2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)

DewROS: a Platform for Informed Dew Robotics in ROS

Giovanni Stanco* , Alessio Botta, Giorgio Ventre
Department of Electrical Engineering and Information Technology
University of Napoli Federico II, Napoli, Italy
{gio.stanco}@studenti.unina.it, {alessio.botta}@unina.it, {giorgio.ventre}@unina.it,

***Abstract*—In recent years Cloud Robotics technology has been proposed to overcome the constraints imposed by the resources of standalone robots. We can imagine that in near future robots will be very present in everyday life and interact with humans, so it is necessary to guarantee that robots could make decisions even if the connection to the cloud is unavailable. It is then important to move the critical tasks on the edge devices in order to make them always accessible, not following the Cloud Robotics paradigm but the Dew Robotics one instead.**
In this paper we propose DewROS, a platform for Dew Robotics that uses monitoring entities to monitor the system status in order to adapt the application operating conditions. In particular in this work we describe the DewROS platform and its application in the case of video analysis in a surveillance scenario. The results provided in this paper demonstrate how DewROS allows us to exploit at their best the limited resources of our robots.

***Index Terms*—Cloud robotics, Dew robotics, ROS, SHERPA.**

I. Introduction

In the past decades robotics saw a significant development and has been applied to several fields of interest [1]. The robots used in past applications were standalone machines limited by their hardware and by their computing power. A couple of decades ago we saw the rise of *networked robotics* which connected a group of robots through a wired or wireless connection. Networked robotics allows the robots to share the perceived data and to solve a task in a cooperative manner, however networked robots encounter some problems as the standalone robots. The main problem is that there are limitations on the computing and storage capacity caused by the constraints imposed by the hardware of each robot and by the difficulties to upgrade the resources once a robot has been designed and deployed. Their limited resources are a drawback in complex real-life problems that need real-time execution and in machine-to-machine communication protocols that require high computation and memory. Another constraint is that networked robots can only access to the information they have accumulated, so they have a good performance in a well known environment but their performance degrades when they work in a new and different environment.
To address these challenges, researchers have proposed the *Cloud Robotics* technology which utilizes the on-demand resources provided by an ubiquitous cloud infrastructure taking advantage of its computation capabilities for parallel computation and data sharing. Cloud Robotics consists of two main components: the cloud infrastructure and the robots. The cloud infrastructure consists of high-performance servers and massive databases while the robots can be of various types, from automated machinery to UAVs (unmanned aerial vehicles). The transition to the Cloud Robotics approach has been made possible thanks to the improvements in communication technology and the increasing availability of network connections, overcoming the constraints imposed by the limited on-board processing, storage and battery capacities.

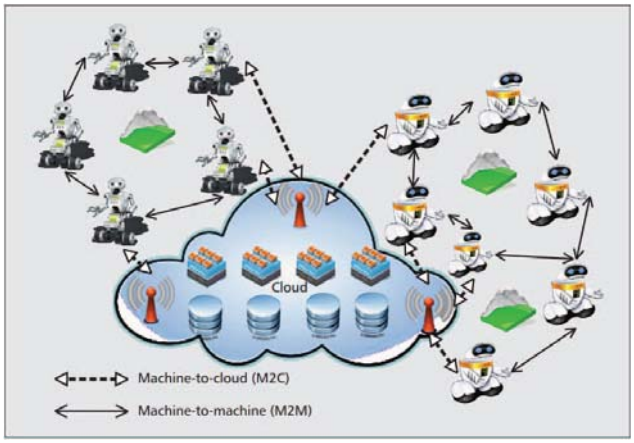

Fig. 1: Cloud Robotics: robots are interconnected via M2M/M2C communications, sharing their resources and accessing to remote cloud resources [2].

We can imagine a near future where robots will be very present in our lives: in our homes to help us with our domestic chores, in our hospitals to assist the patients, in our streets with the employment of self driving cars. Obviously small devices are not equipped with powerful resources so it is necessary to offload resource intensive tasks to external computers, e.g. the Cloud. However, in the scenarios we have described before, robots will often interact with humans so much stricter requirements for safety are needed, in order not to collide with or hurt people. Depending completely on the Cloud means that our robots are not able to take decisions and that a smaller or larger latency is introduced in every operation. Furthermore a robot will not be able to react to a situation if the network

*Giovanni Stanco is a PhD Student at the University of Napoli Federico II. His work is supported through a grant by Rislab SRL, Italy.

978-1-7281-1035-6/20/$31.00 ©2020 IEEE
DOI 10.1109/MobileCloud48802.2020.00010

IEEE computer society

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

connection is temporarily unavailable, putting the safety of the human user at risk. These obstacles can prevent the robot from respecting the safety requirements, which basically means that not all the computation can be moved to the Cloud. We can then affirm that for future robotics application it will be fundamental to have some tasks executed locally in order to safeguard human users and other tasks offloaded to more powerful devices.

In this paper we first of all describe some elements of the state of the art in Cloud Robotics, including applications, software platforms and architectures. One of the most interesting architectures proposed in literature for future robotics applications is *Dew Robotics*, which allows to develop solutions that do not rely completely on the Cloud but whose computational capabilities are distributed among different devices. Following the Dew Robotics architecture, we propose DewROS, a platform that uses ROS, one of the software platforms available for robotics. DewROS exploits the intelligence of edge devices to change the operating conditions of our applications and adapt them to the system's status. We then focus our attention on one of the possible use cases of DewROS, which is the surveillance scenario, and describe how a DewROS software platform that performs video analysis works. The last part of this paper is dedicated to the deployment of the aforementioned DewROS software platform on two machines and to the tests carried out to validate the efficiency of our solution and to evaluate the impact of the software on our machines.

II. Related works

A. Robotics applications

Image processing is used to recognize objects and scenes in a large number of applications, but it is hard to imagine to have a high-portability device with the specific requirements for image processing. Bhargava et al. [3] proposed an architecture for a mobile-cloud pedestrian crossing guide for blind people based on a mobile device with integrated GPS receiver and a Cloud server. The mobile device is used for the most critical tasks such as local navigation while the cloud server analyzes the pictures captured by a camera module integrated into sunglasses to detect the presence and status of pedestrian signals. The server sends an appropriate response back and a speech feedback notifies the pedestrian whether it is safe to cross an intersection.

SLAM (Simultaneous Localization And Mapping) is the problem of constructing and updating a map of an unknown environment by a mobile robot while simultaneously navigating the environment using the map. SLAM can be used for both 2D and 3D motion and its main steps are: landmark extraction, data association, state estimation, state update and landmark update. The processing resources required to carry out SLAM in real time can be quite high. Ayush et al. [4] presented a system that used a private cloud infrastructure so that was possible to deploy different resources assignment strategies in order to satisfy the needs of the vision, odometry and mapping operations.

Manipulation is an important task in robotics and has wide applications, from domestic assistance to robot-assisted surgery. If the object to grasp is not precisely known, the problem becomes a challenge and some works in literature have introduced a cloud infrastructure to provide a solution. Kehoe et al. [5] have proposed a cloud-based grasp planning algorithm that takes as input an approximate object contour along with Gaussian uncertainity around each vertex. In a later work they illustrated how cloud based data and computation can improve 3D grasping tasks.

Cloud Robotics has furthermore been introduced in **smart homes** to overcome the limitations of hardware existing in simple service robots used in this scenario. Cloud computing improved these systems by offering it advanced sensing and interaction capabilities based on image processing and voice recognition [6].

Another field of application for Cloud robotics is **healthcare**. Cloud robotics can contribute to the advancement of a new model in this area, promoting a dedicated assistance to the patients. A proposed application [7] is the construction of a rehabilitation database which can be used by physical therapy robots to record and save information and that can help physical therapists to gather information about the health conditions and progresses of patients. The rehabilitation system consists of physical therapy robots and a rehabilitation server connected through a network. The server maintains a database of movement therapy and can also extract knowledge based on statistical processing.

Robotics and automation systems can serve a critical role in **disaster management** as they can perform unmanned search and rescue operation in areas which are dangerous and difficult for humans to access. Jangid et al. [8] explained in detail the utility and tremendous benefits that can be offered by a Real-Time Cloud (RTC) for efficient disaster management. RTC can help intelligent robots perform complex processing via a request and response model.

B. Software platforms

An important component for the development of distributed robotics applications is the use of established software platforms. Steps in this direction have been taken and open sources Cloud Robotics platforms such as Rapyuta and ROS have been proposed in literature.

Rapyuta is an open source platform that helps robots to offload heavy computation providing computing environments in the cloud. Each robot connected to Rapyuta has one or more secured computing environments giving them the ability to move their heavy computation onto the cloud. In addition, the computing environments are tightly interconnected with each other, giving the opportunity to work in team. With Rapyuta robots can authenticate on the platform, creating one or more computing environments in the Cloud, and launch the desired processes. Computing environments are private, secure, optimized for data transmission, and can be connected to build parallel architectures [9].

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

ROS is an open-source, meta-operating system for robots [10]. ROS provides the typical services offered by an operating system (for example hardware abstraction, message-passing between processes etc.) and it also provides tools and libraries for software development and execution. ROS is an excellent solution for the robotics applications that need distributed computation [11] and that rely on software that runs across several different computers. One of the basic goals of ROS is to design software as a collection of small independent programs called *nodes*, processes that perform computation and communicate with one another. Communication between nodes is possible thanks to the different paradigms of communication provided by ROS:

- synchronous communication over services for implementing a request/reply communication;
- asynchronous streaming of data over topics, buses over which nodes exchange messages, intended for unidirectional communication and supporting publish/subscribe semantics;
- storage of data on a Parameter Server, a shared dictionary used to store and retrieve parameters at runtime.

C. Cloud Robotics system architectures

Nowadays one of the main research activities about Cloud Robotics concerns Cloud Robotics system architectures (communications protocols between robots and the cloud). Hu et al. [2] proposed an architecture organized into two complementary tiers: a machine-to-machine (M2M) level and a machine-to-cloud (M2C) level. On the M2M level robots communicate on wireless links to form a collaborative ad-hoc cloud, while on the M2C level, the cloud provides shared computation and storage resources and the robots offload computation-intensive tasks for remote execution. Robots can communicate if they are within communication range of each other and can reach the cloud servers if they are close to the access point of the cloud infrastructure. A wireless M2M communication network can be formed by robots working cooperatively to route information and is often formed in an ad-hoc manner. In a practical situation, robots can leave and join the network or become unavailable because of unpredictable failures or obstructions in the environment and the routing protocols used must take in consideration these aspects. Traditional network routing protocols could represent a heavy and unjustified load on the network nodes in this case, therefore it is often worth adopting easier methods like gossip protocols that do not require route discoveries and maintenance and are thus suited for highly dynamic mobile robotic networks. These protocols are also very simple to implement, and require minimal additional computation and memory resources. However, the trade-off is that gossiping may result in a high message latency.

As seen before, a common problem for future applications is that they can not be fully based on Cloud computing. This issue has been addressed in literature as well, mainly proposing scarcely reusable application-specific solutions and not a generic architecture. But for future robotics applications, a new approach is necessary and a common architecture is needed. In literature two new approaches that satisfy the needs of future applications have been proposed and they are the Fog and the Dew robotics [12].

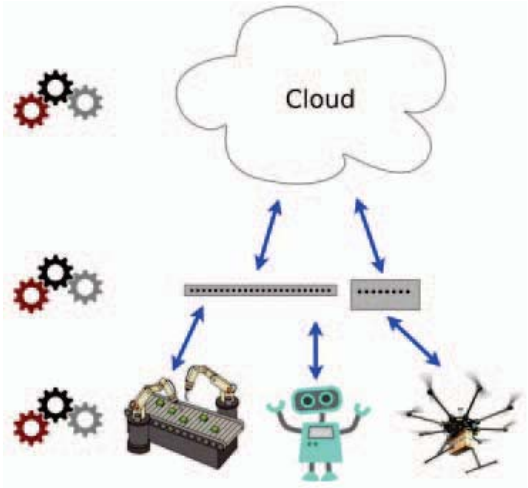

Fig. 2: Dew Robotics: the computational tasks are distributed between the dew, the fog and the cloud [12].

The Fog computing paradigm is a distributed computing paradigm that provides data and services closer to end users. The Fog computing paradigm deploys the decision-making processes that require more reliability closer to the robots. The vicinity between the computing nodes and the end user applications lowers network latency. The Dew computing paradigm extends the concept of resources distribution and it is based on the concept of microservices provided by the end user devices, without the help of centralized nodes. The main goal of Dew Computing is to improve scalability. The processing tasks are distributed over a large number of devices without the use of central nodes and the on-site devices are always able to provide a set of functionalities without Internet connection. The Dew paradigm can coexist with the Cloud and the Fog paradigms. The on-site devices are able to collaborate with central nodes when the Internet connection is available but they are not dependent on them. In this case the computation and storage can be split in three parts: locally on the robots, on the Fog nodes and on the Cloud. The most critical tasks are kept locally so that the robot can always react properly, the least critical are moved on the Fog and on the Cloud to exploit their computing, storage and power resources.

III. DewROS

DewROS is a platform for Dew Robotics that uses ROS. Its architecture is composed of different nodes interacting with each other and distributed over different network hosts. These nodes can communicate with each other through wired or wireless network technology, depending on the scenario.

In a DewROS architecture we have two main groups of nodes. The first group is made up of the nodes that actually execute the needed operations to fulfill our final goal and they

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

are deployed on the machine that best satisfies their resources' needs. These nodes in fact carry out different tasks, which may or may not need powerful resources and that can be more or less critical. Typically the dew nodes have simple hardware and limited power supply, so they host the less computation-demanding operations, while the fog hosts have more computational, storage and power resources, appropriate for the most demanding tasks.

The second group is composed of different ROS nodes that monitor performance parameters regarding the hosts they run on [13]–[15] and interact with the nodes belonging to same group or the other one. The parameters monitored by these nodes can be various, e.g. CPU and memory usage, battery percentage, network parameters. All this information collected is then shared with, e.g. the first group of nodes in order to change their operating conditions and to optimise their operations. The communication among the nodes is fully in line with the ROS communications paradigm.

The DewROS platform works as follows: the main nodes start their operations on the dew or fog machines, in the meantime the monitoring nodes, which run on the dew and the fog machines, begin their work and periodically inform the other nodes, sending messages containing the values measured via ROS topics (figure 3). Thanks to the information received, the first nodes can make changes about their operating conditions and adapt to the system's status. For example, we can implement a node that monitors the CPU usage and the battery level of the robot and another one that uses this information to reduce the number of running processes when the CPU is overloaded or to decide where to perform a task according to the energy requirement and availability on the robot.

We believe that DewROS will be more and more important in the future as it provides the necessary support for several operations in a multi-layer Cloud robotics scenario. DewROS is suitable for all the distributed robotic applications where devices with limited resources are employed and where the monitoring of performance parameters can bring advantages. In the following, we will focus our attention on a specific use case where video analysis is used for rescuing and surveillance.

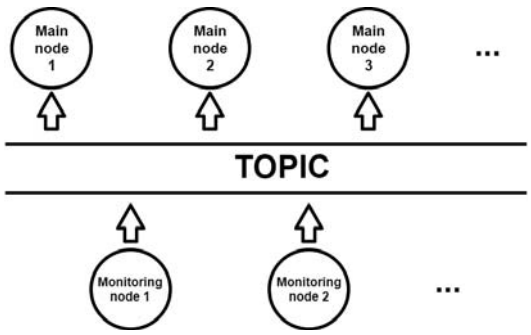

Fig. 3: Main and monitoring nodes in a DewROS solution.

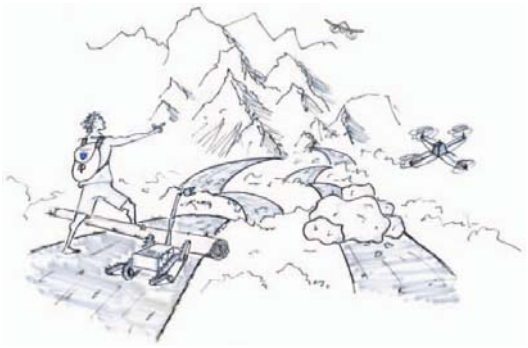

Fig. 4: A sketch of the SHERPA team.

IV. DewROS in SHERPA

A. The SHERPA project

SHERPA [16] is a European project whose research activities started in 2013. The real world scenario inspiring the SHERPA activities is the one of surveillance and rescuing in unfriendly and hostile environments like mountains or forests. The main goal of SHERPA is to develop a robotic platform supporting the rescuers in their activity in order to improve their ability while decreasing the costs and the risks. The adverse environmental conditions in which the platform operates ask for robust automatic control and communication capabilities of the robotic platform.

The activities of SHERPA are focused on a combined aerial and ground robotic platform to support human operators in surveillance and rescuing tasks in hostile environments, like the alpine scenario targeted by the project.

B. DewROS in SHERPA

One of the purposes of rescuing missions is to find survivors after a natural disaster in an alpine environment. The UAVs employed by SHERPA are therefore sent in the location of the calamity to capture videos and images in order to find dispersed people. The huge amount of information captured can not be easily analyzed by a human operator, who will easily become the bottleneck of the system performance. Exploiting the computational resources of a Cloud service can surely help. Capturing videos at high definition is necessary for good video analysis results. But such videos involve a large volume of byte, which cannot be easily transferred from the drone (the dew node) to the ground (a fog node), especially in alpine scenarios, where the network conditions are not optimal and largely variable with the drone moving around.

DewROS can provide several benefits in this scenario. It can monitor the network conditions and performance so as to maximize the video quality according to such conditions and to capture videos at the highest resolution possible. Our monitoring nodes continuously check the network conditions and allow the dew node to decide the best quality of the video captured to be transferred towards the Cloud for the analysis.

For this use case, we devised a solution made up by three main ROS nodes that execute the video analysis: a *reader*

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

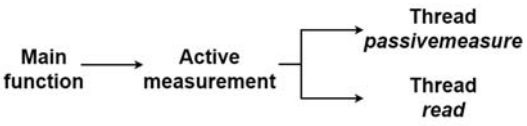

Fig. 5: Reader node diagram.

node that captures video frames from a camera, a *writer* node that receives the frames and writes them in a video file, and an *annotator* node that receives the video to analyze and sends an annotation request to a cloud service. The intelligence of the system is therefore divided among different hosts. The reader node is deployed on the *dew* host and the other two run on the *fog* host, while the most resource demanding task is performed by the cloud services.

These nodes use network monitoring tools to evaluate the network status. The monitoring nodes are launched by the reader node and influence its behaviour providing it with useful information. They are divided in two groups. The first group contains the *active* nodes, that evaluate the initial value of the available bitrate between the dew and the fog hosts before the frames capture begins. The second group contains the *passive* nodes, which use Linux tools to check the conditions of network interfaces and status of TCP connections over them. This second group of nodes works at the same time of the frames capture and provides the information that the reader node will use to change its operating conditions.

1) Reader: The reader node is responsible to start the active measurement, the passive measurement and the frames capture. The main function of the script initializes the node, starts the active measurement and then, when the active measurement is finished, launches two threads: one for the passive measurement and one for the frames capture.

The active measurement is done by three nodes that use D-ITG (Distributed Internet Traffic Generator) [17], a tool capable to measure the most common performance metrics (e.g. throughput, delay, jitter, packet loss) at packet level. The three D-ITG nodes work as follows:

- **receiver node**: on the fog host, it calls the D-ITG Receiver entity and starts listening;
- **sender node**: on the dew host, it calls the D-ITG Sender entity which sends packets to the receiver node;
- **decoder node**: on the fog host, it calls the D-ITG Decoder entity to analyze the log file stored during the experiment, retrieves the measured bitrate value and saves it on the ROS parameter server to make it accessible to the other nodes.

The initial values of the video resolution (i.e. the frame size) are decided on the basis of the measured bitrate. Once the active measurement is completed, the reader node starts two threads at the same time. One of the threads launches two *passive* nodes. The first passive measurement nodes uses *ifconfig* to retrieve the number of bytes sent by the dew host in a time interval in order to evaluate the outgoing bitrate on

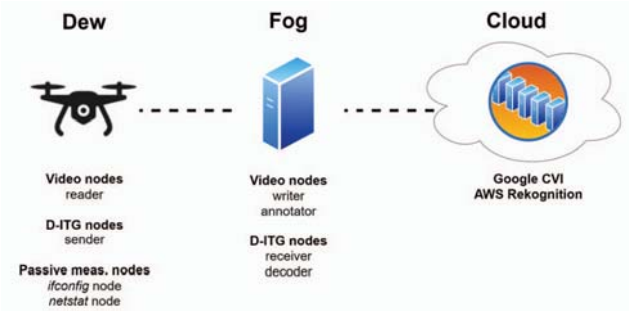

Fig. 6: Nodes for a DewROS platform in SHERPA.

the interface we used. The bitrate can be easily calculated as follows:

$$\frac{8 * (bytes1 - bytes0)}{timestamp1 - timestamp0} \quad (1)$$

According to our needs, this bitrate value can be saved in a text file or sent over a ROS topic, making it available for every node that may need it. The other passive node checks the size of the queues of the data that are waiting to be transmitted in order to decide whether the bitrate provided by the network is enough to transmit the video frames at the resolution its is using. In order to know the size of these queues, the node periodically calls *netstat*, a command-line utility that displays network connections and network protocol statistics. This size value gives us an idea about the bitrate availability and will be sent over a ROS topic and used by the reader node to increase or decrease the video resolution.

The second thread launched by the reader node is responsible for frame capture. For the frame capture we used OpenCV, an open source computer vision and machine learning software library. The reader node creates an OpenCV *VideoCapture* object, sets width and height according to the bitrate value determined by the active measurement and starts capturing. Every frame captured using *VideoCapture* is converted in a CompressedImage message using the JPEG compression and sent over a topic to the writer node. While the node is capturing video frames, it periodically checks if a new message from the second passive node is available. If a new message is not available the node will keep on capturing, otherwise it will check the received queue size value and will decide if it is necessary to change the video resolution using a simple adaptive algorithm we implemented.

2) Writer: The writer node runs on the fog host. After its initialization, the node subscribes to the topic over which the reader node sends the frames in a *CompressedImage* message. Every time a new frame is received, the callback function converts the CompressedImage in an OpenCV image and accesses its width and its height: we need to check the dimensions of every frame received to decide whether it is possible to write the new frame on the current video chunk or if it necessary to open a new one.

3) Annotator: The annotator node is on the fog host and works with videos saved locally. We implemented two dif-

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

ferent annotator nodes that use two different Cloud services: one for Google Cloud Video Intelligence and one for Amazon Web Services Rekognition. The common part of these two nodes is that they subscribe to a ROS topic over which the writer node will transmit the name of the video file to upload and analyze. The Google annotator can communicate with the Google servers thanks to the *videointelligence* library provided by Google, while in the Amazon annotator we need to import the *boto3* library to exploit the AWS functionalities. The result of the annotation process is a text file containing the labels, that is the name of the objects identified in the video, the percentage confidence in the accuracy of the detected label and the time the object was detected in the video.

V. Experimentations

A. Using the measurement nodes

To evaluate our solution we tested DewROS in controlled and uncontrolled conditions, using a wired connection in the former case and a wireless one in the latter one. In both cases the dew host was a Raspberry Pi, while the fog one was an Ubuntu virtual machine running on a Windows PC.

1) Testing in controlled conditions: For our testing activities in controlled conditions we used an Ethernet connection and tool that limits the available bitrate on our network in order to test our adaptive algorithm. The tools we used are *tc* (traffic control) and *netem* (network emulator). *Tc* is used to configure traffic control settings in the Linux kernel and allows us to add a queueing discipline to our selected interface; *netem* is an enhancement of the Linux traffic control facilities that allows to add delay, packet loss and more other characteristics to packets outgoing from a selected network interface. Specifically we use *netem*'s *rate* option to set the maximum bitrate outgoing from our selected interface, as shown in Figure 7.

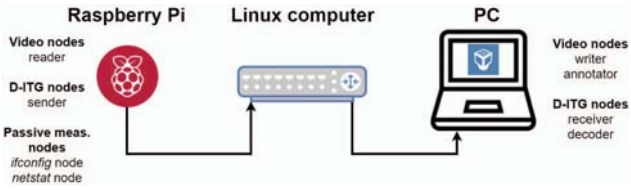

Fig. 7: Devices used for the testing activities.

We used *tc* on the Linux computer, specifically on the interface that links the Linux computer to our PC, and periodically changed the maximum outgoing bitrate, randomly choosing its value from a set of seven values ranging from 0.5 to 32 Mbps. We also tested our system changing two parameters: the first parameter is the time period used in the script that sets a *traffic control* constraint, the second is the time period used in the *netstat* node to communicate the aggregate value of the queues. Fig. 8 shows the results of a test lasting about 40 minutes, with a *traffic control* period of 180 seconds and a *netstat* period of 5 seconds.

The blue line in figure 8 shows the values of bitrate imposed by *tc* and *netem*, the red dots show the bitrate measured by the *ifconfig* node on our Raspberry Pi. We can see how the outgoing bitrate follows the restrictions imposed by *netem*.

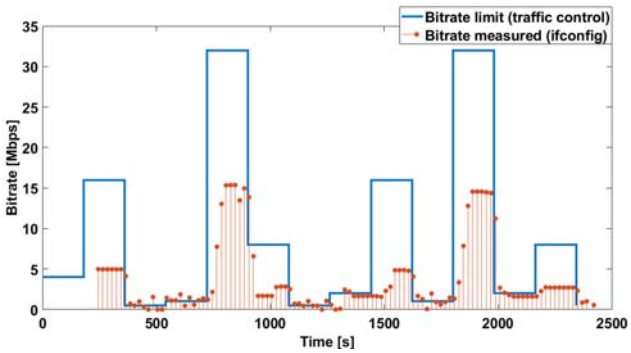

Fig. 8: Bitrate values.

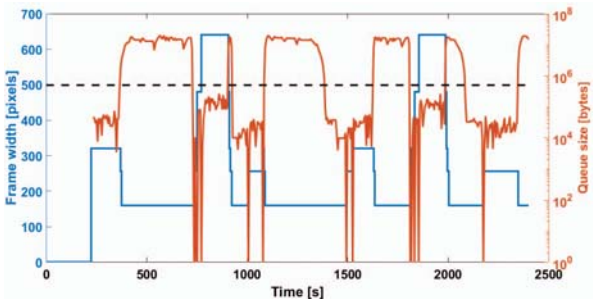

Fig. 9: Frame width and queue sizes.

Fig. 9 shows how the video resolution changes according to the queue size. The blue line is the width of the frames, the red line shows the aggregate values of the queues: these are the values received by the reader node from the node using *netstat* and are plotted using a logarithmic scale for the y-axis. The dashed black line is the threshold above which the video resolution is decreased. We can easily see that every time there is a large queue the frame's width decreases while when we do not have a queue the width increases.

If we compare the two graphs we can notice a trend: when the available bitrate is high there are no queues, so the adaptive algorithm can use a higher video resolution; when the available bitrate drops the queues' sizes increase so the reader uses a lower resolution. We carried out several experiments using three different values of *traffic control* periods and three values of *netstat* periods. We changed the traffic constraint every 30, 60 or 180 seconds while the three periods we used with *netstat* are 2, 5 or 10 seconds. For every combination of these values we performed three experiments and calculated the average error in Mbps between the available bitrate and the produced bitrate. In table I we reported the error values for every combination of the two time periods.

For the *netstat* periods of 2 and 5 seconds we observe the expected trend: the average difference is minimum when the channel changes slowly (that is when the traffic control constraint changes every 180 seconds), while it is maximum

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

	30 s	60 s	180 s
2 s	5.714	5.499	3.155
5 s	8.44	6.595	5.019
10 s	6.961	6.612	8.564

TABLE I: Average error in Mbps between the available and used bitrate.

when the channel changes rapidly. In particular for the 2 seconds period we have lower differences since our passive node is more sensible to the channel's variations so that our system can follow the traffic control constraints better than it does when the *netstat* period is 5 seconds. This expected trend is not respected for the 10 seconds period: the 10 seconds interval between the notification of two queues values is in fact too high for our application and it does not allow to follow the variability of the channel and to exploit the maximum available bandwidth.

2) Testing in uncontrolled conditions: We also tested our solution using Wi-Fi as well. Obviously it was not possible to use *tc* and *netem* in this case, but we had to move our Raspberry Pi around an access point in order to have a variable channel. In our tests our Windows PC, on which the virtual machine is running, works as an access point, while the Raspberry Pi is mounted on a moving device. The device we used is a Smart Video Car Kit for Raspberry Pi produced by the company Sunfounder [18].

Fig. 10: Sunfounder Smart Video Car Kit for Raspberry Pi.

The first experiments we carried out were inside our laboratory. Fig. 11 shows is a map of the laboratory and the trajectory followed by the Smart Video Car. This experiment was repeated three times but we report only the results of the first repetition, since the three results are similar.

The first plot of Fig. 12 shows the bitrate measured by the *ifconfig* node every ten seconds. The frame widths are reported in the second plot. The queue sizes measured every two seconds are reported in the third plot. As we can see, our system works as expected: the reader decreases the video resolution when the queue size exceeds the threshold, while the video resolution is increased when the queue size is null.

We then carried out a second group of experiments, driving our Smart Video Car outside our laboratory, introducing a brick wall obstacle between the Raspberry Pi and the access

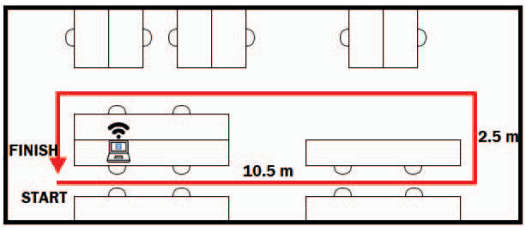

Fig. 11: Path inside our laboratory.

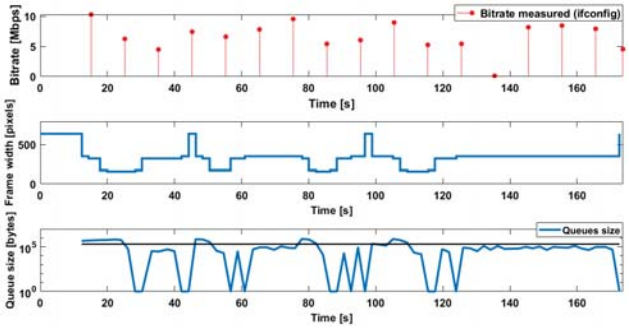

Fig. 12: Results of the experiment inside the laboratory.

point provided by our Windows PC. The path that our car followed is reported in Fig. 13.

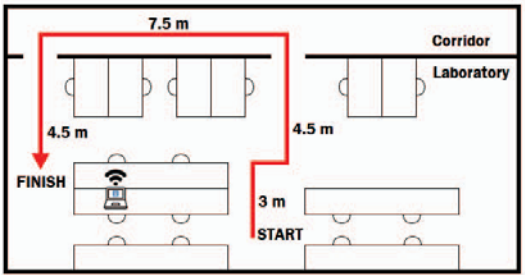

Fig. 13: Path in the laboratory and in the corridor

In this case we used the same interval of two seconds for both of our passive nodes in order to have more information in our graphs. In the first plot of Fig. 14 we also reported the bitrate values retrieved from *iwconfig*The Wi-Fi NIC of our Raspberry supports multiple bit rates, so we can interpret the red dots of the first plot as the nominal available bitrate. In Fig. 14 we can see a sudden drop of the bitrate reported by *iwconfig* before the 120th second of the experiment. These moments correspond at the exit of the car from the laboratory and the beginning of the straight path in the corridor. Even though this seems to be the section where the network conditions are the worst, our platform was still able to capture frames at the lowest resolution possible because it had previously lowered the video quality. Actually, we can see that on the 120th second the queue was empty, so it was possible to increase the video resolution, even if the available bitrate was limited.

As we can see from Fig. 12 and Fig. 14, the frames capture

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.

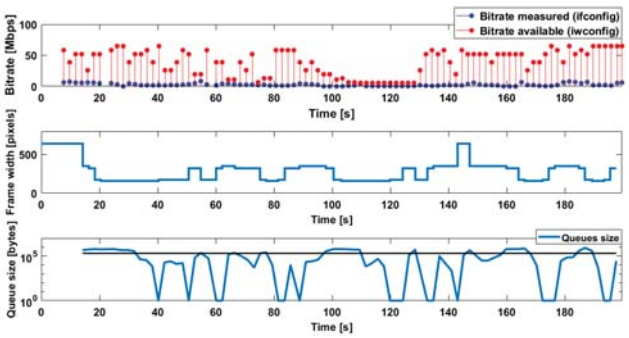

Fig. 14: Results of the experiment in the laboratory and in the corridor.

starts with the 640x480 resolution, which is decided after the active measurement, that in our experiments always returned a bitrate of 30-35 Mbps. Despite having this available bitrate and using only around 5-10 Mbps to send the video frames, we observe large queues at the beginning of our experiment and the video resolution is quickly reduced. The large number of bytes not acknowledged by the remote host may be caused by the ACK mechanisms in IEEE 802.11.

VI. Conclusion

In this paper we have discussed one of the main problems regarding future robotics applications. In particular we highlighted how robots will be pervasive in everyday life and how robots will need computational power to respond in safety-critical tasks.

We then proposed the DewROS platform, that could overcome the obstacles of a Cloud robotics solution, moving part of the computational resources closer to the end user. We also described one of the possible scenarios where such a platform could be used, analysing how to implement a DewROS solution in the case of a rescuing mission. The main distinguishing characteristic of DewROS is the provisioning of a few monitoring nodes that continuously measure the performance of the network (and possibly of other interesting parameters in the future) and provide such important information to the other ROS nodes. Having such vital information, the other nodes can better perform their tasks and fully exploit the available resources, typically limited, at the robot side.

We also tested DewROS both in controlled and uncontrolled network conditions, to see if everything was working properly and to check the befits achievable.Our results show that the introduction of DewROS in a real working environment can actually improve the performance of ROS nodes and allow to fully exploit the available, usually limited, resources.

Our ongoing work is focused in implementing more monitoring nodes in DewROS and testing it in other use cases, also using real drones.

Acknowledgment

This work was partially supported by MIUR through the "ICT for Health" project, Dipartimento di Eccellenza (2018-2022) "Ingegneria Elettrica e delle Tecnologie dell'Informazione" and by Cisco Systems through the Sponsored Research Agreement "Research Project for Industry 4.0".

References

[1] O. Saha and R. Dasgupta, "A comprehensive survey of recent trends in cloud robotics architectures and applications," *Robotics*, vol. 7, 08 2018.

[2] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, May 2012.

[3] P. Angin, B. Bhargava, and S. Helal, "A mobile-cloud collaborative traffic lights detector for blind navigation," 01 2010, pp. 396–401.

[4] K. Ayush and N. K. Agarwal, "Real time visual slam using cloud computing," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 2013, pp. 1–7.

[5] B. Kehoe, D. Berenson, and K. Goldberg, "Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 576–583.

[6] J. Pinta, J. Maestre, I. Jurado, and S. Reyes de Cózar, "Off the shelf cloud robotics for the smart home: Empowering a wireless robot through cloud computing," *Sensors*, vol. 17, 03 2017.

[7] T. Yokoo, M. Yamada, S. Sakaino, S. Abe, and T. Tsuji, "Development of a physical therapy robot for rehabilitation databases," in *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, March 2012, pp. 1–6.

[8] N. Jangid and B. Sharma, "Cloud computing and robotics for disaster management," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, Jan 2016, pp. 20–24.

[9] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, April 2015.

[10] "Ros - introduction," Available at http://wiki.ros.org/ROS/Introduction (2018).

[11] J. M. O'Kane, *A Gentle Introduction to ROS*. Independently published, 10 2013, available at http://www.cse.sc.edu/~jokane/agitr/.

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

[13] V. Persico, A. Botta, P. Marchetta, A. Montieri, and A. Pescapé, "On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe," *Computer Networks*, vol. 112, pp. 67 – 83, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861630353X

[14] G. Aceto, A. Botta, A. Pescape, and C. Westphal, "Efficient storage and processing of high-volume network monitoring data," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 162–175, June 2013.

[15] A. Botta, W. de Donato, A. Pescape, and G. Ventre, "Discovering topologies at router level: Part ii," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, Nov 2007, pp. 2696–2701.

[16] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis, "The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov 2012, pp. 1–4.

[17] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[18] "Smart video car kit for raspberry pi," Available at https://www.sunfounder.com/learn/category/Smart-Video-Car-for-Raspberry-Pi.html.

Authorized licensed use limited to: Universita degli Studi di Napoli Federico II. Downloaded on August 31,2021 at 13:24:42 UTC from IEEE Xplore. Restrictions apply.