robotics

MDPI

Article

A Plug and Play Transparent Communication Layer for Cloud Robotics Architectures

Alessandra Sorrentino [1,2], Filippo Cavallo [1,2,3] and Laura Fiorini [1,2,*]

[1] The BioRobotics Insititute, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera, Italy; alessandra.sorrentino@santannapisa.it (A.S.); filippo.cavallo@santannapisa.it (F.C.)
[2] Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna, Piazza Martiri della Libertà 33, 56127 Pisa, Italy
[3] Department of Industrial Engineering, University of Florence, Via Santa Marta 3, 50139 Florence, Italy
* Correspondence: laura.fiorini@santannapisa.it

Received: 31 January 2020; Accepted: 20 March 2020; Published: 22 March 2020

Abstract: The cloud robotics paradigm aims at enhancing the abilities of robots by using cloud services, but it still poses several challenges in the research community. Most of the current literature focuses on how to enrich specific robotic capabilities, overlooking how to effectively establish communication between the two fields. Our work proposes a "plug-and-play" solution to bridge the communication gap between cloud and robotic applications. The proposed solution is designed based on the mature WebSocket technology and it can be extended to any ROS-based robotic platform. The main contributions of this work are the definition of a reliable autoconnection/autoconfiguration mechanism as well as to outline a scalable communication layer that allows the effective control of multiple robots from multiple users. The "plug-and-play" solution was evaluated in both simulated and real scenarios. In the first case, the presence of users and robots was simulated with Robot Operating System (ROS) nodes running on five machines. In the real scenario, three non-expert users teleoperated, simultaneously, three remote robots by using the proposed communication layer with different networking protocols. Results confirmed the reliability at different levels: at startup (success_rate = 100%); during high-rate communications (message_lost = 0%); in performing open-loop spiral trajectories with enhancement, with respect to similar works; and in the quality of simultaneous teleoperations.

Keywords: cloud robotics; system architecture; remote control; tele-operated robots

1. Introduction

Over the past decades, robotics solutions have been applied to several real-world problems in a broad list of contexts, like unmanned search and rescue [1], healthcare [2] and medical applications [3]. If, from one side, there is the requirement of improving the autonomous capability of the robotic platforms, on the other side, there is the need of being in control of the platforms, adopting a teleoperating approach. As described by [4], several robotic systems are teleoperated based on the Internet, by using appropriate communication architectures and network infrastructures. In recent years, constant improvements in telecommunication infrastructures and the recent growth of cloud technology have led the birth of a new branch of research, namely cloud robotics, where cloud solutions are used to enhance the abilities of robots. The cloud robotics paradigm can be defined as "the combination of cloud computing and robotics" [5]. The concept is "not related to a new kind of robot but to the way in which robots access and store information". Cloud robots have recently been defined as "any robot or automation system that relies on either data or code from a network to

support its operation, where not all sensing, computation, and memory are integrated into a single standalone system" [6]. Nowadays, cloud robotics solutions are applied in different applications [7–9].

Beyond these specific applications, several works focus on the definition and implementation of the architecture for communication and interaction between physical robots and virtual resources hosted on cloud infrastructure [10–14]. Different solutions have been provided, but the problem of bi-directional communication among agents (i.e., from a user interface to a mobile robot platform and vice-versa) on different networks is often not addressed or underestimated. Virtual Private Networks (VPNs) are usually introduced as an operative solution to solve visibility issues, but this does not take into account the effort needed for the configuration of each agent in the virtual network.

This paper aims to describe a "plug and play" communication layer which is untied from the hardware composing the system (no further setup work when new hardware is integrated) and which guarantees stability in any situation where a set of robots has to be remotely controlled by agents (e.g., user interfaces) outside the robots' network. As in the case of VPN configurations, the "plug-and-play" solution relies on configuration methods rather than on the implementation of new software. The rationale beyond this choice is related to the goal of achieving a reliable system starting from mature technologies already available. This paper represents and describes a use-case in using technologies from the world of telecommunication networks into the robotic world. This could represent a interesting point of view for the robotics community. The proposed approach is based on:

- The WebSocket protocol, which allows a full-duplex communication client-server, solving the bi-directional visibility issue;
- Reverse tunneling, which is a popular technique to establish a connection with remote devices;
- Port remapping, to automatically request the connection on dedicated ports. (To manage the presence of multiple involved agents, the information related to the association between devices and ports is stored on a central server.)

Consequently, the challenge addressed in our work is to define a reliable autoconnection/autoconfiguration mechanism as well as to outline a scalable communication layer that allows the effective control of multiple robots from multiple users.

2. Related Works

Cloud robotics provides an efficient solution for migrating intensive computation from the robot side to the cloud computing infrastructure. Among the several cloud-based applications proposed in the literature, the challenges of bi-directional visibility and communication among agents (i.e., robots and cloud resources) are often underestimated or based on non-"plug-and-play" solutions such as VPNs. Besides, in [11], the authors state three challenges of computation offloading:

1. Traditional approaches do not consider the characteristics of networked cloud robotics (NCR) (e.g., heterogeneity and robotic cooperation);
2. They fail to capture the characteristics of tasks in a robotic streaming workflow (RSW) (e.g., strict latency requirements and varying task semantics);
3. They do not consider Quality-of-Service (QoS) issues for cloud robotics.

In the aforementioned paper, a QoS-aware robotic streaming work-flow allocation algorithm for networked cloud robotics, with joint optimization of latency, energy efficiency, and cost, while considering the characteristics of both robotic streaming workflow and networked cloud robotics, is proposed. Focusing on networked robotics (i.e., agents are part of the same network), the problem of agents' visibility is not included as an issue to be tackled. Similarly, in [12], authors focus on the managing of cloud resources arguing the presence of technical challenges for multi-robot systems on accessing the cloud and retrieve resources in near real-time. A general framework for setting up a cloud robotics system with a novel resource management strategy is presented, and the problem is formally described

as a Stackelberg game, proposing an optimal solution with proof. QoS criteria are then defined and evaluated, without describing the operational method for providing connections among cloud and robot resources. In [13], cloud robotics is claimed as "one of the most promising applications in the robotics world", but its growth is still below expectations due to the risks associated with security and privacy. Therefore, the paper focuses on security for cloud-based robotic services and it explains a framework that provides authentication and key agreement using Elliptic Curve Cryptography (ECC) for accessing the robotic services hosted in the cloud. Nevertheless, despite the interesting results on robustness against various security attacks, it does not detail the bi-directional communication infrastructure for agents in different networks.

In [14], authors aim to integrate the cyber world and the physical world by bringing up the idea of "Robot Cloud" to combine the power of robotics and cloud computing. To make this possible, they design a novel Robot Cloud stack and adopt the service-oriented architecture (SOA) to make the functional modules in the Robot Cloud more flexible, extensible and reusable. In particular, at a functional level, the last command is retrieved by sending an HTTP request periodically (i.e., *polling*) to the scheduling service and receiving their commands in the XML format as the response, in a similar way to what most SOA applications do. However, it can be argued and demonstrated that polling solutions are not feasible for low-level remote control (as teleoperation), due to the high rate of commands. Furthermore, the continuous requests could overload the use of the bandwidth. Focusing the attention on the "robot side", several frameworks have been presented to foster the development of robotic applications, but the Robot Operating System (ROS) [15] can be considered as the *de facto* operating system for robots. As a consequence, several works focus on cloud-ROS case studies rather than more generic cloud-robots.

In [16], a framework that helps users to work with ROS in a remote master was presented, based on the use of SpaceBrew [17]. SpaceBrew is defined as "*an open, dynamically re-routable software toolkit for choreographing interactive spaces*". A web-based visual switchboard can be used to connect or disconnect publishers and subscribers to each other. Unfortunately, the documentation provided on SpaceBrew is still under development and it is not clear if visibility among agents is requested or managed by SpaceBrew. However, in the architecture presented in [17], some limitations are presented but not discussed, as ''*it is not possible to use SpaceBrew with two computers connected at the same network*". Moreover, the results presented are limited to a qualitative analysis.

In the work presented in [18], ROS packages are encapsulated as web services in cloud virtual machines and a middleware based on web service technology is designed as the core of the whole cloud robotics system. This is responsible for parsing the cloud robotics task requests and scheduling ROS nodes in a distributed network. The communication is based on a proxy virtual machine and the presented results are limited to one robot, with claims that the situation with multiple robots will be considered in subsequent research.

In [19], the authors propose a software cloud architecture for cloud robotics which is intended for three subsystems in the cloud environment: the middleware subsystem, the background tasks subsystem, and the control subsystem. The architecture invokes cloud technologies such as cloud computing, cloud storage, and other networking platforms arranged with the assistance of congregated infrastructure and shared services for robotics, for instance, the Robot Operating System (ROS). Bi-directional communications are presented in the paper, but it is not detailed how these are achieved. Furthermore, the proposed architecture is built upon the ROS Multimaster FKIE [20], which requires bi-directional visibility among all the agents in the system.

RoboEarth [21] is one of the most important European funded projects on cloud robotics and focuses on the collection, storing, and sharing of data independent of specific robot hardware. With RoboEarth, the Rapyuta Cloud Engine [22] has been developed; it is based on the WebSocket protocol to guarantee the bi-directionality of the flow of data, but neither RoboEarth nor Rapyuta detail the visibility requirements between the agents. Furthermore, case studies did not involve the analysis of forwarding control from remote users to the robot. Nowadays, Rapyuta is a company

with its headquarters in Tokyo [23], and provides a cloud robotics framework at an enterprise level. However, it is still limited in an early developer program.

Since the beginning of the 2010s, the rosbridge protocol has been introduced. It is a specification for sending JavaScript Object Notation (JSON) based commands to ROS (and in theory, to any other robot middleware). The specification is a programming language and is transport agnostic. The idea is that any language or transport that can send JSON can talk the rosbridge protocol and interact with ROS. The protocol covers subscribing and publishing topics, service calls, getting and setting parameters, and even compressing messages. Upon the rosbridge protocol, several tutorials and applications have been developed [24–27]. Among these, Robot Web Tools [28] represents the most successful and widespread implementation. Since its official introduction in 2012, the Robot Web Tools project has grown tremendously as an open-source community, enabling new levels of interoperability and portability across heterogeneous robot systems, devices, and front-end user interfaces. At the heart of Robot Web Tools is the rosbridge protocol as a general means for messaging ROS topics in a client–server paradigm suitable for wide area networks, and human–robot interaction at a global scale through modern web browsers. On the other hand, while the rosbridge library provides the JSON<->ROS conversion, it leaves the transport layer to others. This can be overcome through the use of rosbridge server, which provides a WebSocket connection. As a common use, a rosbridge server runs locally on the robot platform allowing to reach ROS topic and services through the rosbridge protocol from the outside. Nevertheless, this requires the visibility of the robot machine from the outside, a requirement that is hardly satisfied, especially for platforms that operate in a wireless local network.

In 2017, a new protocol has been introduced in [10] to integrate Robot Operating System (ROS) enabled robots with the Internet of Things, arguing for a lack of ROS functionality in monitoring and controlling robots through the Internet. Actually, the proposed protocol is very similar to the rosbridge protocol; the tests are limited to a few simulated robots, and the results obtained in the performing of a spiral trajectory with an open-loop control using commands from the remote server are not promising.

The solution developed in our work is proposed to endow the ability of remote control to a set of generic robots controlled using the ROS framework. As introduced, the rosbridge server provides a layer that can run on the local machine upon the ROS framework, providing an interface for communications outside the local machine using the rosbridge protocol. Therefore, the problem issued in this paper can be described as the definition of a method to remote control the desired robot (chosen from among a not-defined set) from a device (smartphone, tablet, laptop, or desktop PC). Both controlled robots and controlling devices are in different private networks; in other words, there is no visibility between robot and user device.

The current state of the art, described in this paragraph, is summarized in Table 1.

Table 1. Summary of other available solutions in the current state of the art.

References	Features of the System Used	Limitation of the Presented Work
[11]	QoS-aware robotic streaming work-flow allocation algorithm	Problem of agents' visibility not included
[12]	Problem formally described as Stackelberg game	No description of the operational method for providing connections among cloud and robot resources.
[13]	Security for cloud-based robotic services	Bi-directional communication infrastructure for agents in different networks not detailed
[14]	Last commands retrieved by sending an HTTP request periodically (i.e., *polling*)	Polling solutions are not feasible for low-level remote control (as teleoperation)
[16]	Based on the use of SpaceBrew	It is not clear if visibility among agents is requested or managed by SpaceBrew. Some limitations are presented in the paper but not discussed

Table 1. *Cont.*

References	Features of the System Used	Limitation of the Presented Work
[18]	Communication based on a proxy virtual machine	Listed results are limited to one robot
[19]	Based on three subsystems in the cloud environment: middleware subsystem, background tasks subsystem, and control subsystem	Built upon the ROS Multimaster FKIE [20], which requires bi-directional visibility among all the agents in the system
[21]	Based on WebSocket protocol to guarantee the bi-directionality of the flow of data	Visibility requirements between the agents not detailed. Still limited in an early developer program
[10]	New protocol, similar to rosbridge protocol	Tests limited to few simulated robots
Presented system	Approach based on WebSocket communication, reverse tunnelling and port remapping according to information stored on a central server. Able to manage lack of visibility among agents	

3. System Description

The proposed system implements a communication layer for cloud-based robotic scenarios where multiple-users-multiple-robots are involved. Due to the variety of technologies, interoperability between heterogeneous devices (e.g., smartphone, tablets, computers) and different kinds of robots is required. Besides, the communication mechanism should guarantee real-time performance to enhance the user's experience with the robot, even remotely.

The configuration of the system is depicted in Figure 1. It is composed of a central server, namely a virtual machine hosted on a cloud service infrastructure, characterized by a static public IP that allows gateway porting as a Secure Shell (SSH) configuration[1]. We explicitly set GatewayPorts parameter in the file SSH configuration file. Concerning the architecture described in [29], the cloud resource is a Linux virtual machine running on FIWARE [30], which executes a LAMP server (Linux Apache MySQL PHP). The robotic platforms introduced in the system are based on ROS middleware for local control. On each robot, both the rosbridge server [31] and the ROS web video server [32] are running to allow incoming connections on default ports, one handled by the rosbridge protocol [33] and one dedicated for video streaming through video server. Since multiple agents are involved, a database is implemented on the LAMP server where each robot is mapped to a port number. The port number is defined as the MAC (Media Access Control) address of the robot network hardware, since it is unique information that can be used to characterize it. This strategy guarantees the flexibility of the system, since any ROS-based robot can be integrated into the scenario by saving its own information on the database. Human users can remotely control the robotic platforms through a web page running on the central server and accessible from any device.

Based on this configuration, each agent (user or robot) can be in a different private network, while only the server, characterized by a static public IP address, is reachable from outside. The bi-directional visibility between the elements of the system is obtained as described in the following paragraphs.

[1] In file /etc/ssh/sshd_config it should be explicit GatewayPorts yes.

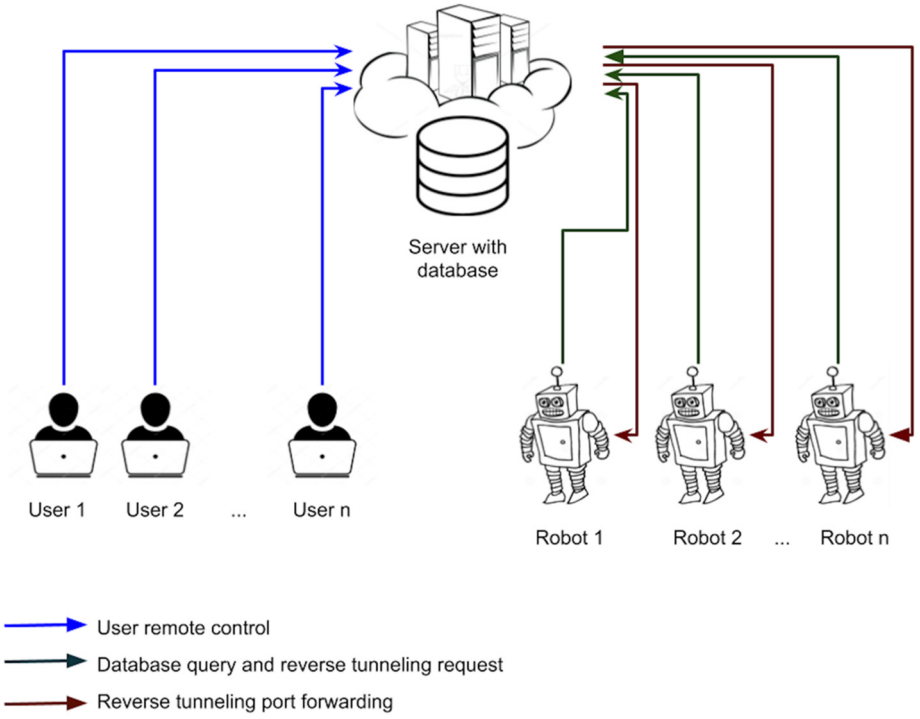

Figure 1. Server-based architecture scheme. Each agent (user or robot) can be in a different private network.

3.1. Robot Port Forwarding Configuration

At the startup phase, the robot can reach the server by executing "reverse SSH tunneling". This technique is an alternative strategy to Virtual Private Networks (VPN) to securely access a remote server that is behind a firewall. In this work, the robot automatically queries the database of the dedicated ports (for rosbridge controls and video streaming) related to its MAC address. If a record with the robot MAC address exists in the database, reverse tunneling is performed from the robot towards the server. The sequence of commands is summarized in the Unified Modeling Language (UML) diagram shown in Figure 2. The use of SSH reverse tunneling overcomes the problem of lack of bi-directional visibility: communication is opened by the robot (instead of by the server), creating a tunnel that allows bi-directional communication.

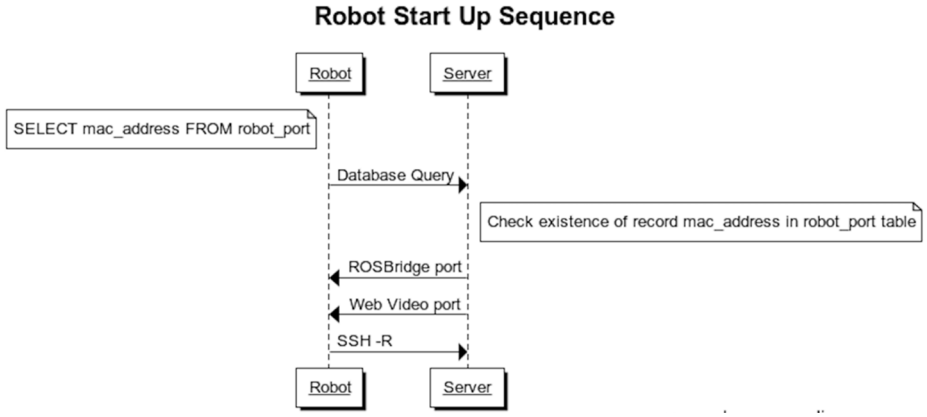

Figure 2. Unified Modeling Language (UML) sequence at robot start up. The database is queried and Secure Shell (SSH) reverse tunneling is performed.

3.2. User Remote Control

User remote control is performed through a web page hosted on the public server. The user can select the use of a specified robot and this choice is used to instantiate a rosbridge client and a video client pointing to server IP and the ports specified in the database. Therefore, the communication is forwarded through the opened reverse tunneling to the selected robot. This sequence is summarized in the UML diagram shown in Figure 3.

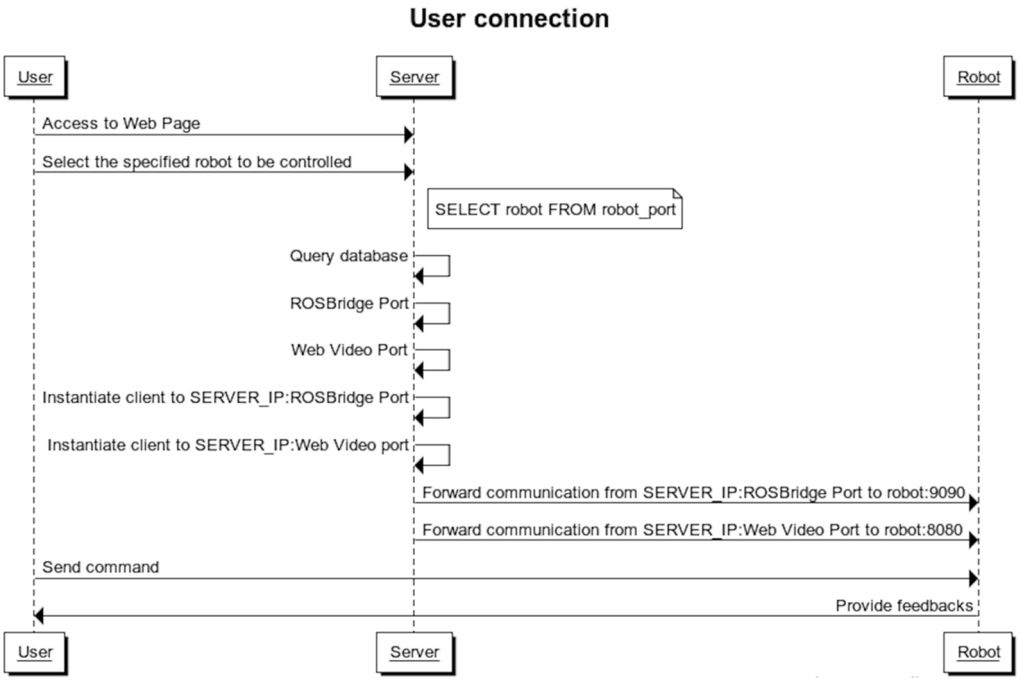

Figure 3. UML sequence at user connection. Opened SSH reverse tunneling is used to both forward user commands and retrieve feedbacks from the robotic platform.

Once the communication between the user and the robot is established, the user can send commands to the robot by using web pages stored on the server. By dedicated web pages, the user receives feedback on the requested commands (e.g., video streaming, executing velocity).

4. Experimentation

In this paper, five experimental setups were developed to investigate and to demonstrate the reliability and effectiveness of the proposed approach. In detail, tailored experiments have been performed to verify the following characteristics:

Requirement 1 (RE1): reliability of autoconnection/autoconfiguration at startup, without any specific action to the single machine;
Requirement 2 (RE2): scalability on several robots;
Requirement 3 (RE3): effectiveness of the remote control (e.g. teleoperation) from multiple users to multiple robots.

The experimental setups differ from each other based on the number of agents (users and robots), network infrastructures and scenarios. For the works described in [10,18], the number of user–robot pairs is always equal to or higher than three.

The first four proposed experimental setups evaluate the system in simulated scenarios, in which the agents involved (i.e., robots and users) have been approximated with a computer running several ROS cores. These tests reflect a practical scenario where the number of agents is high (e.g., more than three robots are involved). The final experimental setup recalls a real situation, where multiple

users interact with multiple robots. The robots used in this experimental phase are two Astro robotic platforms [34] and one Coro robotic platform [35]. Both robots are based on the SCITOS G5 mobile platform (Metralabs GmbH, Germany). They are equipped with a front and rear laser scanner to safely navigate the environment. Cameras for video streaming are also mounted. Astro and Coro platforms implement a teleoperation service, which allows a remote user to send velocity commands to the robot and to receive images of the environment where the robot is moving. Both robotic systems are developed based on the ROS framework. This section aims to describe the different setups, while the results are detailed in Section 5.

4.1. Test 1: Reliability at Startup

The first test aims to demonstrate the reliability at startup (RE1) by analyzing the eventual occurrence of problems in the sequence depicted in Figure 2.

A set of five local machines were used to locally instantiate 10 ROS cores for each machine. Multiple ROS cores can work on the same machine by changing the local port, running the dedicated command. A total of 50 ROS cores were set up. A rosbridge server was instantiated for each ROS core. For this specific purpose, the default port of the rosbridge server has been changed to allow multiple rosbridge servers on the same machines. Programs automatically started at the starting of the Ubuntu system. One simple std_msgs::String was sent from a remote machine to each ROS core through the described architecture to test the opening of communication, while a subscriber is already running waiting for receiving the message. The success rate of communication opening was computed through the check reception of the message sent. Test 1 was repeated 1000 times, each time using 50 ROS cores at the same time.

4.2. Test 2: High Rates Communication

The second test aimed to demonstrate the reliability of communications that involve messages at high frequency (RE2). A set of 100,000 std_msgs::String messages was sent at 100 Hz (i.e., one message every 0.01 s for 1000 s was sent) from five machines (users) to five different machines (robots), through the described architecture, using the server on FIWARE as in Test 1. A subscriber was already running, waiting to receive the messages. The metric used to evaluate the reliability of communication was data loss. It was computed as the count of receptions of the total messages sent. Furthermore, we evaluated the communication bandwith by using iperf [36] and we limited the server bandwidth through Wondershaper [37]. Test 2 was repeated 10 times, each time using five user–robot pairs.

4.3. Test 3: Different Network Hardware

In Tests 1 and 2, all the machines were connected to the same network infrastructure. To unbind the results from the hardware used, Tests 1 and 2 were repeated subdividing the groups of five machines into three sub-groups:

1. Machines connected to the building network infrastructure (fixed line);
2. Machines connected to a router with a 4 g connection;
3. Machines connected to another router with a 4 g connection (same phone company).

The evaluation metrics adopted in Tests 1 and 2 were used to evaluate the reliability of the system. This allows the comparison of the performance in the different scenarios.

4.4. Test 4: Open-Loop Spiral Trajectories

The effectiveness of the communication layer (RE3) was evaluated based on the experiment described in [10]. The experiment assesses real-time control of motion of Turtlesim robot (default simulator in ROS) following an open-loop spiral trajectory. A spiral trajectory is defined as a combination of an increasing linear velocity over time and a constant angular velocity. Speed commands have been remotely sent from five machines (users) to five different machines, each one running a Turtlesim robot,

through the described architecture, using the server on FIWARE as in Test 1. Since a spiral trajectory is sensitive to delays and jitters, the final pose of the simulated platform was used as a qualitative metric to evaluate real-time performance, namely the variability of the received commands along the path. Test 4 was repeated 10 times, each time using five user–robot pairs. Unfortunately, in [10] details of the network configuration are not provided.

4.5. Test 5: Qualitative Evaluation of Simultaneous Teleoperation

While the previous setups were performed in simulated scenarios, Test 5 involved the presence of real robots. In detail, remote teleoperation was evaluated for three users simultaneously controlling three different real robots (RE3). This experiment extends the results already obtained in [29], where only one-user-one-robot was considered. Since received speed commands were evaluated in Test 4, the analysis of performance in varying image resolution was computed. Namely, the image quality rate was tuned ranging from 90% to 50%. The teleoperation experiment setup is described in the following:

- One operator located in the Biorobotics Institute in Pontedera (PI, Italy) controls the Coro robot, situated in the Assistive Robotics Lab in Peccioli (PI, Italy);
- One operator situated in the Biorobotics Institute in Pontedera (PI, Italy) teleoperates the Astro 1 robot, located in the WVO Zorg Ter Reede residential care center in Vlissingen (ZE, Netherlands);
- One operator located in the WVO Zorg Ter Reede residential care center in Vlissingen (ZE, Netherlands) controls the Astro 2 robot, located in Ospedale Casa Sollievo della Sofferenza in San Giovanni Rotondo (FG, Italy).

Figure 4 summarizes the experimental setup. During the test, all the robotic platforms and one operator's laptop were connected to the WiFi network of the building, except for one operator that was connected with an Ethernet cable and another operator that was connected to a personal 4G router. The network speed specifications were computed using a speed test by the Ookla website and they are reported in Table 2. The network specifications reported in Table 2 are useful to assess the quality of connection at each robot side. The ping response time can be used to assess the latency; the download and the upload speeds provide additional information for evaluating the performance at the robot side. Due to this experimental design, the metrics used to evaluate the performance were the throughput in terms of packets per second(pps) and Mb/s Besides, the average delay between consecutive received packets at each teleoperation center was computed. The WireShark tool was used to analyze the HTTP packets received at each teleoperation center on the dedicated port.

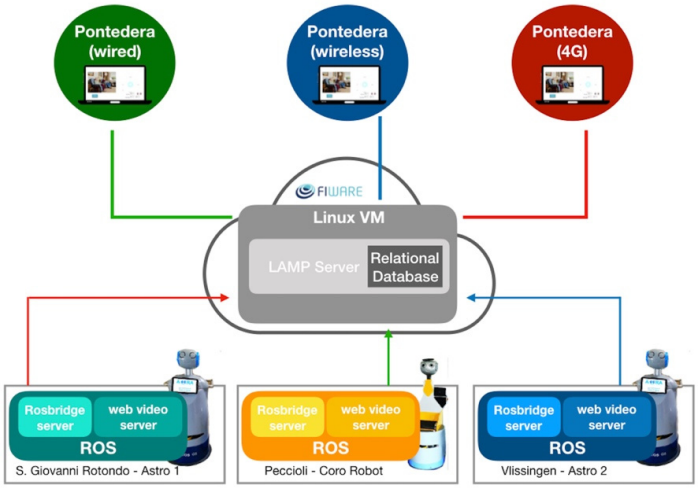

Figure 4. Experimental setup for Test 5. The association between user and robot is identified by the color of the arrows.

Table 2. Network speed specifications.

Network Name	Ping (ms)	Download (Mbps)	Upload (Mbps)
Pontedera (wired)	7	94.41	89.44
Pontedera (wireless)	7	287.77	113.18
Pontedera (4G)	38	7.13	4.97
San Giovanni Rotondo	19	31.13	8.91
Vlissingen	15	80.07	19.36
Peccioli	31	5.41	5.36

5. Results

The results obtained in all the performed tests demonstrate both the reliability and the feasibility of the "plug-and-play" system. Specifically, communication was successfully established in every trial of Test 1 (success_rate = 100%), and no message was lost through the network in Test 2 (message_lost = 0%). Additional analysis was performed to evaluate the influence of bandwidth. Bandwidth between the local machine and the FIWARE server was 39.4 Mbit/s during Test 2, and the obtained results highlight that the proposed architecture can work until the sum of all communications through the server (i.e., communications to every agent) is smaller than the server bandwidth. Communications bandwidths were measured through the analysis of ROS topic bandwidths. This confirmed that port forwarding through reverse tunneling does not increase the amount of bandwidth requested for communication. Test 3 confirmed the results obtained in Tests 1 and 2, demonstrating that the "plug-and-play" system does not depend on the network hardware. In Test 4, the parameters used to evaluate open-loops spiral trajectory were a rated frequency equal to 2 Hz, an initial velocity of 1.0 m/s, a constant angular velocity equal to 1.0 rad/s and a linear step equal to 0.05. The results were obtained by analyzing the final position of the simulated robot at the end of the open-loop in the overall 50 tests (Test 4 has been repeated 10 times, each time using five user–robot pairs). Results show that the proposed architecture can transmit open-loop commands, introducing a negligible variability $\sigma_x = 0.001$ m, $\sigma_y = 0.004$ m over a path long 78.275 m. Concerning the results reported in [10], the trajectories were correctly accomplished following spiral trajectories. The performance was not influenced by the presence of delays and jitters at the communication layer, which caused misbehaviours in [10], as shown in Figure 5.

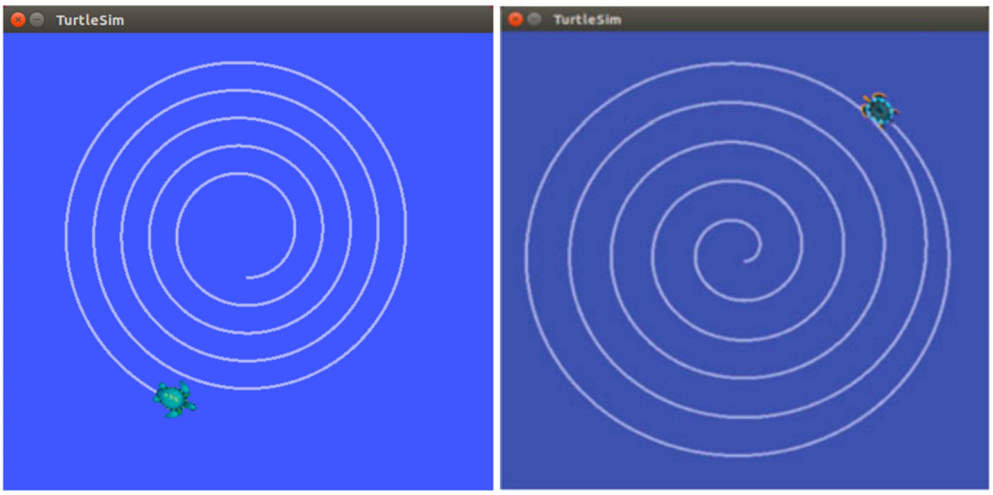

Figure 5. Visual comparison between spiral trajectories performed in open loop using the proposed architecture (on the left) and obtained through the ROSLink protocol (on the right) [10]. It can be highlighted that misbehaviors are present in the second case, especially at the end of the trajectory.

The results of Test 5 show that three operators can simultaneously control three robots without experiencing any significant delay able to corrupt their own experience. Quantitative results on their experience were collected by measuring the video streaming parameters during the teleoperation. In detail, five teleoperation modes were tested, each of them characterized by a different encoding quality of the image. A comparison between the packet size received by each operator at the teleoperation center is shown in Figure 6. As expected, the packet size decreased as the resolution of the image sent by the robot camera decreased. One misbehaviour was recorded for the video streaming at 70% sent by the Astro 1 robot, in which the packet size was bigger than 80% resolution. The average (μ) pps exchanged by FIWARE and the operator was almost stable (μ_{th} = 29 pps and σ_{th} = 0 pps for wireless connection, μ_{th} = 20.60 pps and σ_{th} = 2.41 pps for 4G connection), except for the experiment involving Coro robot (and the wired connection), in which the pps varied accordingly to the image resolution (μ_{th} = 10.80 pps and σ_{th} = 4.55 pps). The throughput reported in the graph has been calculated as the amount of http data sent over a certain period of time. It was thus influenced by the network speed both at the robot and operator's side. For the mentioned reason, the throughput value of images recorded by the Coro's camera (with the wired connection) was extremely low. As shown in the bottom-right graph in Figure 6, the most significant delay between consecutive packets was perceived in the case of the 4G connection (up to 190 ms in case of max resolution of the video streaming). The significant delay for the 4G connection derived both from the high ping value of the 4G connection (see Table 2) and from the low upload values at the robot's side, which was located at San Giovanni Rotondo, as shown in Figure 4.

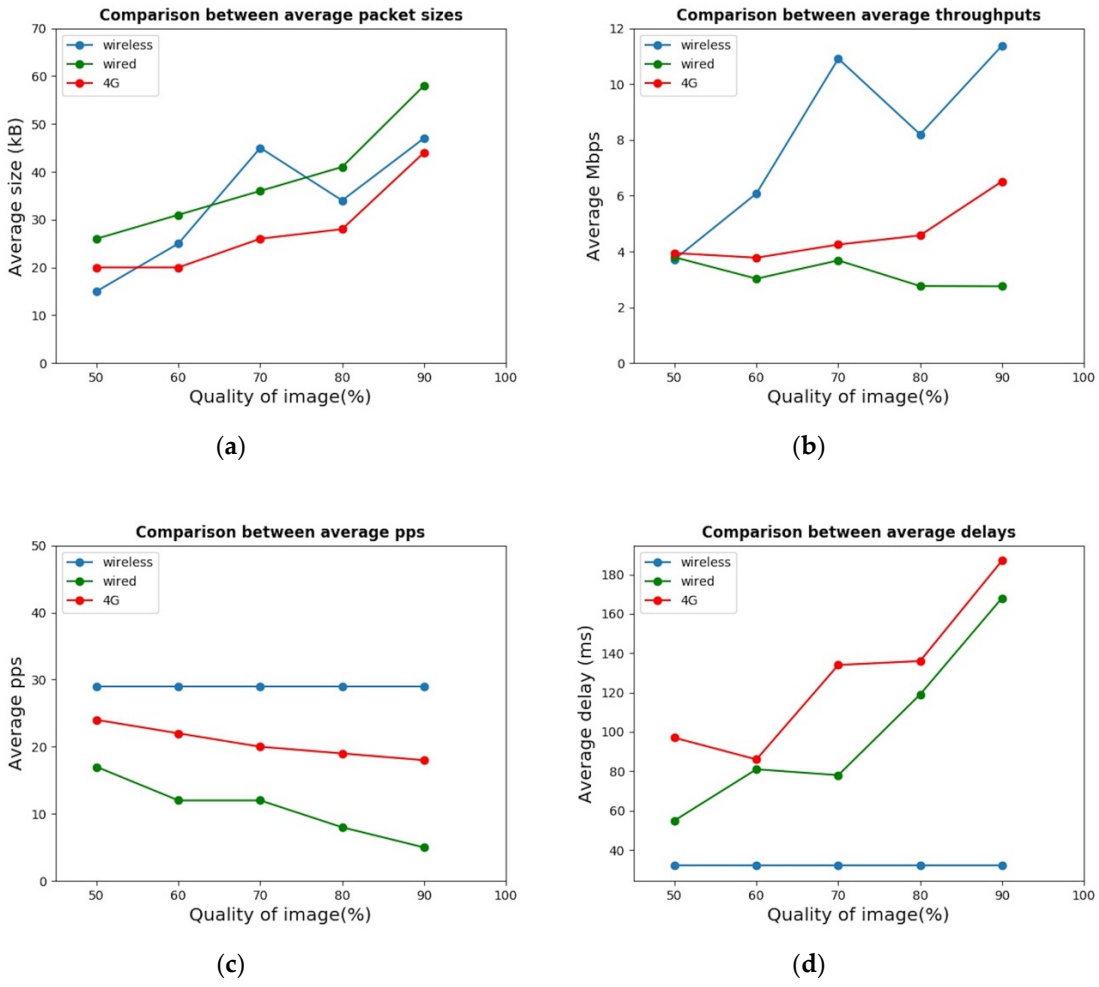

Figure 6. Quantitative results of the three simultaneous teleoperations: Avg. packet size (**a**), Avg. pps (**b**), Avg. throughput (**c**), Avg. delay (**d**).

6. Discussion

The rationale behind the development of the proposed architecture for cloud robotics relies on the need for a reliable solution that can allow the inclusion of new agents in the system without any specific configuration of the local machines. The term "plug and play" refers to the two main features of the system. First, a new robotic platform can be easily integrated into the system by adding a new record in the database. This strategy allows non-technical users to easily change the system configuration and it can also be performed remotely. Consequently, it is possible to increase or decrease the number of agents in the system in a flexible way. The second feature regards the types of technical elements involved in the architecture. The development relies on mature and dependable technologies, such as LAMP servers, WebSockets, SSH reverse tunneling, the rosbridge protocol, and servers. With respect to other approaches in the state of the art, the bi-directional visibility issue has been moved to a "configuration problem" rather than an "implementation problem". Due to the introduction of rosbridge technology, no new code or communication protocol is introduced to deal with the presented issue.

The experimentations and results reported in Sections 4 and 5 confirm the reliability and effectiveness of the proposed solution. On one side, the simulated scenarios validate the stability of the communication layer in case of a large number of agents involved (50 ROS cores in Tests 1 and 3) and in case of a high quantity of messages exchanged (100,000 messages in Tests 2 and 3). On the other side, the experimentation in the real scenario shows the efficiency of the approach with concrete agents due to the higher number of challenges, which are discarded in the simulated scenario. It is often the case that cloud-based applications' performances are affected by network glitches, bandwidth fluctuations which provoke irregular robot mobility [11]. Testing the communication layer in both kinds of scenarios provides precise evidence of its efficiency. Although our system satisfies the requirements detailed in Section 4, a few limitations have to be highlighted.

One concerns the network architecture topology. Since every communication travels through a central server, the resulting configuration recalls a typical start topology. This leads to the limitation that the sum of the bandwidth of every communication has to be smaller than the maximum server bandwidth. Besides, in the presence of robot reboots, the SSH tunneling is interrupted at robot shutdown and it is restarted after robot startup. This requires that, on the server, the port has to be released in this interval of time. By releasing the port in a certain interval of time, it may be possible to handle multiple incoming requests. In the context of a high number of agents connected through the cloud, the communication system should integrate a planner to allocate the available services in an efficient way. In the real scenario of the presented work, each agent directly accesses a dedicated robot, because the number of robots is limited. By increasing the number of elements in the networked cloud robotics, the presence of a service planner becomes essential.

As already introduced, the importance of untying the communication layer of architecture from the network currently used by the agents become more important in the scenario of mobile platforms, where the agents can use different WiFi connections. At the operator side, the reliability of the proposed solution despite the hardware used is confirmed by the results achieved in Tests 4 and 5. In the conducted experimentation, the delay between consecutive packets at the teleoperation center strictly depends on the network latency and the dimension of the information exchanged. The tradeoff between the data-rate of the communication channel and the quantity of data can be a limitation for cloud-based robotics applications. This leads to the challenge of defining the types of robotic tasks that should be kept on board. Cloud robotics provides a solution for intensive computation and storage of a high quantity of data, with no clear evidence of which types of information can be exchanged so that the delay at the receiver side (i.e., robot or operator side) is minimized.

As described in [37–39], ROS suffers from significant security issues. In our work, we address this issue by introducing an authentication system for the users that want to access the web pages hosted on the public server (as shown in Figure 3). This design choice can be improved by integrating secure HTTP and/or by enabling optional features of the rosbridge. It is worth noticing that one of the

improvements of the protocol is Transport Layer Security (TLS) support for WebSocket connections, an authorization mechanism to restrict Application Programming Interface (API) calls and to limit available topics.

In the current approach, the actual problem of bi-directional visibility has been solved by using a mature networking strategy (e.g., Websockets, reverse SSH, etc.) in a robotic domain. Indeed, to the best of the authors' knowledge, few robotics research works have used this approach in their research ([4,40]), still with a low number of robotic platforms. One possible direction of this work is to improve the proposed solution by adopting the current trends of networking approaches, such as the named-data-networking approach [41].

7. Conclusions

This paper aims to describe and evaluate an approach for the cloud robotics system to overcome the issue of bi-directional visibility, often not dealt with in related works. Even if virtual private networks represent an effective solution, their implementation requires specific interventions and configurations for each agent involved in the system. This kind of activity requires the involvement of technicians or experts.

The proposed system offers a "plug and play" solution, meaning that the configuration is automatically retrieved from a public database and that reverse tunneling allows any kind of protocol for the local connection (local WiFi, public WiFi, mobile 3G/4G, etc.). The main contributions of the proposed work are:

- The implementation of a reliable autoconnection/autoconfiguration mechanism;
- The design a scalable communication layer that allows the effective control of multiple robots from multiple users;
- The effective control of multiple heterogeneous robots from multiple users, since the communication layer is untied from the hardware components.

As a consequence, our work facilitates the setup operations needed to install a robotic system in a real scenario (outside the lab environment). Moreover, in the specific case of mobile platforms, a robot can travel among different WiFi networks in a large area. The proposed solution avoids managing particular configurations for each network, basing the communication on a public resource.

The developed system is based on mature technologies that allow achieving encouraging results on reliability and feasibility for remote control applications. Few limitations arose, mainly related to server performance in port-management and network architecture star topology. For instance, aspects of communication bandwidth have to be taken into account related to the specific application and to the number of agents simultaneously involved.

In conclusion, a deeper introduction of already developed technologies in the context of cloud robotics can strongly enhance the readiness of the technology, in particular by providing solutions that can reduce the need for intervention by expert users or developers.

Author Contributions: A.S. and L.F. were responsible for the methodology definition and for designing and conducing the experimental sessions. They contributed in structuring and writing the paper. F.C. was the scientific supervisor, guarantor for the work, and contributed in methodology definition, paper writing, discussion, and conclusion. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the ACCRA Project, founded by the European Commission – Horizon 2020 Founding Programme (H2020-SCI-PM14-2016) and National Institute of Information and Communications Technology (NICT) of Japan under grant agreement No. 738251.

Acknowledgments: The authors would like to thank the partners of ACCRA Project. Furthermore, the authors would like to thank Raffaele Limosani for supporting us into the design of experimental procedures.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tomic, T.; Schmid, K.; Lutz, P.; Domel, A.; Kassecker, M.; Mair, E.; Grixa, I.; Ruess, F.; Suppa, M.; Burschka, D. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robot. Autom. Mag.* **2012**, *19*, 46–56. [CrossRef]
2. Riek, L.D. Healthcare robotics. *Commun. ACM* **2017**, *60*, 68–78. [CrossRef]
3. Su, H.; Yang, C.; Ferrigno, G.; De Momi, E. Improved human-robot collaborative control of redundant robot for teleoperated minimally invasive surgery. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1447–1453. [CrossRef]
4. Silva, Y.M.L.R.; Simões, W.C.S.S.; Naves, E.L.M.; Bastos Filho, T.F.; De Lucena, V.F. Teleoperation training environment for new users of electric powered wheelchairs based on multiple driving methods. *IEEE Access* **2018**, *6*, 55099–55111. [CrossRef]
5. Kuffner, J. Cloud-enabled humanoid robots. In Proceedings of the 2010 10th IEEE-RAS International Conference Humanoid Robots (Humanoids), Nashville, TN, USA, 6–8 December 2010.
6. Kehoe, B.; Patil, S.; Abbeel, P.; Goldberg, K. A survey of research on cloud robotics and automation. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 398–409. [CrossRef]
7. Mohanarajah, G.; Usenko, V.; Singh, M.; D'Andrea, R.; Waibel, M. Cloud-based collaborative 3D mapping in real-time with low-cost robots. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 423–431. [CrossRef]
8. Bekris, K.; Shome, R.; Krontiris, A.; Dobson, A. Cloud automation: Precomputing roadmaps for flexible manipulation. *IEEE Robot. Autom. Mag.* **2015**, *22*, 41–50. [CrossRef]
9. Toffetti, G.; Bohnert, T.M. *Cloud Robotics with ROS*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 831, ISBN 9783030201906.
10. Koubaa, A.; Alajlan, M.; Qureshi, B. ROSLink: Bridging ROS with the Internet-of-Things for Cloud Robotics. In *Robot Operating System (ROS)*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 265–283.
11. Chen, W.; Yaguchi, Y.; Naruse, K.; Watanobe, Y.; Nakamura, K. QoS-aware Robotic Streaming Workflow Allocation in Cloud Robotics Systems. *IEEE Trans. Serv. Comput.* **2018**. [CrossRef]
12. Wang, L.; Liu, M.; Meng, M.Q.-H. Real-time multisensor data retrieval for cloud robotic systems. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 507–518. [CrossRef]
13. Nandhini, C.; Doriya, R. Towards secured cloud-based robotic services. In Proceedings of the 2017 International Conference Signal Processing and Communication (ICSPC), Coimbatore, India, 28–29 July 2017; pp. 165–170.
14. Du, Z.; He, L.; Chen, Y.; Xiao, Y.; Gao, P.; Wang, T. Robot Cloud: Bridging the power of robotics and cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 337–348. [CrossRef]
15. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, no. 3.2. p. 5.
16. Pereira, A.B.M.; Bastos, G.S. ROSRemote, using ROS on cloud to access robots remotely. In Proceedings of the 2017 18th International Conference Advanced Robotics (ICAR), Hong Kong, China, 10–12 July 2017; pp. 284–289.
17. SpaceBrew. Available online: http://docs.spacebrew.cc/ (accessed on 10 May 2018).
18. Luo, J.; Zhang, L.; Zhang, H.Y. Design of a cloud robotics middleware based on web service technology. In Proceedings of the 2017 18th International Conference Advanced Robotics (ICAR), Hong Kong, China, 10–12 July 2017; pp. 487–492.
19. Miratabzadeh, S.A.; Gallardo, N.; Gamez, N.; Haradi, K.; Puthussery, A.R.; Rad, P.; Jamshidi, M. Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots. In Proceedings of the World Automation Congress (WAC), Rio Grande, PR, USA, 31 July–4 August 2016; pp. 1–6.
20. Tiderko, A.; Hoeller, F.; Röhling, T. The ROS multimaster extension for simplified deployment of multi-robot systems. In *Robot Operating System (ROS)*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 629–650.
21. Waibel, M.; Beetz, M.; Civera, J.; d'Andrea, R.; Elfring, J.; Galvez-Lopez, D.; Häussermann, K.; Janssen, R.; Montiel, J.M.M.; Perzylo, A.; et al. Roboearth. *IEEE Robot. Autom. Mag.* **2011**, *18*, 69–82. [CrossRef]
22. Mohanarajah, G.; Hunziker, D.; D'Andrea, R.; Waibel, M. Rapyuta: A cloud robotics platform. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 481–493. [CrossRef]
23. Rapyuta Company. Available online: https://www.rapyuta-robotics.com/company/ (accessed on 10 May 2018).

24. Crick, C.; Jay, G.; Osentoski, S.; Pitzer, B.; Jenkins, O.C. Rosbridge: Ros for non-ros users. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 493–504.
25. Crick, C.; Jay, G.; Osentoski, S.; Jenkins, O.C. ROS and rosbridge: Roboticists out of the loop. In Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, Boston, MA, USA, 5–8 March 2012; pp. 493–494.
26. Lee, J. *Web Applications for Robots Using Rosbridge*; Brown University: Providence, RI, USA, 2012.
27. Blaha, M.; Krec, M.; Marek, P.; Nouza, T.; Lejsek, T. Rosbridge web interface. *Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Tech.* **2013**, *166*, 27.
28. Toris, R.; Kammerl, J.; Lu, D.V.; Lee, J.; Jenkins, O.C.; Osentoski, S.; Wills, M.; Chernova, S. Robot web tools: Efficient messaging for cloud robotics. In Proceedings of the 2015 IEEE/RSJ International Conference Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 4530–4537.
29. Manzi, A.; Fiorini, L.; Limosani, R.; Sincak, P.; Dario, P.; Cavallo, F. Use Case Evaluation of a Cloud Robotics Teleoperation System (Short Paper). In Proceedings of the 2016 5th IEEE International Conference Cloud Networking (Cloudnet), Pisa, Italy, 3–5 October 2016; pp. 208–211.
30. FIWARE Documentation. Available online: https://www.fiware.org/ (accessed on 10 May 2018).
31. ROSBridge Server Documentation. Available online: https://http//wiki.ros.org/rosbridge_server (accessed on 10 May 2018).
32. ROS Web Video Server Documentation. Available online: http://wiki.ros.org/web_video_server (accessed on 10 May 2018).
33. ROSBridge Protol Documentation. Available online: https://github.com/RobotWebTools/rosbridge_suite/blob/groovy-devel/ROSBRIDGE_PROTOCOL.md (accessed on 10 May 2018).
34. Fiorini, L.; De Mul, M.; Fabbricotti, I.; Limosani, R.; Vitanza, A.; D'Onofrio, G.; Tsui, M.; Sancarlo, D.; Giuliani, F.; Greco, A.; et al. Assistive robots to improve the independent living of older persons: Results from a needs study. *Disabil. Rehabil. Assist. Technol.* **2019**. [CrossRef] [PubMed]
35. Cavallo, F.; Esposito, R.; Limosani, R.; Manzi, A.; Bevilacqua, R.; Felici, E.; Di Nuovo, A.; Cangelosi, A.; Lattanzio, F.; Dario, P. Robotic services acceptance in smart environments with older adults: User satisfaction and acceptability study. *J. Med. Internet Res.* **2018**. [CrossRef] [PubMed]
36. IPERF Documentation. Available online: https://iperf.fr/ (accessed on 10 May 2018).
37. Dieber, B.; Breiling, B.; Taurer, S.; Kacianka, S.; Rass, S.; Schartner, P. Security for the Robot Operating System. *Rob. Auton. Syst.* **2017**, *98*, 192–203. [CrossRef]
38. Demarinis, N.; Tellex, S.; Kemerlis, V.P.; Konidaris, G.; Fonseca, R. Scanning the internet for ROS: A view of security in robotics research. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019.
39. Toris, R.; Shue, C.; Chernova, S. Message authentication codes for secure remote non-native client connections to ROS enabled robots. In Proceedings of the IEEE Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 14–15 April 2014.
40. Limosani, R.; Manzi, A.; Faggiani, A.; Bianchi, M.; Pagliai, M.; Ridolfi, A.; Allotta, B.; Dario, P.; Cavallo, F. Low-cost solution in international robotic challenge: Lessons learned by Tuscany Robotics Team at ERL Emergency Robots 2017. *J. F. Robot.* **2019**, *36*, 587–601. [CrossRef]
41. Amadeo, M.; Campolo, C.; Iera, A.; Molinaro, A. Named data networking for IoT: An architectural perspective. In Proceedings of the EuCNC 2014—European Conference on Networks and Communications, Bologna, Italy, 23–26 June 2014.