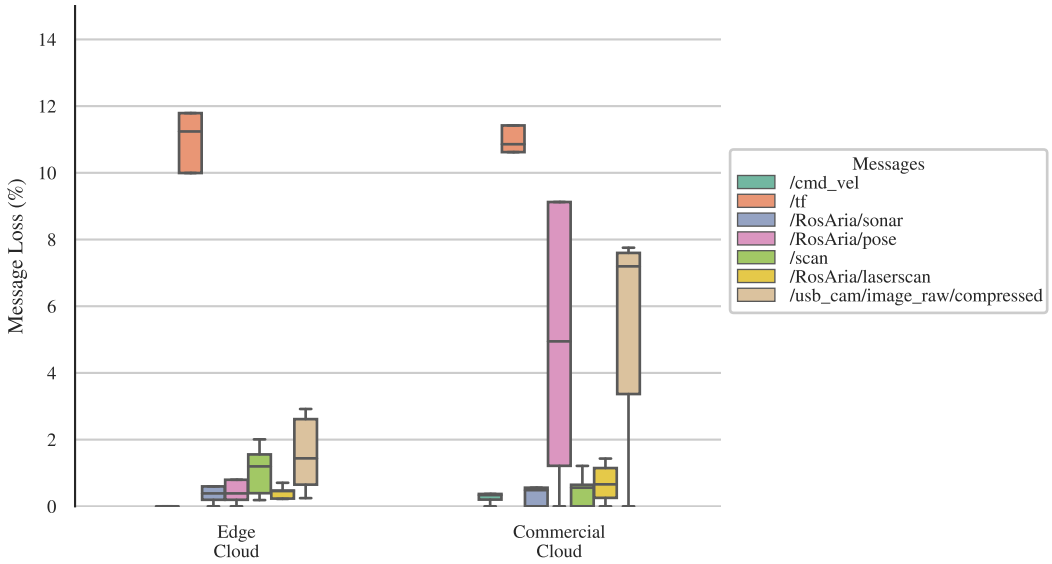


Figure 32 – Case study 1: distribution of message loss rate throughout the second and third parts of the experiment; messages are ordered, left-to-right, in ascending payload size.

the camera is subject to the largest jitter as it is transported in the largest packets, while the much smaller packets containing the laser scans and velocity commands arrive at the destination with near-ideal periodicity more frequently. Again, the increased network complexity when dealing with the remote cloud also impacts the jitter distributions.

5.6 Case Study 2: Autonomous Navigation

In the second case study, we implement an autonomous navigation service as stated in Section 5.4. Again, we performed an experiment to evaluate the impacts of applying cloud robotics concepts into this class of application. In the following subsections, we detail the service implementation, the experiment, and the obtained results.

5.6.1 Autonomous Navigation Service

Autonomous navigation is the basis for most tasks involving UGVs and thus, as a second use case, we chose to implement a cloud service responsible for controlling the robot's navigation throughout the environment. Given a task (e.g., reach a series of goal positions), the service processes the robot's sensors data to localize the robot, generate local maps, plan its trajectory, and calculate the reference velocities to feed the robot's low-level controllers. We use the standard ROS' *navigation* module to remotely process the data and control the robot.

of messages published in a given time period.

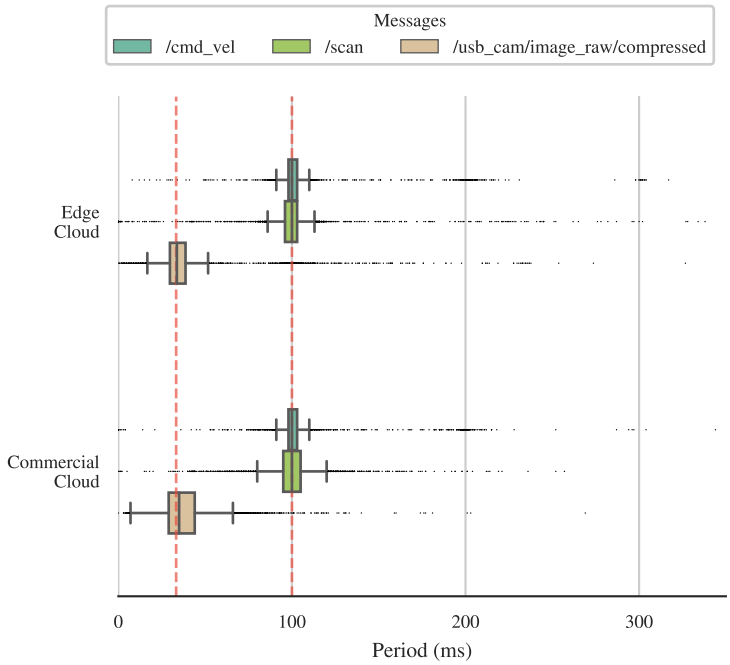


Figure 33 – Case study 1: jitter distribution of the arriving messages relevant to the robot's control during the second and third parts of the experiment. The red dashed lines indicate the ideal period values as indicated in Table 2.

Figure 34 illustrates the implemented architecture and the distribution of the processing modules. Once again, the robot's embedded computing is responsible only for the processing of critical modules such as the safety system and the velocity supervisor. The raw data gathered by the robot's sensors is sent to the cloud to be processed by the remote service. Then, the navigation service is responsible for motion planning and for generating the desired velocities to be followed by the robot. We define the navigation task as reaching a sequence of goal positions: when the current goal is reached, the next goal is used to feed the *Planner System*.

5.6.2 Experimental Protocol

The autonomous navigation task is defined by a sequence of goal positions that the UGV must reach. A total of nine goal positions are established forming a closed path with approximately 41 m of length in a semi-structured environment (see Figure 35). To assess the effects of moving obstacles in a controlled fashion, a person walks in front of the UGV after the first goal position is reached, momentously obstructing the robot's path towards the second goal. That's the only instance of a moving obstacle.

The cloud-based navigation service expects information gathered by the robot's front LRF, odometry, and pose estimation, as well as its locally calculated transform coordinates (i.e., ROS' */tf* topic). The cloud service must localize the UGV in such an environment and plan paths towards the goal positions to generate the velocity commands that are

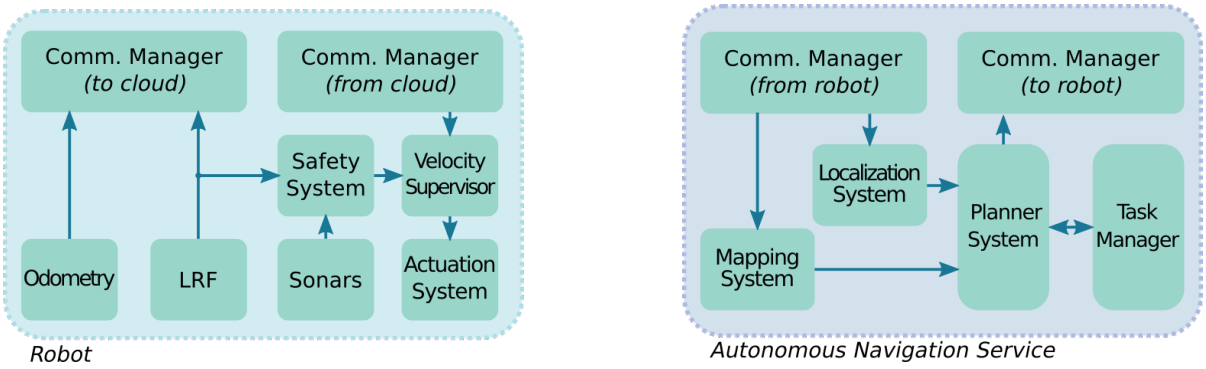


Figure 34 – The building blocks of the autonomous navigation service: a) components and processing tasks performed at the robot; b) navigation controller and it's processing modules executed by the cloud service.

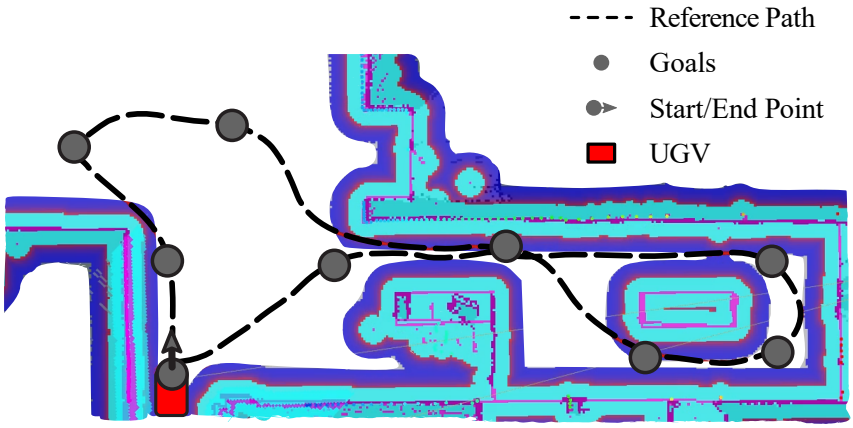


Figure 35 – Autonomous navigation task: the goal positions and a representation of the navigation path.

sent to the UGV.

The experiment is composed of four scenarios and a total of twenty trials, five trials per scenario. Once again, the first scenario is defined as the baseline scenario, and all computation is performed at the UGV's embedded hardware. The second and third scenarios are performed instantiating the service at our edge cloud and the commercial cloud platform, respectively. The last scenario makes use of the FUTEBOL cloud at Brazil to run the service.

The task is considered accomplished if the robot reaches all of its goals. We are interested in observing how quickly the robot can finish the task and how the performance of the navigation is affected by such distributed computation. It is of particular importance to register how the navigation behaves in a static environment and how it reacts to the sudden appearance of dynamic obstacles. Again, we store metrics related to hardware usage, to network parameters, and to the robot's actions to assess the effects of implementing our cloud robotics architecture. The collected metrics are the same as the ones discussed in

Section 5.5.2, the only difference being that the KTE is substituted by the total distance covered by the robot, in meters, during navigation. The distance metric is mainly affected by the planner's capability to compute optimal paths towards the goals.

Table 4 summarizes the messages exchanged among the robot and cloud platform. Now, only the data which is relevant to the controller is transmitted. Sonar data are still used by the robot's safety system but are not sent to the cloud. Once again, the */tf* topic transmits data regarding the coordinate frames is set at the highest priority in ROS Pound. The robot's pose and the front LRF scans are the other messages transmitted from the robot to the cloud. The cloud service responds with the velocity commands, the */cmd_vel* topic.

Table 4 – Case study 2: the distinct types of data exchanged during robot-cloud communication

ROS Topic	Source	Message Size (Bytes)	Period (ms)	Message Priority	Description
/tf	UGV	125–220	≈ 10	1	Coordinate frames data
/RosAria/pose	UGV	745	100	2	Pose data
/RosAria/laserscan	UGV	2,284	120	3	LRF scan data
/cmd_vel	Cloud	80	33	1	Velocity commands

Due to constraints set by the robot's embedded hardware, the control and planning frequencies of the navigation algorithms are much smaller than what is made possible by the cloud platform. Table 5 brings the navigation-related ROS packages used and their main parameters as configured in each scenario. These values were empirically defined as sufficient to illustrate our case study.

Table 5 – Case study 2: relevant differences in the configuration of navigation-related ROS packages.

ROS Package	Parameter Name	Embedded Processing (Hz)	Remote Procesing (Hz)
move_base	controller_frequency	6.0	30.0
	planner_frequency	0.4	30.0
costmap_2d	update_frequency (local)	4.0	40.0
	publish_frequency (local)	0.5	5.0
	update_frequency (global)	1.0	10.0
	publish_frequency (global)	0.4	4.0

5.6.3 Results

The UGV reached all of the goal points in all of the trials. The mean time for task completion was 128.7 ± 13.1 s during the first scenario, decreasing to 100.6 ± 3.6 s on the second scenario. When using the commercial cloud and FUTEBOL cloud, the completion time was 109.8 ± 9.0 s and 112.1 ± 8.1 s, respectively. The observed mean linear velocity

was of 0.33 ± 1.0 m/s, increasing to 0.39 ± 0.9 m/s, 0.38 ± 1.0 m/s, and 0.36 ± 0.9 m/s in the following three scenarios, respectively.

The quality of the path planning influences on the total distance covered by the robot during the execution of the task. Table 6 brings information regarding the average distance covered by the UGV during each scenario. The results are similar in all of the cloud-based scenarios, with the second scenario presenting the best performance. The distance covered by the UGV during the largest given the lower frequencies at the navigation control loop. In particular, this difference is mostly due to the navigation performed after reaching the first goal and moving towards the second goal, which is when the moving obstacle appears. The increased frequency of the control loop plays an important role here, leading to faster updating of obstacle locations and an enhanced ability of the planner in *costmap* clearing routines. This is illustrated in Fig. 36: the path planning suggests longer paths in the first scenario due to the slower reaction times, whereas more efficient planning is possible in the second scenario.

Table 6 – Case study 2: distance covered by the UGV.

Scenario	Total Distance Covered (m)	Distance Covered from goal 1 to goal 2 (m)
Embedded Processing	43.4 ± 4.0	6.5 ± 3.5
Edge Cloud Processing	40.7 ± 0.3	3.8 ± 0.2
Commercial Cloud Processing	41.7 ± 1.0	4.2 ± 1.0
FUTEBOL Cloud Processing	41.8 ± 0.8	4.4 ± 1.0

Table 7 displays the mean CPU and memory use in each device during the experiment. The offloading of the computation tasks to the cloud reduces the UGV's CPU and memory usage in over 40 % and 35 %, respectively. Here, offloading also helps in preventing CPU overloads, which were fairly common during the experiment. Once again, the VMs are shown to be over-dimensioned and most of its resources remain idle. The difference in memory usage in the VMs can be explained by our implementation of the edge, as the laptop emulating the edge's VM also runs the Ubuntu Desktop's graphical interface, whereas the other VMs do not.

A total of 4544 *ping* measurements were performed during the second scenario (i.e., edge cloud processing), amounting to an average RTT of 28.7 ± 20.9 ms with a 0.5 % packet loss rate. The minimum and maximum measurements were of 1.3 ms and 1693.0 ms. In the third scenario, a total of 4817 *ping* measurements averaged an RTT of 152.2 ± 24.7 ms and a 0.1 % packet loss rate. The minimum and maximum measurements were of 116.2 ms and 1763.1 ms. Finally, during the fourth scenario, we performed a total of 4832 *ping* measurements towards FUTEBOL cloud, averaging 196.2 ± 24.1 ms and a 2.7 % packet

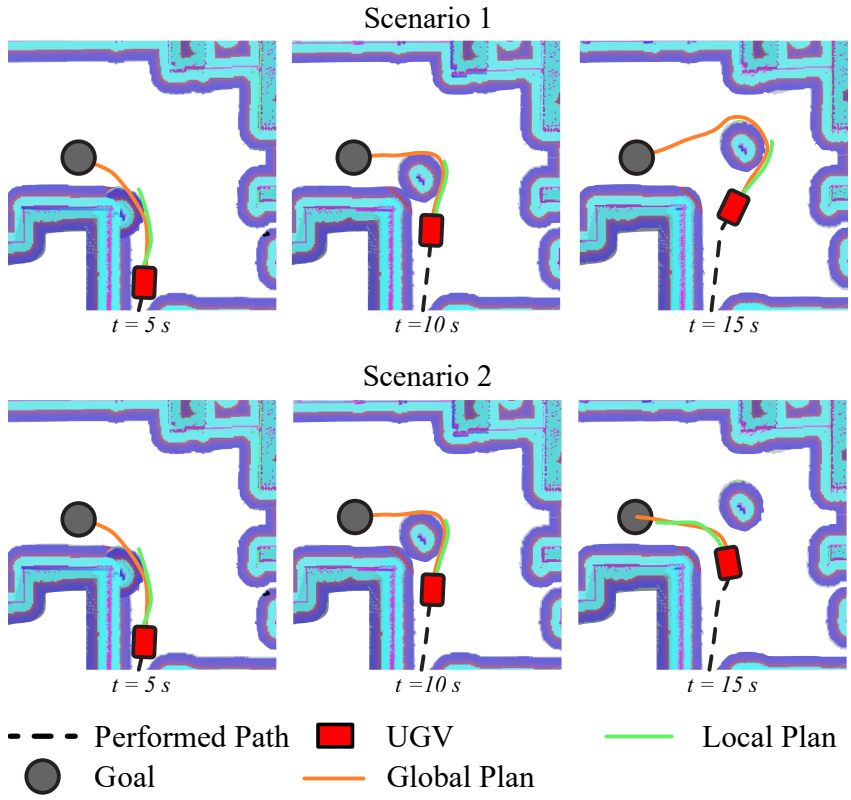


Figure 36 – Avoiding moving obstacles during navigation: path planning comparison in different scenarios.

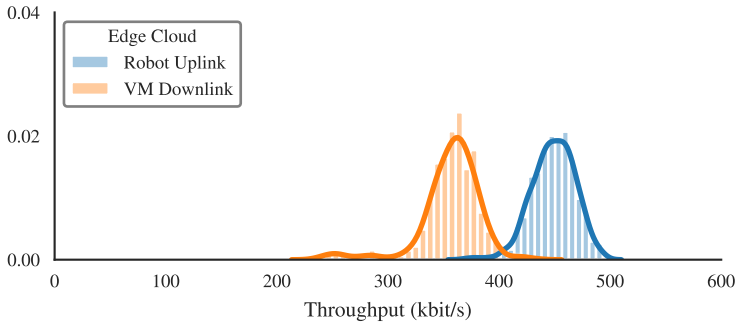
Table 7 – Case study 2: resource usage measured in each device per scenario.

Scenario	Device	Mean CPU Usage (%)	Mean Memory Usage (MB)
Embedded Processing	UGV	76.9 ± 21.5	866 ± 14
Edge Cloud Processing	UGV	43.1 ± 5.7	560 ± 14
	Edge	18.2 ± 4.2	$4,032 \pm 176$
Commercial Cloud Processing	UGV	47.1 ± 5.2	552 ± 14
	Cloud VM	26.7 ± 4.8	$1,264 \pm 112$
FUTEBOL Cloud Processing	UGV	46.4 ± 5.4	560 ± 14
	Cloud VM	22.9 ± 6.1	$1,152 \pm 153$

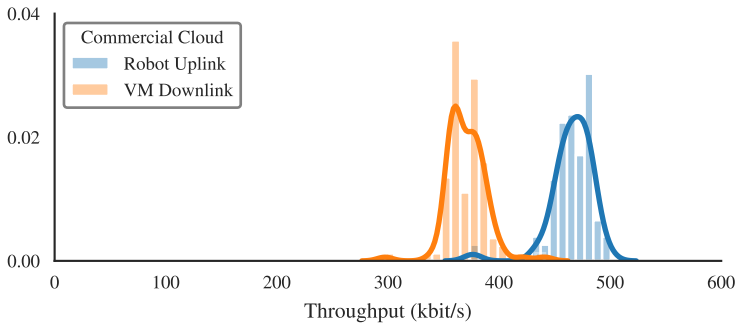
loss rate. The minimum and maximum measurements were of 177.4 ms and $8,428.7$ ms, an extreme value that would render robotic-related messages unsuitable.

Figure 37 displays the throughput distributions of robot uplink and cloud downlink measured during our experiments. The way that the distributions are shifted in the throughput axis indicates the amount of information loss in E2E communication. Again, we only plot the robot-to-cloud distribution as it surpasses the cloud-to-robot traffic by an order of magnitude. The average robot uplink and cloud downlink were similar in all three scenarios, fluctuating around 450 kb/s and 350 kb/s, respectively, amounting to a bit more than 20 % loss.

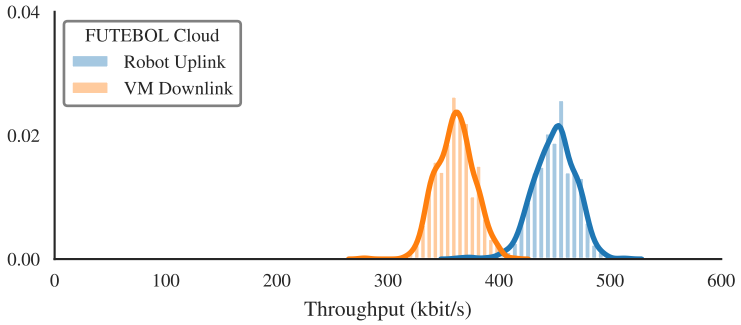
Figure 38 displays the message loss rate observed during the experiment. Once again, the loss rate of most messages stays below 2 % in all experiment scenarios. The */tf* loss



(a)



(b)



(c)

Figure 37 – Case study 2, throughput distribution: robot uplink and VM downlink observed during the (a) edge, (b) commercial, and (c) FUTEBOL cloud scenarios.

rate is significantly higher when compared to the other messages, at similar levels as observed in case study 1.

Fig. 39 displays the jitter distribution of the arriving messages, omitting the */tf*. Data from the LRF readings generate the largest payload and suffers more with jitter. Nevertheless, the median value observed for the */RosAria/laserscan* period is close to the ideal period in all scenarios. The period of smaller payload packets associated with the */cmd_vel* and */RosAria/pose* messages are less affected by the network and the observed distributions approximate the ideal value.

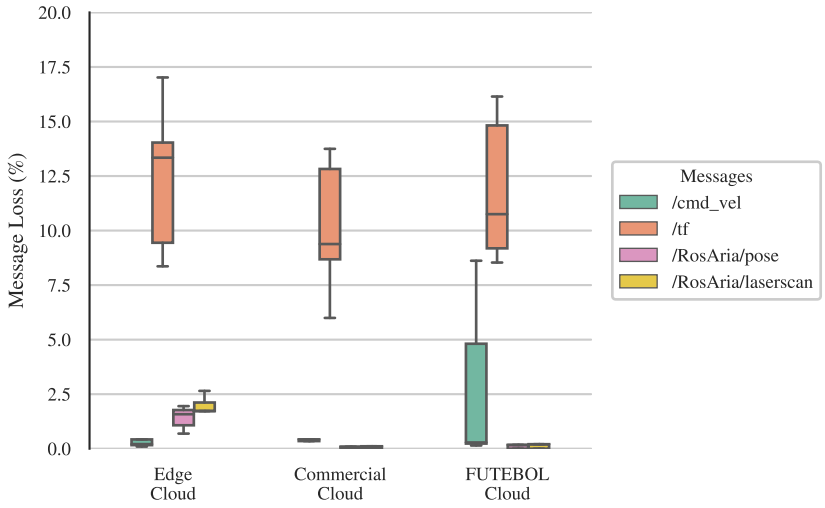


Figure 38 – Case study 2: distribution of message loss rate throughout the last three scenarios; messages are ordered, left-to-right, in ascending payload size.

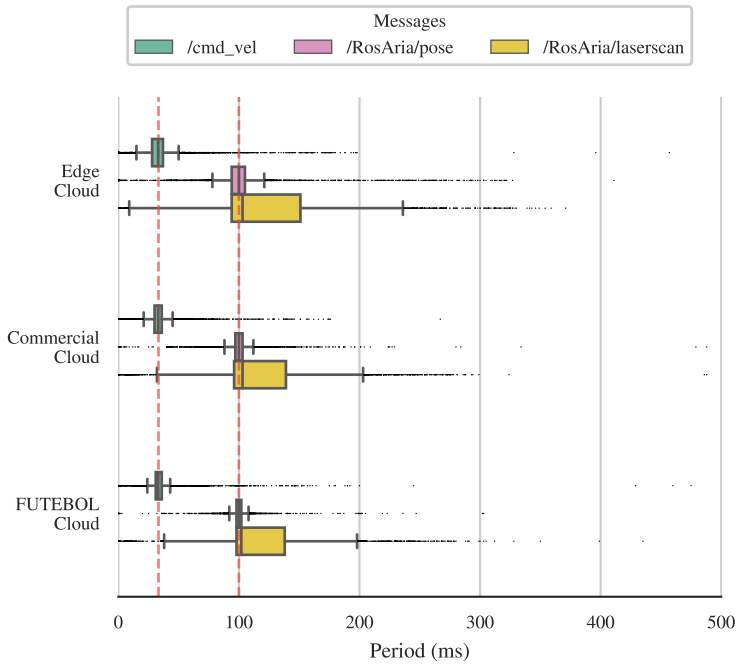


Figure 39 – Case study 2: jitter distribution of the arriving messages relevant to the robot's control. The red dashed lines indicate the ideal period values as indicated in Table 4.

5.7 Discussion

The results obtained in both use cases indicate the feasibility of employing cloud robotics into ROS-based systems in real-world scenarios. This favors our methodology and communication framework, indicating that it can be used as a basis for future cloud robotics implementations.

Besides the expected result that small latency favor better performances, our results also

demonstrate that the use of relatively high latency and complex networks can outperform local computation in simple tasks involving HRI. In previous works of our [61] and in Chapter 2, we indicate that latency is a major factor affecting the user's QoE during human-robot interaction. In general, the HRI is directly affected by latency whenever interaction-related processing blocks are remotely instantiated. This human-in-the-loop component plays a fundamental role during HRI and the acceptance of cloud-based systems is directly linked to the provided QoS [32]. Relating QoS and QoE is an intricate problem on its own [44] and Munir *et al.* [12] argues for the importance of understanding the effects of the human-in-the-loop in control as different and sophisticated applications appear.

In Chapter 2, we observe that the HRI may not be compromised when RTT is kept under 100 ms – or even under 300 ms. Despite the different tasks, our results corroborate those as our HRI experiments were accomplished with average RTT slightly above the 100 ms mark. Few works address the effects of network-related issues into the perceived HRI in distributed robotics systems, which hampers direct comparisons. The effects of RTT in the control loop vary with the specificity of the HRI task and, whereas we reported in Chapter 2 poor performance on the presence of 500 ms RTT, robotic teleoperation tasks have been investigated under RTT of 35–190 ms [121], 800 ms [138] and even $4,000$ ms [139], to cite a few examples.

Navigation tasks are impacted by the network differently. Whereas the HRI systems respond directly to the human inputs, autonomous navigation tasks usually rely on a planning stage taking place before action. Thus, considering network-related issues, the navigation planner might respond to old or incomplete data, hampering the quality of the plan. Moreover, as our experiments demonstrated, the slower reaction times linked to the distributed processing are important factors to be considered when dealing with dynamic and unstructured environments.

To contextualize the results observed during our navigation experiments with the expected capability of 5G networks, the 3GPP defines a target latency of 10 ms and packet loss probability smaller than 10^{-5} % for discrete automation in URLLC [17]. Nevertheless, the URLLC specifications aim at small data payloads (e.g., 32 bytes), which are one or two orders of magnitude smaller than the messages exchanged in our experiments [140]. The H2020 AUTOWARE consortium investigated robot mobility in industrial scenarios and observed RTT values up to 200 ms for $29 - -40$ B payload packets when connecting an industrial robot, a mobile robot, and a local controller using multi-path TCP connections over 2.4 GHz wireless networks [141]. Among the solutions for reliable robot mobility, the Mobile Access Point concept presented by Martinez *et al.* [19] achieved low RTT values over 2.4 GHz networks and UDP, posing itself as an alternative to multi-path TCP and other multi-connectivity schemes.

Cloud-based control of mobile robots was also investigated in the context of the FUTEBOL Project [14]. Carmo *et al.* [106] explored edge computing in an experiment

where a cloud service processes multiple video streams to localize a robot and control its displacement. The robot used in Carmo *et al.* [106] did not present any sensors nor performed processing tasks and merely executed the commands generated by the cloud service. Still in Carmo *et al.* [106], an optical fiber link connected the robot environment and the edge cloud (the robot itself connected to an access point via a 5 GHz wireless network), achieving RTT of 16 ms highly dominated by the latency in the wireless communication. Here, the experiments performed over the emulated edge cloud observed slightly higher RTT values, an expected result considering the pollution in the 2.4 GHz spectrum and the consumer-grade network devices linking the access point and the edge. Salmerón-Garcia *et al.* [110] compared the effects of different WiFi technologies in a set of cloud robotics experiments.

5.8 Conclusions

In this thesis, we have addressed and proposed solutions to some key cloud robotics requirements. This chapter extended our open framework for robot-cloud communication into a reproducible methodology for cloud robotics implementation. We discussed the implementation of the overall cloud robotics stack using open-source software and commercial off-the-shelf devices in such a way to enable other researchers to replicate our work. To showcase the performance of our system, we developed two cloud services, one focusing on HRI and the other on autonomous navigation, and performed a series of experiments over such use cases. Our experiments demonstrate the feasibility of the presented ROS-based cloud robotics system when dealing with non-ideal networks and different cloud platforms. Moreover, our demonstrative use cases exploited several aspects of ROS' systems that fit within a broad range of applications. Finally, our work was put into context and compared with other similar works while discussing challenges and opportunities in the cloud robotics field. In the next chapter, we conclude this thesis with an in-depth discussion of what we have achieved.

6 Conclusions and Future Work

In this concluding chapter, we summarize this thesis and the work hereby presented to discuss the implications of our research. We then offer our view regarding possible research directions that may arise from our work and how we believe that other techniques and technologies can be used to extend cloud robotics. Finally, we finish the thesis with our concluding remarks.

6.1 Thesis Summary

While carrying out this work, we have explored various aspects of the cloud robotics paradigm to further understand its implications and how it can be applied in different use cases. Given our group expertise with assistive robotics, our first steps working with cloud robotics were marked by several questions: "are there any gains in using the cloud to aid assistive devices?"; "is it viable to offload processing tasks to the cloud?"; "how would such a change of paradigm affect the user?"; "how network/cloud-related issues translate into the HRI?". The work presented in Chapter 2 explores these questions experimentally. Our pilot study indicated a relatively large tolerance to the insertion of latency in our human-in-the-loop system despite the extreme use case proposed and the close physical interaction present in smart walkers. Even though the user-perceived QoE degrades in worse QoS conditions, the same did not happen when evaluating other important features, such as the perception of safety. The limitations of our pilot study prevent us from generalizing our results to other populations and other assistive devices; at the same time, the results observed allow us to draw sufficient conclusions regarding the use of cloud-enabled assistive devices and their requirements.

The case study presented in Chapter 2 considered the remote placement of all of the device's controllers and, on top of that, purposefully considered worst-case RTT values. If we consider that, in the realistic use of cloud-aided assistive devices, RTT figures would be lower and some degree of local computation would be present, the effects of the network into the QoE would be minimized, thus rendering the cloud robotics paradigm seamless to the user. Considering eHealth, we envision a combination of physician-patient-therapist assessment not only for QoE but also including "acceptability" metrics [71]. In future works, one should also include alternative tools to measure long-term as well as short-term QoE measurements. Therefore, although there is room for extending our pilot study, it allowed us to answer the questions we first had in mind and also the questions that we stumbled with along the way.

Despite the effects of latency in cloud robotics systems, latency can be mitigated to a

degree by implementation choices, such as choosing a cloud provider in physical proximity or by deploying an on-site edge cloud. Nevertheless, other aspects of the network must be considered and we identified the uninterrupted wireless connectivity to be of paramount importance to mobile robots. In Chapter 3, we discussed the design of a cloud-based service for mobility assistance using as a basis the knowledge acquired during the realization of the pilot study presented in Chapter 2. We also presented a cloud robotics architecture that leverages the programmability of the network to link mobile robots and the cloud. Our solution for wireless communication is directed at providing reliable handovers to avoid connectivity interruptions. We validated the use of such a solution in smart walkers by implementing all of the layers of our proposed architecture, achieving positive results, and indicating that our technique can be used by mobile robots in the cloud robotics paradigm.

In Chapter 4, we presented the PoundCloud framework for robot-cloud communication. We pointed at the use of the ROS as a way of establishing a common set of practices in cloud robotics implementations and how the use of the PoundCloud can enable reproducible works. The PoundCloud fills a gap in the ROS ecosystem, which had no previous standard way of communicating robots with remote computing platforms. The functional validation of the PoundCloud also demonstrates its capacity and versatility, indicating that it can be readily used in a wide range of applications.

The PoundCloud is also a step towards the methodology for cloud robotics experimentation presented in Chapter 5. Our methodology allows for reproducible experiments in cloud robotics research and is powerful enough to contemplate a variety of applications, as portraited by the case studies presented in Chapter 5. By performing practical experiments over the public Internet with consumer-grade equipment, we validated the PoundCloud. As one of the case studies presented relyed on HRI, we also further validated the findings presented in Chapter 2 regarding the latency-tolerant aspect of human-in-the-loop systems. Although we used an already existing university network in our implementation to ease the replication of our setup, the network solution for wireless communication presented in Chapter 3 could be integrated to extend our methodology. Finally, our experiments demonstrated that our methodology can be used as a basis for other researchers in the field.

Overall, our work contributed to the field of cloud robotics by addressing often-neglected aspects. Not only we presented insight on the use of HRI and wireless communication in cloud robotics, but also we presented an open methodology that can be replicated and independently validated to foster practical implementations of cloud robotics. The next section expands our vision on how our work can be extended and what research lines may arise from it.

6.2 Future Work

Despite our efforts to limit the scope of this work, our research activities have inevitably touched on a variety of other topics. For instance, we have explored only single-cloud architectures in our work; in other words, we have either used a local edge cloud or a remote cloud. Mixed architectures can be used to leverage the advantages of each type of cloud, as we have previously discussed in Chapter 4 in the text regarding Fig. 28. Going beyond the realm of the cloud, cloud robotics can be merged with the so-called fog robotics to explore distributed processing setups to mitigate latency. Moreover, edge devices – not to be confounded with edge clouds – can be responsible for data pre-processing and even execute parts of a service. In this topic, the architecture presented in Chapter 3 can be expanded to include the fSTAs (i.e., the infrastructure devices that provide wireless connectivity) as edge computing nodes. To illustrate how this can be done, let us take as an example the follow-in-front service presented in Chapter 5.5.1. Part of this service relies on a simple machine learning algorithm to detect the user's legs position by clustering LRF readings. Another part of the service, the controller, uses the information generated by the leg and face detection systems to generate control signals to command the robot. These parts of the service (i.e., the leg detection system and the controller) require low processing power and thus could be executed in our fSTAs, which can be implemented using mini-PCs or even Raspberry Pis. Then, the cloud would only be in charge of the image processing part of the service – the most resource-consuming part. This idea can be escalated if one considers a fleet of mobile robots that rely on a set of services of which parts can be processed in the edge. Thus, our research can be extended to incorporate hybrid distributed computing architectures as a way to enhance the cloud robotics paradigm.

Breaking a robotics service into micro-services that can be distributed through the network resembles, in some way, the idea of Service Function Chainning (SFC). SFC is a networking mechanism that interconnects – or chains – groups of service functions to compose a complex service [142]. This mechanism is often enabled by SDN, which deals with the forwarding of packet flows among Virtual Network Functions (VNFs). In the networking context, VNFs are functions such as firewalls, proxy servers, and load balancers. Future works in cloud robotics can bring SFC and VNFs concepts into robotics to stitch together distributed fragments of a service. This can be explored to enable the hybrid architecture scenario discussed in the previous paragraph, linking edge nodes, edge clouds, and remote clouds.

As we thin the line separating robotics and networking, in-network computing [143,144] concepts can be incorporated to include the execution of robotic functions (e.g., pre-processing sensor data and fail-safe mechanisms) into networking devices such as intelligent switches and – as previously hinted – wireless access points. To this end, the P4 language allows for programming compatible networking devices to express how packets should be processed and forwarded. As shown by our colleagues from the NERDS group in

Mafioletti *et al.* [145], it is possible to use P4 to execute certain VNFs directly into P4-enabled hardware – a technique the authors called embedded Network Function (eNF). This removes the overheads of traditional VNF implementations and has the potential to drastically reduce VNF-related latency. We are starting to explore in-network computing in the context of robotics to leverage the eNF concept to execute latency-sensitive tasks. Right now, we envision the use of embedded robotics functions to parse flows of robot-cloud communication for on-line monitoring and event detection. To provide a few examples of possible applications, this can be used to extend telemetry data related to fleets of robots and also to trigger alarms in case of communication instability, sensor failure, or when unsafe situations are detected. Unfortunately, when it comes to introducing robotic control functionality inside the network, there are fundamental challenges such as the absence of floating-point calculation in P4-enabled devices. This hampers the possibility of implementing controllers into eNFs in the foreseeable future and demands the use of virtualized functions on top of hypervisors – which is similar to our proposal of using edge devices such as fSTAs implemented using mini-PCs.

To further link our work with in-network computing, the PoundCloud can be extended to become a network function – and thus transparent to the robot. Instead of instantiating PoundCloud nodes in each robot, as we did in Chapters 4 and 5, a service-oriented version of the PoundCloud could be placed in P4-enabled edge devices to bridge robots and clouds while also allowing for the chaining of embedded functions. One possible use-case would be to encrypt robot-cloud traffic to allow for secure communication even in applications where encryption is not a local requirement. In other words, the robots could use the standard ROS and ROS 2 communication middlewares, which do not encrypt the transmitted data, while the PoundCloud would be responsible for edge-to-cloud and cloud-to-edge encryption. In such a case, as the PoundCloud network function would be generating the security keys, such keys could be shared with other eNFs to allow for actual data inspection and manipulation. This would not be trivial to manage in case each robot encrypted its data and would be a special challenge when multiple robots are present. This line of research can be investigated to assess if there are actual performance gains in such an approach. Since in-networking computing for robotics is a newborn line of research, maybe the biggest challenge right now is to foresee what efforts are worth pursuing and how to avoid the one-hammer pitfall[1]. In case in-network computing becomes widely adopted, pioneer work aiming at robotics may have a deep impact in future networked robotics systems.

As we argued in Chapter 2, there are reasons for cloud robotics service providers to pay attention to their operational QoS levels. Application developers should detail QoS requirements for each application and build ways to monitor if such requirements are being met – a necessity that we pointed in Chapter 3. One possible way to move

[1] By calling it the one-hammer pitfall, we take the liberty of naming a concept that was neatly phrased by Abraham Maslow in his 1966 book, The Psychology of Science, p. 15: *"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."*

forward is to model the network to understand its capacity and responsiveness; this may be possible by employing the network calculus paradigm [146]. With a mathematical network model in hands, one can establish latency bounds considering different network scenarios (e.g., possible routes, concurrent traffic, devices' capacity, etc.), which can feed offloading mechanisms responsible for deciding when to delegate processing tasks to the cloud. This can also be coupled with robotics-oriented eNFs to monitor parts of the network to update the model and expected boundaries. Fine-grained QoS measurements could then be used to refine the pilot study presented in Chapter 2 and better estimate the QoS x QoE relationship.

Besides looking at the network, future work should also address the application-side of cloud robotics. As for the CloudWalker project, not only have we adopted the PoundCloud, but we are also currently exploring the use of behavior trees to manage HREI. Behavior trees is an artificial intelligence technique capable of reacting to changes in the robot's surroundings and making decisions regarding the proper actions – or behaviors – to perform. We suspect that the use of behavior trees may also help with offloading decision-making and with decoupling local and remote processing. Whereas the former may leverage the inherent decision-making mechanism of behavior trees, the latter can be achieved if we consider the set of behaviors that can be processed locally to be a subset of the total available behaviors enabled by the cloud. Thus, in case the QoS requirements are not met, the local subtree takes control of the device to execute critical tasks; this may assist application developers when designing cloud robotics systems.

The CloudWalker was used in this work mainly as an experimentation platform and as an application for cloud robotics. While our first steps with the CloudWalker involved relatively simple controllers and setups – when considering the big picture of cloud robotics – our research on the field is moving towards exploring more complex HRI and HREI. For instance, the HRI strategy presented Scheidegger *et al.* [115] and used in Chapter 5.5 was first designed with the CloudWalker in mind and future work will address its implementation. Moreover, the current state of the CloudWalker project also envisions the use of a depth camera and deep learning algorithms to extract semantic information from the environment. Navigation systems such as the one discussed in Chapter 3 and implemented in Chapter 5.6 will also be used in the CloudWalker. This further increases the complexity of the overall system and opens gates for several cloud-based implementations with high throughput and reliability requirements.

Furthermore, the present year was marked by a pandemic that drastically changed the dynamics of healthcare. The highly contagious SARS-CoV-2 virus became a serious hazard in hospital environments and robots have been employed to avoid human exposure. Telepresence robots are assisting hospital staff in patient triage and consultation, whereas disinfection robots are used to sanitize environments. These new necessities also present opportunities to extend our work towards the cloud-based management of heterogeneous

fleets of robots in clinical environments.

6.3 Final Remarks

Computing trends come in waves that follow technological advancements; we have seen it before as the many-terminals-single-mainframe paradigm was shadowed by the rise of personal computers, whereas through the last twenty years the cloud dominated the market and rendered personal computers and smartphones to become deeply dependent on connectivity with the Internet. Since the cloud robotics term was first used in 2010 by James Kuffner, and even also after the beginning of this work, the consumer hardware market changed radically. The introduction of the low-cost Raspberry Pi computer favored hobbyist roboticists, researchers, and even the industry, allowing for the cheap deployment of complex robotic applications. The posterior introduction of the Nvidia Jetson and its embedded GPU brought an enormous computing power in a compact form factor, enabling applications based on deep learning without the need for external computing nodes. Now, even microcontrollers costing just a few dollars are capable of processing some machine learning algorithms and act as edge computing nodes. And yet, despite the advancements in embedded hardware resources, cloud-enabled robotics solutions are attractive for a large range of applications.

Internet connectivity is a fundamental requirement in most recent robots. Even if we look at robots that should not be connected to the Internet (e.g., for security reasons), some degree of networking is desirable to allow for easy configuration, logging, and diagnostic. To give a simple example, remote monitoring is a feature that is ubiquitous not only in industry and service robotics, but also on the last generation of consumer items such as security cameras and even doorbells. When it comes to managing fleets of robots, it is fundamental to centralize information into a platform to allow for monitoring and data storing. Moreover, connectivity is necessary for automatically updating firmware and correcting bugs. The cloud is a strong candidate solution in such examples and cloud-native apps are already common-place as AWS' Cloud Robotics service and the rise of companies such as Freedom Robotics[2] indicate.

When it comes to HRI, the cloud is likely to remain the best solution for several use cases. As illustrated by Softbank's Pepper and Anki's Vector, cloud access allows for powerful speech recognition and natural language processing. Cloud-aided robots such as these can leverage chatbot features to interact and provide information. Analogous to the cloud-native apps in our smartphones, there are few justifications for embedding this kind of feature in robots regardless of embedded processing power advancements. Even when machine learning models can be minimized to be used with resource-constrained

[2] Website: <https://www.freedomrobotics.ai/>. Accessed on December 1, 2020.

hardware, as in the case of the Tinier-YOLO [147][3], this comes with a loss of accuracy and degraded performance. In this work, we have looked into the uninterrupted connectivity problem, which paired with the throughput capacity of current WiFi standards allows for connecting robots and cloud platforms seamlessly, as we showed in Chapter 5.

Although in this work we have used the cloud as a part of the control loop, there are other ways to leverage the cloud while minimizing network-related constraints interference in control. For instance, the cloud can be used to process sensor data and feed the robot with updated parameters or semantic information, removing itself from the immediate control loop. Even if we can achieve the low latency and high-reliability figures envisioned by 5G, low-level controllers are safety-critical and should remain embedded – even if only simplified versions of the controllers can be executed when connectivity fails. Such hybrid architectures can benefit from decision-making mechanisms based on network modeling and monitoring to decide when to offload controllers, as discussed in the previous section.

Finally, even after all these years in which we have conducted this work, cloud robotics is still an exciting field of research. It is clear to us that cloud robotics is already a viable paradigm to enable and empower a large range of robots and robotics applications, which was not the case when we started with this line of research. Furthermore, the current technological advancements may render cloud robotics a ubiquitous paradigm in the next few years. The market for commercial cloud robotics platforms continues to increase and the same is likely to happen with service providers in the field. Nevertheless, although established cloud providers are well prepared to manage the cloud side of things, application developers and network engineers still face several challenges, including providing an adequate QoS and guaranteeing the integrity of the field operation. As we indicate in the previous section, cloud robotics overlaps with many other topics and should remain a relevant research area in the foreseeable future. We conclude by reaffirming our belief that the work present in this thesis provides solid guidelines and interesting insight to foster cloud robotics research and experimentation.

[3] The Tinier-YOLO is a minimalist version of the YOLO [148], a state-of-the-art real-time object detection system.

Bibliography

[1] WAN, J. et al. Cloud robotics: Current status and open issues. *IEEE Access*, v. 4, p. 2797–2807, 2016. ISSN 2169-3536.

[2] SAHA, O.; DASGUPTA, P. A Comprehensive Survey of Recent Trends in Cloud Robotics Architectures and Applications. *Robotics*, v. 7, n. 3, p. 47, set. 2018.

[3] MOHANARAJAH, G. et al. Rapyuta: A Cloud Robotics Platform. *IEEE Transactions on Automation Science and Engineering*, v. 12, n. 2, p. 481–493, abr. 2015. ISSN 1545-5955.

[4] ZHENG, S. et al. IAPcloud: A Cloud Control Platform for Heterogeneous Robots. *IEEE Access*, v. 6, p. 30577–30591, 2018. ISSN 2169-3536.

[5] CHEN, W. et al. A Study of Robotic Cooperation in Cloud Robotics: Architecture and Challenges. *IEEE Access*, v. 6, p. 36662–36682, 2018. ISSN 2169-3536.

[6] CHEN, W. et al. QoS-aware Robotic Streaming Workflow Allocation in Cloud Robotics Systems. *IEEE Transactions on Services Computing*, p. 1–1, 2018. ISSN 1939-1374.

[7] SHAH, T. et al. Remote health care cyber-physical system: Quality of service (QoS) challenges and opportunities. *IET Cyber-Physical Systems: Theory & Applications*, v. 1, n. 1, p. 40–48, 2016. ISSN 2398-3396.

[8] CIFUENTES, C. A. et al. Multimodal Human-Robot Interaction for Walker-Assisted Gait. *IEEE Systems Journal*, v. 10, n. 3, p. 933–943, 2016. ISSN 1932-8184.

[9] WALSH, C. Human-in-the-loop development of soft wearable robots. *Nature Reviews Materials*, p. 78–80, 2018.

[10] NUNES, D. S. S.; ZHANG, P.; SILVA, J. S. A Survey on human-in-The-loop applications towards an internet of all. *IEEE Communications Surveys and Tutorials*, v. 17, n. 2, p. 944–965, 2015. ISSN 1553877X.

[11] SENFT, E. et al. Teaching robots social autonomy from in situ human guidance. *Science Robotics*, Science Robotics, v. 4, n. 35, 2019.

[12] MUNIR, S. et al. Cyber-Physical System Challenges for Human-in-the-Loop Control. In: *Cyber Physical System Challenges for Human-in-the-Loop Control*. San Jose, CA: USENIX, 2013. p. 4.

[13] HAMMER, F.; EGGER-LAMPL, S.; MOLLER, S. Position Paper: Quality-of-Experience of Cyber-Physical System Applications. In: *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. [S.l.: s.n.], 2017. p. 1–3. ISBN 978-1-5386-4024-1.

[14] RIBEIRO, M. R. N. 5G Research and Testbeds in Brazil. In: *2019 Optical Fiber Communications Conference and Exhibition (OFC)*. [S.l.: s.n.], 2019. p. 1–3.

[15] CHINCHALI, S. et al. Network Offloading Policies for Cloud Robotics: A Learning-based Approach. *arXiv:1902.05703 [cs]*, fev. 2019.

[16] Expert Working Group (Joint; PAPER), W. *5G: Challenges, Research Priorities, and Recommendations*. [S.l.], 2014. 45 p.

[17] 3GPP. *Service Requirements for next Generation New Services and Markets. Rev. 16.3.0*. [S.l.], 2018.

[18] WAN, J. et al. Context-Aware Cloud Robotics for Material Handling in Cognitive Industrial Internet of Things. *IEEE Internet of Things Journal*, v. 5, n. 4, p. 2272–2281, ago. 2018. ISSN 2327-4662.

[19] MARTINEZ, V. M. G. et al. Ultra Reliable Communication for Robot Mobility enabled by SDN Splitting of WiFi Functions. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. Natal: IEEE, 2018. p. 00527–00530. ISBN 978-1-5386-6950-1.

[20] YOSHIKANE, N. et al. First Demonstration of Geographically Unconstrained Control of an Industrial Robot by Jointly Employing SDN-based Optical Transport Networks and Edge Compute. p. 3.

[21] NAMAN, A. T. et al. Responsive high throughput congestion control for interactive applications over SDN-enabled networks. *Computer Networks*, v. 134, p. 152 – 166, 2018. ISSN 1389-1286.

[22] LIBERATO, A. et al. RDNA: Residue-defined networking architecture enabling ultra-reliable low-latency datacenters. *IEEE Transactions on Network and Service Management*, v. 15, n. 4, p. 1473–1487, Dec 2018. ISSN 2373-7379.

[23] HU, G.; TAY, W.; WEN, Y. Cloud robotics: Architecture, challenges and applications. *IEEE Network*, v. 26, n. 3, p. 21–28, maio 2012. ISSN 0890-8044.

[24] MOHANARAJAH, G. et al. Cloud-Based Collaborative 3D Mapping in Real-Time With Low-Cost Robots. *IEEE Transactions on Automation Science and Engineering*, v. 12, n. 2, p. 423–431, abr. 2015. ISSN 1545-5955.

[25] SUGIURA, K.; ZETTSU, K. Rospeex: A cloud robotics platform for human-robot spoken dialogues. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2015. p. 6155–6160.

[26] LU, D. et al. VC-bots: A vehicular cloud computing testbed with mobile robots. In: *Proceedings of the First International Workshop on Internet of Vehicles and Vehicles of Internet - IoV-VoI '16*. Paderborn, Germany: ACM Press, 2016. p. 31–36. ISBN 978-1-4503-4345-9.

[27] MAHLER, J. et al. Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2016. p. 1957–1964.

[28] ALI, S. S.; HAMMAD, A.; ELDIEN, A. S. T. FastSLAM 2.0 tracking and mapping as a Cloud Robotics service. *Computers & Electrical Engineering*, v. 69, p. 412–421, jul. 2018. ISSN 00457906.

[29] QUIGLEY, M. et al. ROS: an open-source robot operating system. *ICRA Workshop on Open Source Software*, v. 3, 01 2009.

[30] KOUBAA, A.; ALAJLAN, M.; QURESHI, B. ROSLink: Bridging ROS with the Internet-of-Things for Cloud Robotics. In: KOUBAA, A. (Ed.). *Robot Operating System (ROS)*. Cham: Springer International Publishing, 2017. v. 707, p. 265–283. ISBN 978-3-319-54926-2 978-3-319-54927-9.

[31] JIMÉNEZ, M. F. et al. Admittance Controller with Spatial Modulation for Assisted Locomotion using a Smart Walker. *Journal of Intelligent & Robotic Systems*, v. 94, n. 3-4, p. 621–637, jun. 2019. ISSN 0921-0296, 1573-0409.

[32] MELLO, R. C. et al. On Human-in-the-Loop CPS in Healthcare: A Cloud-Enabled Mobility Assistance Service. *Robotica*, p. 1–17, fev. 2019. ISSN 0263-5747, 1469-8668.

[33] BONSIGNORIO, F. A new kind of article for reproducible research in intelligent robotics [from the field]. *IEEE Robotics Automation Magazine*, v. 24, n. 3, p. 178–182, Sep. 2017. ISSN 1558-223X.

[34] GUGLIELMELLI, E. Research reproducibility and performance evaluation for dependable robots [from the editor's desk]. *IEEE Robotics Automation Magazine*, v. 22, n. 3, p. 4–4, Sep. 2015. ISSN 1558-223X.

[35] BASTOS-FILHO, T. F. et al. Towards a new modality-independent interface for a robotic wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 22, n. 3, p. 567–584, maio 2014. ISSN 1534-4320.

[36] LIU, Y. et al. Review on cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica*, v. 4, n. 1, p. 27–40, jan. 2017. ISSN 2329-9266.

[37] BECKERLE, P. et al. A Human–Robot Interaction Perspective on Assistive and Rehabilitation Robotics. *Frontiers in Neurorobotics*, v. 11, p. 24, maio 2017. ISSN 1662-5218.

[38] HAQUE, S. A.; AZIZ, S. M.; RAHMAN, M. Review of cyber-physical system in healthcare. *International Journal of Distributed Sensor Networks*, Hindawi Publishing Corporation, v. 2014, 2014. ISSN 15501477.

[39] REPPOU, S. E. et al. RAPP: A Robotic-Oriented Ecosystem for Delivering Smart User Empowering Applications for Older People. *International Journal of Social Robotics*, Springer Netherlands, v. 8, n. 4, p. 539–552, 2016. ISSN 18754805.

[40] KEHOE, B. et al. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering*, v. 12, n. 2, p. 398–409, abr. 2015. ISSN 1545-5955.

[41] TSARDOULIAS, E. G. et al. Towards an integrated robotics architecture for social inclusion – The RAPP paradigm. *Cognitive Systems Research*, v. 43, p. 157 – 173, 2017. ISSN 1389-0417.

[42] CARDARELLI, E. et al. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics*, v. 45, p. 1–13, ago. 2017. ISSN 09574158.

[43] ROY, A.; ROY, C.; MISRA, S. CARE : Criticality-Aware Data Transmission in CPS-based Healthcare Systems. *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, p. 1–6, 2018.

[44] VARELA, M.; Skorin-Kapov, L.; EBRAHIMI, T. Quality of Service Versus Quality of Experience. In: MÖLLER, S.; RAAKE, A. (Ed.). *Quality of Experience*. Cham: Springer International Publishing, 2014. p. 85–96. ISBN 978-3-319-02680-0 978-3-319-02681-7.

[45] PONS, J. L. *Wearable Robots: Biomechatronic Exoskeletons*. Chichester, UK: John Wiley & Sons, Ltd, 2008. ISBN 978-0-470-98766-7.

[46] BORDEL, B. et al. Cyber–physical systems: Extending pervasive sensing from control theory to the Internet of Things. *Pervasive and Mobile Computing*, v. 40, p. 156–184, 2017. ISSN 15741192.

[47] LEE, H. et al. Human-robot cooperative control based on pHRI (Physical human-robot interaction) of exoskeleton robot for a human upper extremity. *International Journal of Precision Engineering and Manufacturing*, v. 13, n. 6, p. 985–992, 2012. ISSN 12298557.

[48] FLEMISCH, F. O. et al. Towards cooperative guidance and control of highly automated vehicles: H-Mode and Conduct-by-Wire. *Ergonomics*, v. 57, n. 3, p. 343–360, mar. 2014. ISSN 1366-5847.

[49] HOSSAIN, M. S. Cloud-supported cyber-physical localization framework for patients monitoring. *IEEE Systems Journal*, v. 11, n. 1, p. 118–127, mar. 2017. ISSN 19379234.

[50] ZHANG, Y. et al. Health-CPS : Healthcare Cyber-Physical System Assisted by Cloud and Big Data. *IEEE Systems Journal*, v. 11, n. 1, p. 88–95, mar. 2017. ISSN 1932-8184.

[51] CHEN, M. et al. Smart Clothing: Connecting Human with Clouds and Big Data for Sustainable Health Monitoring. *Mobile Networks and Applications*, Mobile Networks and Applications, v. 21, n. 5, p. 825–845, 2016. ISSN 15728153.

[52] DOGMUS, Z.; ERDEM, E.; PATOGLU, V. RehabRobo-Onto: Design, development and maintenance of a rehabilitation robotics ontology on the cloud. *Robotics and Computer-Integrated Manufacturing*, v. 33, p. 100–109, jun. 2015. ISSN 07365845.

[53] TSUJI, T.; KANEKO, T.; SAKAINO, S. Motion Matching in Rehabilitation Databases With Force and Position Information. *IEEE Transactions on Industrial Electronics*, v. 63, n. 3, p. 1935–1942, mar. 2016. ISSN 0278-0046.

[54] FOSCH-VILLARONGA, E. et al. Cloud services for robotic nurses? Assessing legal and ethical issues in the use of cloud services for healthcare robots. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2018. p. 290–296.

[55] FIORINI, L. et al. Enabling personalised medical support for chronic disease management through a hybrid robot-cloud approach. *Autonomous Robots*, Springer US, v. 41, n. 5, p. 1263–1276, jun. 2017. ISSN 15737527.

[56] RADU, C.; CANDEA, C.; CANDEA, G. Towards an Assistive System for Human-Exoskeleton Interaction. In: *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '16*. [S.l.: s.n.], 2016. p. 1–4. ISBN 978-1-4503-4337-4.

[57] LI, H.-J. J.; SONG, A.-G. G. Architectural Design of a Cloud Robotic System for Upper-Limb Rehabilitation with Multimodal Interaction. *Journal of Computer Science and Technology*, v. 32, n. 2, p. 258–268, mar. 2017. ISSN 10009000.

[58] FU, J. et al. A novel mobile-cloud system for capturing and analyzing wheelchair maneuvering data: A pilot study. *Assistive Technology*, v. 28, n. 2, p. 105–114, 2016. ISSN 19493614.

[59] SALHI, K. et al. Navigation assistance to disabled persons with powered wheelchairs using tracking system and cloud computing technologies. *Proceedings - International Conference on Research Challenges in Information Science*, v. 2016-Augus, 2016. ISSN 21511357.

[60] MELLO, R. et al. Towards a New Generation of Smart Devices for Mobility Assistance: CloudWalker, a Cloud-Enabled Cyber-Physical System. In: *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*. Enschede: IEEE, 2018. p. 439–444. ISBN 978-1-5386-8183-1.

[61] MELLO, R. *UFES CloudWalker: A Cloud-Enabled Cyber-Physical System for Mobility Assistance*. Tese (Master's Thesis) — Universidade Federal do Espírito Santo, Vitória, Brazil, mar. 2018.

[62] WACHAJA, A. et al. Navigating blind people with walking impairments using a smart walker. *Autonomous Robots*, v. 41, n. 3, p. 555–573, mar. 2017. ISSN 0929-5593, 1573-7527.

[63] PANTELERIS, P.; ARGYROS, A. A. Vision-based SLAM and moving objects tracking for the perceptual support of a smart walker platform. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8927, p. 407–423, 2015. ISSN 16113349.

[64] European Parliament and Council of European Union. *Regulation (EU) 2016/679*. 2016. `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN`. Online; accessed on December 1, 2020.

[65] United States' Food and Drug Administration. *Policy for Device Software Functions andMobile Medical Applications*. 2019. `https://www.regulations.gov/document?D=FDA-2011-D-0530-0112`. Online; accessed on December 1, 2020.

[66] LI, Y. et al. Toward QoS-Aware Cloud Robotic Applications: A Hybrid Architecture and Its Implementation. In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. [S.l.: s.n.], 2016. p. 33–40.

[67] POCOVI, G. et al. Achieving ultra-reliable low-latency communications: Challenges and envisioned system enhancements. *IEEE Network*, v. 32, n. 2, p. 8–15, mar. 2018. ISSN 0890-8044.

[68] DOMINICINI, C. K. et al. VirtPhy: Fully Programmable NFV Orchestration Architecture for Edge Data Centers over a Fully Programmable SDN Infrastructure for

Small Data Centers. *IEEE Transactions on Network and Service Management*, v. 14, n. 4, p. 817–830, 2017. ISSN 19324537.

[69] MARTINELLO, M. et al. Keyflow: A prototype for evolving SDN toward core network fabrics. *IEEE Network*, v. 28, n. 2, p. 12–19, 2014. ISSN 0890-8044.

[70] PANTELERIS, P.; ARGYROS, A. A. Vision-based SLAM and moving objects tracking for the perceptual support of a smart walker platform. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2015. v. 8927, p. 407–423. ISBN 978-3-319-16198-3. ISSN 16113349.

[71] HOSSFELD, T. et al. QoE beyond the MOS: An in-depth look at QoE via better metrics and their relation to MOS. *Quality and User Experience*, v. 1, n. 1, p. 1–23, 2016. ISSN 2366-0139.

[72] STREIJL, R. C.; WINKLER, S.; HANDS, D. S. Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives. *Multimedia Systems*, Springer Berlin Heidelberg, v. 22, n. 2, p. 213–227, mar. 2016. ISSN 09424962.

[73] TALMAN, L. S.; al., e. Longitudinal study of vision and retinal nerve fiber layer thickness in multiple sclerosis. *Annals of Neurology*, v. 67, n. 6, p. 749–760, 2010. ISSN 03645134.

[74] ALEXANDER, N. B.; GOLDBERG, A. Gait disorders: Search for multiple causes. *Cleveland Clinic Journal of Medicine*, v. 72, n. 7, 2005. ISSN 08911150.

[75] ZHANG, X.; HAN, Q.; YU, X. Survey on Recent Advances in Networked Control Systems. *IEEE Transactions on Industrial Informatics*, v. 12, n. 5, p. 1740–1752, 2016. ISSN 1551-3203.

[76] DORF, R. C. *The Engineering Handbook*. Second. New York: CRC PRESS, 2004. ISBN 978-0-8493-1586-2.

[77] DEVLIN, A. Wayfinding in Healthcare Facilities: Contributions from Environmental Psychology. *Behavioral Sciences*, v. 4, n. 4, p. 423–436, 2014. ISSN 2076-328X.

[78] GERAVAND, M. et al. An Integrated Decision Making Approach for Adaptive Shared Control of Mobility Assistance Robots. *International Journal of Social Robotics*, Springer Netherlands, v. 8, n. 5, p. 631–648, 2016. ISSN 18754805.

[79] ANDALUZ, V. H. et al. Adaptive Dynamic Path Following Control of an Unicycle-Like Mobile Robot. In: JESCHKE, S.; LIU, H.; SCHILBERG, D. (Ed.). *Intelligent Robotics and Applications: 4th International Conference, ICIRA 2011, Aachen, Germany,*

December 6-8, 2011, Proceedings, Part I. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 563–574. ISBN 978-3-642-25486-4.

[80] LAZAR, J.; H., F. J.; HOCHHEISER, H. *Research Methods in Human-Computer Interaction*. 2. ed. [S.l.]: Morgan Kaufmann, 2017. ISBN 978-0-12-805390-4.

[81] FIEDLER, M.; HOSSFELD, T.; Tran-Gia, P. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, v. 24, n. 2, p. 36–41, mar. 2010. ISSN 08908044.

[82] TATEMATSU, A. et al. QoE assessment in tele-operation with 3D video and haptic media. In: *2011 IEEE International Conference on Multimedia and Expo*. [S.l.: s.n.], 2011. p. 1–6. ISSN 1945-788X.

[83] XU, X.; LIU, Q.; STEINBACH, E. Toward QoE-driven dynamic control scheme switching for time-delayed teleoperation systems: A dedicated case Study. In: *2017 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*. [S.l.: s.n.], 2017. p. 1–6.

[84] MCDONALD, J. H. *Handbook of Biological Statistics*. 3. ed. [S.l.]: Sparky House Publishing, 2014.

[85] WERNER, C. et al. A systematic review of study results reported for the evaluation of robotic rollators from the perspective of users. *Disability and Rehabilitation: Assistive Technology*, Informa UK Ltd., v. 0, n. 0, p. 1–12, 2017. ISSN 1748-3107.

[86] WERNER, C. et al. User-Oriented Evaluation of a Robotic Rollator That Provides Navigation Assistance in Frail Older Adults with and without Cognitive Impairment. *Gerontology*, p. 278–290, 2017. ISSN 14230003.

[87] PACE, P. et al. An Edge-based Architecture to Support Efficient Applications for Healthcare Industry 4.0. *IEEE Transactions on Industrial Informatics*, p. 1–1, 2018. ISSN 1551-3203.

[88] BOGUE, R. Cloud robotics: A review of technologies, developments and applications. *Industrial Robot: An International Journal*, v. 44, n. 1, p. 1–5, 2017. ISSN 0143-991X.

[89] RAAF, B. et al. Key Technology Advancements Driving Mobile Communications From Generation to Generation. *Intel Technology Journal*, v. 18, n. 1, 2014.

[90] KARJEE, J. et al. Distributed Cooperative Communication and Link Prediction in Cloud Robotics. In: *2017 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*. [S.l.: s.n.], 2017. p. 1–7.

[91] WU, Z.; MENG, Z.; GRAY, J. IoT-Based Techniques for Online M2M-Interactive Itemized Data Registration and Offline Information Traceability in a Digital Manufacturing System. *IEEE Transactions on Industrial Informatics*, v. 13, n. 5, p. 2397–2405, out. 2017. ISSN 1551-3203.

[92] CONTRERAS, J.; ZEADALLY, S.; Guerrero-Ibanez, J. A. Internet of Vehicles: Architecture, Protocols, and Security. *IEEE Internet of Things Journal*, v. 4662, n. c, p. 1–1, 2017. ISSN 2327-4662.

[93] YAN, H. et al. Cloud robotics in Smart Manufacturing Environments: Challenges and countermeasures. *Computers & Electrical Engineering*, v. 63, p. 56–65, out. 2017. ISSN 0045-7906.

[94] WANG, C. et al. SDCoR: Software Defined Cognitive Routing for Internet of Vehicles. *IEEE Internet of Things Journal*, v. 4662, n. c, p. 1–1, 2018. ISSN 2327-4662.

[95] LIBERATO, A. et al. Dynamic Backhauling Within Converged Networks. In: *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*. New York, NY, USA: ACM, 2016. (LANCOMM '16), p. 31–33. ISBN 978-1-4503-4426-5.

[96] LI, S. et al. Latency-Aware Task Assignment and Scheduling in Collaborative Cloud Robotic Systems. In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. San Francisco, CA, USA: IEEE, 2018. p. 65–72. ISBN 978-1-5386-7235-8.

[97] GOTSIS, A.; STEFANATOS, S.; ALEXIOU, A. UltraDense Networks: The New Wireless Frontier for Enabling 5G Access. *IEEE Vehicular Technology Magazine*, v. 11, n. 2, p. 71–78, jun. 2016.

[98] MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. [S.l.], 2011. 7 p.

[99] KOUBAA, A. A Service-Oriented Architecture for Virtualizing Robots in Robot-as-a-Service Clouds. In: HUTCHISON, D. et al. (Ed.). *Architecture of Computing Systems – ARCS 2014*. Cham: Springer International Publishing, 2014. v. 8350, p. 196–208. ISBN 978-3-319-04890-1 978-3-319-04891-8.

[100] VARRASI, S. et al. IBM Cloud Services Enhance Automatic Cognitive Assessment via Human-Robot Interaction. In: CARBONE, G.; CECCARELLI, M.; PISLA, D. (Ed.). *New Trends in Medical and Service Robotics*. Cham: Springer International Publishing, 2019. v. 65, p. 169–176. ISBN 978-3-030-00328-9 978-3-030-00329-6.

[101] CHEN, Y.; DU, Z.; García-Acosta, M. Robot as a Service in Cloud Computing. In: *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*. [S.l.: s.n.], 2010. p. 151–158.

[102] ARUMUGAM, R. et al. DAvinCi: A cloud computing framework for service robots. In: *2010 IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2010. p. 3084–3089.

[103] LU, H.; Hai-Shan, C.; Ting-Ting, H. Research on Hadoop Cloud Computing Model and its Applications. In: *2012 Third International Conference on Networking and Distributed Computing*. [S.l.: s.n.], 2012. p. 59–63.

[104] WAIBEL, M. et al. RoboEarth: A World Wide Web for Robots. *Robotics and Automation Magazine*, v. 18, n. 2, p. 69–82, 2011. ISSN 1070-9932.

[105] DU, Z. et al. Robot Cloud: Bridging the power of robotics and cloud computing. *Future Generation Computer Systems*, v. 74, p. 337–348, set. 2017. ISSN 0167739X.

[106] CARMO, A. P. do et al. Programmable intelligent spaces for Industry 4.0: Indoor visual localization driving attocell networks. *Transactions on Emerging Telecommunications Technologies*, abr. 2019. ISSN 2161-3915.

[107] WANG, X. V. et al. Ubiquitous manufacturing system based on Cloud: A robotics application. *Robotics and Computer-Integrated Manufacturing*, v. 45, p. 116–125, jun. 2017. ISSN 07365845.

[108] LIMOSANI, R. et al. Enabling Global Robot Navigation Based on a Cloud Robotics Approach. *International Journal of Social Robotics*, v. 8, n. 3, p. 371–380, jun. 2016. ISSN 1875-4791, 1875-4805.

[109] GAO, Q.; CHENG, H. Design of a CloudROS enabled Mobile Robot. In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.: s.n.], 2018. p. 2487–2492.

[110] SALMERÓN-GARCÍA, J. et al. Study of Communication Issues in Dynamically Scalable Cloud-Based Vision Systems for Mobile Robots. In: KOUBAA, A.; SHAKSHUKI, E. (Ed.). *Robots and Sensor Clouds*. Cham: Springer International Publishing, 2016. v. 36, p. 33–52. ISBN 978-3-319-22167-0 978-3-319-22168-7.

[111] OTHMAN, Z. et al. Comparison on Cloud Image Classification for Thrash Collecting LEGO Mindstorms EV3 Robot. v. 2, n. 1, p. 6, 2018.

[112] LEVINE, S. et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, v. 37, n. 4-5, p. 421–436, abr. 2018. ISSN 0278-3649, 1741-3176.

[113] BAYRAM, B.; İNCE, G. Advances in Robotics in the Era of Industry 4.0. In: *Industry 4.0: Managing The Digital Transformation*. Cham: Springer International Publishing, 2018. p. 187–200. ISBN 978-3-319-57869-9 978-3-319-57870-5.

[114] HUSSNAIN, A.; FERRER, B. R.; LASTRA, J. L. M. Towards the deployment of cloud robotics at factory shop floors: A prototype for smart material handling. In: *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. [S.l.: s.n.], 2018. p. 44–50.

[115] SCHEIDEGGER, W. M. et al. A Novel Multimodal Cognitive Interaction for Walker-Assisted Rehabilitation Therapies. In: *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*. [S.l.: s.n.], 2019. p. 905–910.

[116] TIAN, N. et al. A Cloud-Based Robust Semaphore Mirroring System for Social Robots. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. [S.l.: s.n.], 2018. p. 1351–1358.

[117] QUIGLEY, M. et al. ROS: An open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*. Kobe, Japan: [s.n.], 2009. v. 3, p. 5.

[118] FITZPATRICK, P. et al. A middle way for robotics middleware. p. 8, 2014.

[119] BRUYNINCKX, H. Open robot control software: The OROCOS project. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. [S.l.: s.n.], 2001. v. 3, p. 2523–2528 vol.3. ISSN 1050-4729.

[120] HAJJAJ, S. S. H.; SAHARI, K. S. M. Establishing remote networks for ros applications via port forwarding: A detailed tutorial. *International Journal of Advanced Robotic Systems*, v. 14, n. 3, p. 1729881417703355, 2017.

[121] SORRENTINO, A.; CAVALLO, F.; FIORINI, L. A plug and play transparent communication layer for cloud robotics architectures. *Robotics*, v. 9, n. 1, p. 17, 2020.

[122] SCHWARZ, M. et al. Supervised Autonomy for Exploration and Mobile Manipulation in Rough Terrain with a Centaur-Like Robot. *Frontiers in Robotics and AI*, v. 3, out. 2016. ISSN 2296-9144.

[123] TIDERKO, A.; HOELLER, F.; RÖHLING, T. The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems. In: KOUBAA, A. (Ed.). *Robot Operating System (ROS)*. Cham: Springer International Publishing, 2016. v. 625, p. 629–650. ISBN 978-3-319-26052-5 978-3-319-26054-9.

[124] HARMS, H. et al. Development of an Adaptable Communication Layer with QoS Capabilities for a Multi-Robot System. In: OLLERO, A. et al. (Ed.). *ROBOT 2017: Third Iberian Robotics Conference*. Cham: Springer International Publishing, 2018. v. 694, p. 782–793. ISBN 978-3-319-70835-5 978-3-319-70836-2.

[125] TARDIOLI, D.; PARASURAMAN, R.; ÖGREN, P. Pound: A multi-master ROS node for reducing delay and jitter in wireless multi-robot networks. *Robotics and Autonomous Systems*, v. 111, p. 73–87, jan. 2019. ISSN 0921-8890.

[126] RAHMAN, A. et al. Communication-Aware Cloud Robotic Task Offloading With On-Demand Mobility for Smart Factory Maintenance. *IEEE Transactions on Industrial Informatics*, v. 15, n. 5, p. 2500–2511, maio 2019. ISSN 1551-3203.

[127] CHEN, W. et al. When uav swarm meets edge-cloud computing: The qos perspective. *IEEE Network*, v. 33, n. 2, p. 36–43, 2019.

[128] PAHL, C. Containerization and the paas cloud. *IEEE Cloud Computing*, v. 2, n. 3, p. 24–31, 2015.

[129] ANDIAPPAN, V.; WAN, Y. K. Distinguishing approach, methodology, method, procedure and technique in process systems engineering. *Clean Techn Environ Policy*, v. 22, p. 547–555, 2020.

[130] CARDOSO, D. G. *Dimensionamento Vertical Automatico de Recursos Em Nuvens Computacionais*. Tese (Master's Thesis) — Universidade Federal do Espirito Santo, Brazil, 2019.

[131] CHAKRABORTI, T. et al. Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots in a Mixed-Reality Workspace. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, 2018. p. 4476–4482. ISBN 978-1-5386-8094-0.

[132] DING, X. et al. Communication constrained cloud-based long-term visual localization in real time. *arXiv:1903.03968 [cs]*, mar. 2019.

[133] GONZALEZ, A. G. C. et al. Supervisory Control-Based Navigation Architecture: A New Framework for Autonomous Robots in Industry 4.0 Environments. *IEEE Transactions on Industrial Informatics*, v. 14, n. 4, p. 1732–1743, abr. 2018. ISSN 1551-3203, 1941-0050.

[134] KAHN, G. et al. Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, 2018. p. 5129–5136. ISBN 978-1-5386-3081-5.

[135] HU, J.-S.; WANG, J.-J.; HO, D. M. Design of Sensing System and Anticipative Behavior for Human Following of Mobile Robots. *IEEE Transactions on Industrial Electronics*, v. 61, n. 4, p. 1916–1927, abr. 2014. ISSN 0278-0046, 1557-9948.

[136] SI-MOHAMMED, H. et al. Towards bci-based interfaces for augmented reality: Feasibility, design and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, p. 1–1, 2018. ISSN 2160-9306.

[137] NETO, A. F. et al. Extraction of user's navigation commands from upper body force interaction in walker assisted gait. *BioMedical Engineering OnLine*, v. 9, n. 1, p. 37, ago. 2010. ISSN 1475-925X.

[138] STORMS, J.; CHEN, K.; TILBURY, D. A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay. *The International Journal of Robotics Research*, v. 36, n. 5-7, p. 820–839, jun. 2017. ISSN 0278-3649, 1741-3176.

[139] BOHREN, J. et al. Semi-autonomous telerobotic assembly over high-latency networks. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, 2016. p. 149–156. ISBN 978-1-4673-8370-7.

[140] LI, Z. et al. 5G URLLC: Design Challenges and System Concepts. In: *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. [S.l.: s.n.], 2018. p. 1–6.

[141] AUTOWARE. *D2.3 AUTOWARE Wireless Industrial Communications and Networking*. [S.l.], 2018. 70 p.

[142] BHAMARE, D. et al. A survey on service function chaining. *Journal of Network and Computer Applications*, v. 75, p. 138 – 155, 2016. ISSN 1084-8045.

[143] RüTH, J. et al. Towards in-network industrial feedback control. In: *Proceedings of the 2018 Morning Workshop on In-Network Computing*. New York, NY, USA: Association for Computing Machinery, 2018. (NetCompute '18), p. 14–19. ISBN 9781450359085.

[144] RODRIGUEZ, F.; ROTHENBERG, C. E.; PONGRáCZ, G. In-network p4-based low latency robot arm control. In: *Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies*. New York, NY, USA: Association for Computing Machinery, 2019. (CoNEXT '19), p. 59–61. ISBN 9781450370066. Disponível em: <https://doi.org/10.1145/3360468.3368178>.

[145] MAFIOLETTI, D. R. et al. Piaffe: A place-as-you-go in-network framework for flexible embedding of vnfs. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2020. p. 1–6.

[146] LE BOUDEC, J.-Y.; THIRAN, P. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. 1. ed. Berlin: Springer, 2001. (Lecture notes in computer science).

[147] FANG, W.; WANG, L.; REN, P. Tinier-yolo: A real-time object detection method for constrained environments. *IEEE Access*, v. 8, p. 1935–1944, 2020.

[148] REDMON, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788.

APPENDIX A – Statistical Annalysis of the Results from Chapter 2

In this appendix, we detail the statistical analysis performed to evaluate the results of the pilot study presented in Chapter 2. Therefore, this appendix complements and substantiates the findings presented in Section 2.6 and should not be considered separately.

Before starting the statistical analysis, the data collected through the questionnaires were carefully processed, coded, and consolidated in the CSV format. From there, we could test statistical hypotheses regarding the impact of the round-trip latency on the dependent variables (QoE, perception of control, and perception of safety). In this process, we extensively used some specialized statistical packages, such as RStudio v1.1.3831 and G*Power2 v3.1.9.3.

In the statistical analysis of the data, we initially considered the repeated measures analysis of variance (ANOVA). The analysis of variance is used to compare if there are significant differences among means of a dependent variable as a function of a single independent variable (called a factor) with 2 or more categories or levels. As previously stated, the only independent variable is the RTT.

Given that our questionnaire consists of Likert scale questions, in which the distance between two adjacent points can be unequal, we used the Friedman test, considered the non-parametric equivalent of the repeated measures ANOVA to analyze our data. For the Friedman test, the prerequisites can be considered more relaxed. For example, a group or category needs to be measured three or more times, the measurements need to be independent of each other, the dependent variables must be measured at the continuous or ordinal level, and the samples do not need to be normally distributed. It is noteworthy that our data set satisfied all these requirements.

We used G*Power to investigate whether we could detect statistically significant differences with our sample size. We used the analysis of variance with repeated measures and specified the effect size = 0.253 (mean effect), $p = 0.05$ (significance level α or Type I error), $power = 0.95$, number of groups equal to 1 (analysis within the group) and number of measures per participant equal to 4 (number of RTT conditions). We found that a total of 35 participants were required to obtain a statistically significant result with the parameters provided. In our case ($N = 21$), there was an approximately 77 % chance of correctly rejecting the Null Hypotheses H_0, which can be stated as:

Null hypotheses. *H_0: There were no statistically significant differences between the means of users' reported QoE when using the CloudWalker under different RTT conditions* ($\mu_{QoE\_RTT\_0ms} = \mu_{QoE\_RTT\_100ms} = \mu_{QoE\_RTT\_300ms} = \mu_{QoE\_RTT\_500ms}$).

In the evaluation of the Null Hypothesis H_0, the independent variable included 4 categorical, independent groups: 0 ms ($\mu_{QoE\_RTT\_0ms} = 4.29$, $\sigma = 0.56$, Mdn = 4, n = 21), 100 ms ($\mu_{QoE\_RTT\_100ms} = 3.86$, $\sigma = 0.96$, Mdn = 4, n = 21), 300 ms ($\mu_{QoE\_RTT\_300ms} = 4.05$, $\sigma = 0.67$, Mdn = 4, n = 21), and 500 ms ($\mu_{QoE\_RTT\_500ms} = 3.67$, $\sigma = 0.73$, Mdn = 4, n = 21). We performed the Friedman test in RStudio and found a statistically significant result, $\chi^2(3) = 12.45$ and $p = 0.0060$.

We then performed a post-hoc analysis using the Wilcoxon signed-rank tests with Bonferroni adjustment, resulting in an adjusted significance level of $p < 0.008$. We found that there was no statistically significant difference between the RTT values of $0ms$ and 100 ms ($Z = 1.59$ and $p = 0.14$), of 0 ms and 300 ms ($Z = 1.64$ and $p = 0.16$), of 100 ms and 300 ms ($Z = -0.57$ and $p = 0.56$), of 100 ms and 500 ms ($Z = 1.31$ and $p = 0.18$) and of 300 ms and 500 ms ($Z = 2.21$ and $p = 0.03$). However, we did find a statistically significant difference between the RTT values of 0 ms and 500 ms ($Z = 3.22$ and $p = 0.001$). Therefore, in this case, we reject the Null Hypothesis H_0 ($\mu_{QoE\_RTT\_0ms} \neq \mu_{QoE\_RTT\_500ms}$) and state there are differences in the mean value of user's reported QoE under those two test conditions. Such a result point to QoE degradation under the worst QoS condition.

We followed the same procedure described above to analyze the effect of the RTT factor on the other dependent variables such as the perception of control and the perception of safety when using the CloudWalker. In our analysis, we did not find a statistically significant result for none of these variables (perception of control: $\chi^2(3) = 2.20$, $p = 0.53$ and perception of safety: $\chi^2(3) = 1.53$, $p = 0.68 > 0.05$ significance level α). In other words, this means that we cannot reject any of these Null Hypotheses based on our sample of participants and measurements.