Antevski, K., Groshev, M., Cominardi, L., Bernardos, C.J., Mourad, A. y Gazda, R. (2018). Enhancing Edge robotics through the use of context information (2018). In: *Workshop on Experimentation and Measurements in 5G (EM-5G'18)*. ACM, 2018, pp. 7-12.

DOI: https://www.doi.org/10.1145/3286680.3286682

Enhancing Edge robotics through the use of context information

K. Antevski
University Carlos III of Madrid

M. Groshev
University Carlos III of Madrid

L. Cominardi
University Carlos III of Madrid

C.J. Bernardos
University Carlos III of Madrid

A. Mourad
InterDigital Europe Ltd.

R. Gazda
InterDigital Communications Inc.

ABSTRACT

Cloud robotics aims at endowing robot systems with powerful capabilities by leveraging the computing resources available in the Cloud. To that end, the Cloud infrastructure consolidates services and information among the robots, enabling a degree of centralization which has the potential to improve operations. Despite being very promising, Cloud robotics presents two critical issues: (*i*) it is very hard to control the network between the robots and the Cloud (e.g., long delays, high jitter), and (*ii*) local context information (e.g., on the access network) is not available in the Cloud. This makes hard to achieve deterministic performance for robotics applications. Over the last few years, Edge computing has emerged as a trend to provide services and computing capabilities directly in the access network. This is so because of the additional benefits enabled by Edge computing: (*i*) it is easier to control the network end-to-end, and (*ii*) local context information (e.g., about the wireless channel) can be made available for use by applications.

The goal of this paper is to showcase, by means of real-life experimentation, the benefits of residing at the Edge for robotics applications, due to the possibility of consuming context information locally available. In our experimentation, an application running in the Edge controls over a Wi-Fi link the movement of a robot. Information related to the wireless channel is made available via a service at the Edge, which is then consumed by the application.Results show that a smoother driving of the robot can be achieved when wireless quality information is considered as input of the movement control algorithm.

KEYWORDS

5G, Edge, MEC, Robotics, Experiment, Evaluation, Testbed

ACM Reference Format:
K. Antevski, M. Groshev, L. Cominardi, C.J. Bernardos, A. Mourad, and R. Gazda. 2018. Enhancing Edge robotics through the use of context information. In *Proceedings of ACM CoNEXT '18.* ACM, New York, NY, USA, 6 pages. https://doi.org/TBA

Corresponding author: K. Antevski. Email: kiril.antevski@uc3m.es
This article has been partially supported by the EU H2020 5G-CORAL Project (grant no. 761586) and by the 5G-City project (grant no. TEC2016-76795-C6-3-R) funded by the Spanish Ministry of Economy and Competitiveness.

ACM CoNEXT '18, 4–7 Dec., 2018, Crete, Greece

ACM ISBN TBA…$TBA
https://doi.org/TBA

1 INTRODUCTION

Cloud Robotics leverages and integrates Cloud computing, Cloud storage, and other Internet technologies, into industrial and commercial robotics applications. Cloud technologies enable robot systems to be endowed with powerful capability by leveraging the powerful computation, storage, and communication resources available in the Cloud. Consequently, it is possible to build lightweight, low cost, and smarter robots by placing an intelligent brain in the Cloud which offers a converged infrastructure that can be also used to share services and information from various robots or agents. To that end, Cloud infrastructure for robots shall support the sharing of data between various robots and agents connected to the Cloud, such as images, maps, robot outcomes, trajectories, and control policies [11].

Although robots can benefit from various advantages of Cloud computing, this presents several limitations when applied to the Cloud Robotics field [2]. Cloud facilities traditionally reside far away from the robots and while the Cloud providers can ensure certain performance in their infrastructure, very little can be ensured in the network between the robots and the Cloud, especially when multiple Internet providers are involved. As a result, Cloud-based applications can suffer from high-latency or unpredictable jitter in the network. This is exacerbated for applications relying on real-time data from the robot and the surrounding environment (e.g., Automatic Guided Vehicles). Given the challenges of assuring the network performance at infrastructure level, the applications are hence required to adapt their operations depending on the network conditions. However, accessing information related to the network (e.g., on the radio channel) is equally challenging when multiple domains are involved. Moreover, network operators are not allowed to publish such sensitive data on the Cloud for regulatory and privacy reasons.

Over the last few years, Edge computing has arisen as a promising paradigm in the telecommunication industry in response of the ever increasing traffic demands and stringent requirements expected in forthcoming 5G networks [1]. The Edge computing vision foresees the deployment of computing capabilities directly in the operator's access network, which would enable the provisioning of applications and network services closer to the users compared to the traditional Cloud computing. As a result, operators can offer low latency services to the users whilst simultaneously offloading their core network. Moreover, Edge computing aims at exploiting the context information available locally in the access by making it available to the applications through services. By doing so, applications can subscribe to those services and consume the context information to optimize their functionalities.

Driven by these needs and opportunities, ETSI created the Multi-access Edge Computing (MEC) Industry Specification Group (ISG)

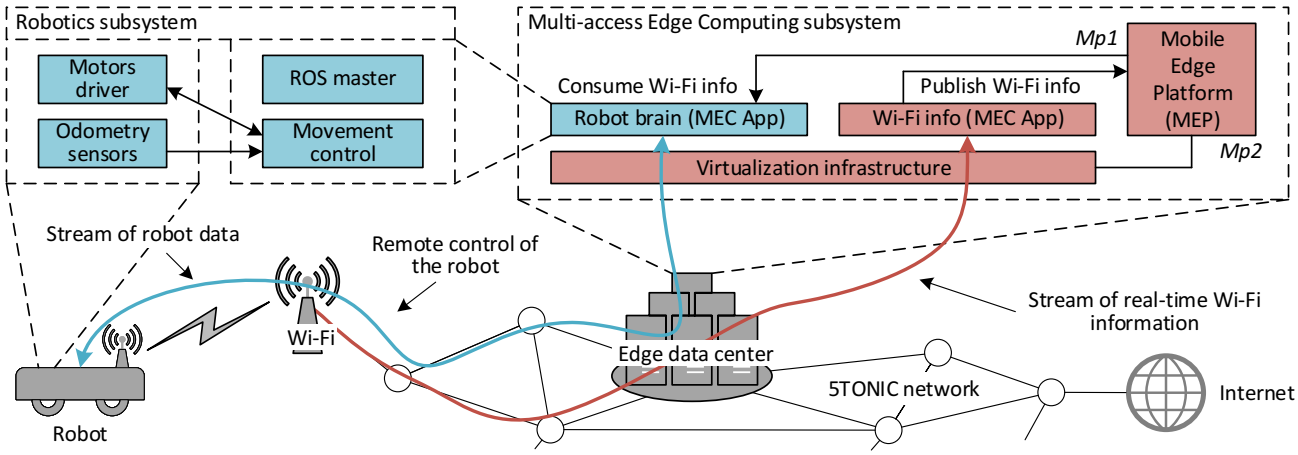

Figure 1: Edge robotics system used for experimentation

with the goal of standardizing the Edge computing ecosystem. Such ecosystem aims at achieving convergence of IT and telecommunications networking to enable new vertical business segments and services for consumers and enterprise customers. The evolution of Cloud robotics towards Edge robotics lies among these services. By placing the brain of the robots in the Edge rather than in the Cloud, it is hence possible (*i*) to ensure low latency between the robots and their brains due to the shorter distance, and (*ii*) to consume context information on the access network in order to adapt the robotics operation to the context, including the communication links status. It is worth highlighting that in case of wireless access, network performance can only be ensured within certain limits and transmission failures are still likely to happen. Consequently, applications can benefit from the context information about the network to adapt to such cases.

This paper aims at showcasing the benefits for robotics applications of adapting their operations to context information available locally at the Edge. To that end, a real-life experimentation is performed in a small-scale environment where the movement of one remotely-controlled mobile robot is adapted in accordance with the wireless information available at the Edge. The remainder of the paper is organized as follows. Sec. 2 presents the Edge robotics system under test, which is characterized next in Sec. 3. Sec. 4 proposes a MEC-enabled control algorithm for the robot, which is then evaluated experimentally in Sec. 5. Finally, Sec. 6 presents the lessons learned and draws the conclusions.

2 EDGE ROBOTICS SYSTEM OVERVIEW

This section presents the Edge robotics system used for experimentation. Fig. 1 shows the overall logic system which comprises two separate but interacting subsystems: (*i*) the robotic system (shown in blue), and (*ii*) the Multi-access Edge Computing system (shown in red). These are detailed in Sec. 2.1 and Sec. 2.2, respectively.

2.1 Robotics subsystem

Today's robotics systems require the deployment of dedicated robotics hardware and software along with access to Cloud infrastructure. As mentioned in [11], the robots maintain their independent operating capabilities and rely on the Cloud for accomplishing complex tasks, such as big data analytics, collective learning, crowd-sourcing, etc. Following these principles, [14] proposes a Cloud-based framework wherein industrial robots are remotely configured so as to enable an ubiquitous manufacturing environment. An example of Cloud-based industrial manufacturing is presented in [12], where the planning of the robotics tasks is distributed and executed across a high-speed wide-area network. In [9], the proposed simultaneous localization and mapping solution uses the Cloud infrastructure to offload the heavy computational tasks and large data sets from the robots. In [13], the Cloud infrastructure is used by the robotics system to consume context information (e.g., cognitive Industrial Internet of Things) as a mean to improve the production efficiency. Finally, [10] proposes a distributed cooperative communication and link prediction framework to cope with the network issues in Cloud Robotics. However, such framework requires pre-knowledge of the link quality in the case of robot mobility.

A streamlined provisioning of robotics software components is seen as a necessity to cope with the rapidly emerging of new robotic services. To that end, new open-source platforms are arising to simplify the software development for different robotic hardware. The most widespread framework nowadays is Robot Operating System (ROS)[1], which provides a meta-operating environment for developing and testing multi-vendor robotics software. In ROS, each software component is called ROS node. Moreover, ROS provides a publish-subscribe messaging framework via a specific node, namely ROS master. By connecting to the ROS master, ROS nodes can register and locate each other. Once registered, nodes can exchange data via configurable topics in a peer-to-peer fashion.

In our set-up (see Fig. 1), the robotics subsystem is implemented as various ROS components distributed across the robot itself and the Edge data center. The robot is equipped with motored-wheels and odometry sensors[2] (e.g., motor encoders). The ROS components running on the robots are essentially drivers that are in charge of (*i*) reading data from the sensors (e.g., odometry) and send them to the brain, and (*ii*) executing the driving instructions received from the brain. The robot brain acts as a ROS master and it is also in charge of driving the robot based on the available information.

[1] http://www.ros.org/
[2] Odometry is the use of data from motion sensors to estimate changes in position over time.

The communication between the robot and the brain crosses over a Wi-Fi link and the wired network connecting to the Edge data center. In accordance with the Edge computing concept, a wireless information service is available locally at the Edge data center. This is consumed by the ROS node controlling the movement to adapt the robot driving. The details regarding the wireless information service are reported in the following paragraphs.

2.2 Multi-access Edge Computing subsystem

As described in [4], Multi-access Edge Computing enables the implementation of mobile edge applications as software-only entities that run on top of a virtualization infrastructure, which is located at or close to the network edge. One realization of these applications is the robot brain described above. The main MEC component is the Edge data center, which acts as *mobile edge host* and it consists of the following entities: (*i*) a Virtualization infrastructure, (*ii*) a Mobile Edge Platform (MEP), and (*iii*) Mobile Edge Applications (MEC Apps). The virtualization infrastructure provides compute, storage, and network resources for the MEC applications. The virtualization infrastructure includes a data plane for routing the traffic among applications, services, local/external networks, and mobile edge platform. The configuration of the data plane is done via the Mp2 reference point. The mobile edge platform offers an environment where the MEC applications can discover, advertise, consume and offer mobile edge services. Finally, MEC applications run on top of the virtualization infrastructure provided by the mobile edge host, and can interact with the mobile edge platform to consume and publish mobile edge services via the Mp1 reference point. How these MEC applications retrieve the data to be published as a service is left unspecified by current ETSI MEC specifications.

ETSI MEC defines a set of exemplary services. For example, the Radio Network Information service (RNIS) [7] provides radio network-related information, such as up-to-date radio network conditions, measurement and statistics information related to the user plane, and information related to users served by the radio nodes. While the original RNIS service was designed for 3GPP networks, a new service is being defined to cover also Wi-Fi networks [8]. Another example is given by the location service [6] which provides location-related information about the users (e.g., all of them or a subset) currently served by the radio nodes. The location information can be geolocation, Cell ID, etc. Finally, the Bandwidth Manager service [5] allows the allocation of bandwidth to certain traffic routed to and from MEC applications and the prioritization of certain traffic. Additional services can be then defined upon necessity based on the same MEC framework.

While the MEC framework can serve multiple access technologies (e.g., 4G, 5G, Wi-Fi, etc.), for the sake of our experimentation we focus on the Wi-Fi access. To that end, we developed a MEC service providing Wi-Fi information regarding the clients connected to the system. This is achieved by creating a MEC application (shown in red as Wi-Fi Info (MEC App) in Fig. 1) that gathers Wi-Fi information from the radio nodes and exposes it via the mobile edge platform to other MEC applications (e.g., the Robot brain (MEC App) in Fig. 1). The Wi-Fi network information service hence provides for each connected client (e.g., the robot) data on the signal level, transmission and reception bit rates, number of

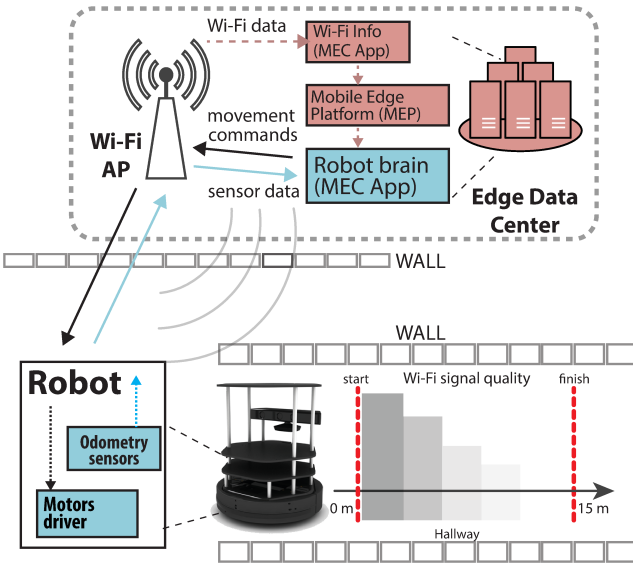

Figure 2: Experimental scenario

retransmission and packet losses at data link level, and number of successfully transmitted/received bytes and packets. Moreover, link layer configuration is also provided: wireless channel, beacon interval, preamble and slot time (i.e., short/long), QoS support and authorization/authentication status. The information is then published in JSON format and can be accessed by MEC applications (e.g., the robot brain) through HTTP requests.

3 EXPERIMENTAL METHODOLOGY

This section describes the experimental set-up and the experiments we performed to explore the benefits and performance of the Edge robotics system described in Sec. 2. Particularly, Sec. 3.1 describes the experiments performed, while Sec. 3.2 evaluates the relevant factors affecting the robot performance while being controlled from the Edge over a Wi-Fi link.

3.1 Experiments description

To evaluate the Edge robotics scenario, we have built an experimental environment in the 5TONIC [3] laboratory. In such environment, we have deployed all the components shown in Fig. 1. The goal of the experimental test-bed is to show how the Edge controlled robotics paradigm improves current Cloud robotics techniques towards the industry demands of high speed and high precision in robotics applications. To that end, we have designed the experiment shown in Fig. 2 and described in the following.

For the mobile robot, we used the ROS-compatible Kobuki[3] robotics platform. The mobile robot maximum speed is 0.75 m/s, while its minimum speed is 0.1 m/s. The sampling frequency for reading the odometry sensor data from the robot's wheels is 16.6 Hz (i.e., odometry sensor data is refreshed every 60 ms). When driving at full-speed (0.75 m/s), the robot covers a distance of 4.5 cm in 60 ms. This results in a precision of 4.5 cm in the robot driving at full-speed since odometry sensor data can not be updated with

[3] http://kobuki.yujinrobot.com

a frequency higher than 16.6 Hz. In the case of minimum speed (0.1 ms), the precision is 0.6 cm. It is worth highlighting that the sampling frequency value is an hardware parameter of our robot. Different robotics platforms may offer higher sampling frequency and consequently better precision.

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

The experiment runs are performed in a closed and straight hallway (3m wide, 30m long) at 5TONIC laboratory. Each run consists of the Robot brain driving the robot on a straight line for 15 m as shown in Fig. 2. The starting position of the robot is placed in the middle of the hallway approximately 7 meters away from the Wi-Fi AP having a thin office wall (approximately 15 cm) separating the two. Then, the robot accelerates from the starting position to the target velocity (e.g., min, max, etc.) and it drives in accordance with the closed-loop mechanism. After having traveled for 15 m, the robot stops. During the driving, an additional thicker wall (approximately 25 – 30 cm) separates the robot from the Wi-Fi AP. At the end of the driving, the robot is approximately 22 m away from the Wi-Fi AP.

3.2 Delay and Signal characterization

This section aims at characterizing how the Wi-Fi signal quality impacts the delay in controlling the robot as perceived by the Robot brain. Indeed, the publish-consume mechanism for exchanging data between the Robot brain and the ROS components is based on TCP. This means that any transmission failure occurring on the Wi-Fi channel (Layer 2) will trigger a retransmission at TCP level (Layer 4), thus introducing an undesired delay in the closed-loop mechanism. Indeed, additional delays in the delivering of the odometry sensor data result in longer reaction times in the Robot brain. Similarly, additional delays in the delivering of the movement instructions degrade the smoothness and precision of the driving. Such characterization is therefore necessary in order to adapt the closed-loop to also consider the Wi-Fi signal. To that end, we performed 10 experiment runs driving the robot at minimum speed

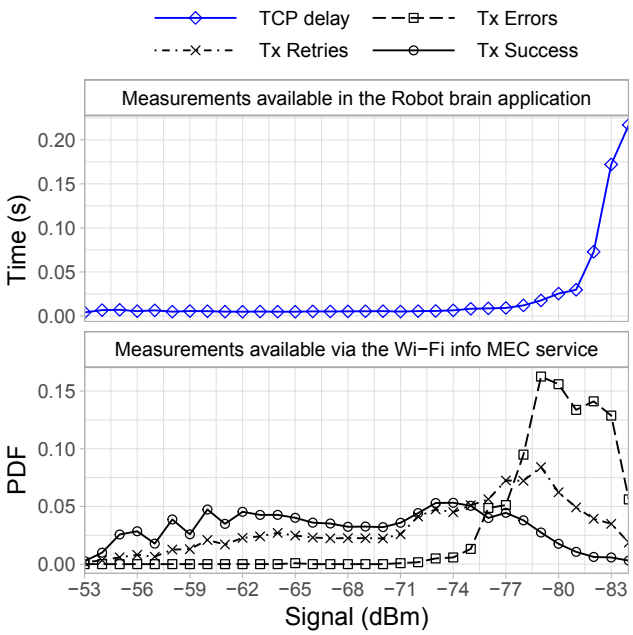

Figure 3: Signal and delay characterization

(0.1 m/s) and 10 experiment runs at maximum speed (0.75 m/s). All the edge robotics system components are synchronized and share the same time reference for accurate measurements. Throughout the duration of the experiment, we recorded in the Robot brain the Wi-Fi information obtained via the Wi-Fi information MEC service, while on the robot itself we measured the delay in receiving the movement instructions.

The obtained data from both experiments is analyzed and aggregated to generate the results presented in Fig. 3. Note that the results shown here are specific for our test-bed, and therefore can only be used as a particular realization that we use later to validate and evaluate the benefits of Edge robotics. In overall, Fig. 3 characterizes the quality of the Wi-Fi channel covering our experimental area. Regarding the measurements available via the Wi-Fi information MEC service, the *MEC: Tx Success* line shows the probability density function (PDF) of all the downstream frames successfully transmitted (from the access point to the robot) over the measured signal level in dBm. Similarly, *MEC: Tx Retries* shows the probability density function of the downstream frames retransmissions. It is worth highlighting that Wi-Fi employs an automatic retransmission mechanism in case of packet transmission error, where frames are retransmitted up to 7 times[4], and if none of the retransmissions succeeds, a frame loss occurs. *MEC: Tx Error* shows the PDF of the failed transmissions.

It can be seen that for high dBm signal values (i.e., good signal level) the probability of successful frame transmission maintains a difference proportional with respect to the number of retransmitted frames. This is due to the fact that frame retransmissions constantly occur in Wi-Fi networks because of its best-effort design principle. For lower signal strengths (below -71 dBm), the probability of having a failed transmission increases. Such probability becomes

[4]The maximum amount of retransmissions is configurable. 7 is a common default value.

drastically higher than the probability of successful transmission at signal levels lower than -77 dBm (it is actually evident that below -80 dBm it is very hard to have a successful transmission). TCP delay measurements (shown in the top graph of Fig. 3) confirm this, with values as high as hundreds of milliseconds.

4 ADAPTIVE SPEED CONTROL ALGORITHM

Based on the results provided in the previous section (Sec. 3.2), next we present the design of a control algorithm which is able to adapt the robot driving speed based on the Wi-Fi information service. The aim of the algorithm is to obtain a displacement accuracy similar to the one obtained while driving at the lowest speed, while reaching the target destination faster. Through this algorithm we showcase the benefits of consuming context information for controlling the robot, nonetheless, we acknowledge that more advanced and optimal algorithms than the one proposed in this section can be eventually designed.

The design approach followed for the proposed algorithm is tailored to the experimental evaluation performed in Sec. 5.

1: **procedure** ComputeRobotSpeed
2: *info* ← GetCurrentWiFiInfo();
3: *buffer* ← buffer.removeOldestWiFiInfo();
4: *buffer* ← buffer.add(info);
5: *signalLevel* ← buffer.average();
6: if *signalLevel* > −71 *dBm* then *speed* ← 0.75;
7: else if *signalLevel* < −81 *dBm* then *speed* ← 0.1;
8: else *speed* ← (*signalLevel*+81 *dBm*)/ 10 *dBm* + 0.1;

Algorithm 1: Adaptive control speed algorithm

During the experiments described in Sec. 3, we collected the information on the Wi-Fi signal every 10 ms. We observed that the Wi-Fi signal level presents significant oscillations in case of averaging it over a short time window (e.g., 50 ms). That is, two subsequent average measurements may report considerably different Wi-Fi signal levels. On the contrary, if we take a longer time window (e.g., 500 ms), the oscillations between subsequent average measurements are substantially reduced and the Wi-Fi signal varies in a smoother way. Based on this finding, the control algorithm will use the Wi-Fi signal level obtained by averaging it over a fixed time frame. Given the robot's speed bound between 0.1 m/s and 0.75 m/s, a time frame of 500 ms is considered to be a reasonable value. The computed Wi-Fi signal is then combined with the robot's odometry sensor data for adapting the robot's speed.

Alg. 1 shows the pseudo-code of the control algorithm. The Robot brain, in real-time, extracts the current signal level from the Wi-Fi MEC information service, stores it in a circular buffer and computes the moving average of the Wi-Fi signal level. For each movement command, the adaptive speed and the adaptive distance are re-calculated. In Sec. 3.2 we observed that packet retransmissions and failures start increasing for signal values below -71 dBm, hitting their maximum between -79 and -81 dBm. Based on this observation, the control algorithm adapts the driving robot's speed to the maximum (0.75 m/s) for an average Wi-Fi signal level higher than -71 dBm. On the opposite end, the minimum robot speed (0.1 m/s) is selected for an average Wi-Fi signal level equal or lower than -81 dBm. Between -71 dBm and -81 dBm, the control algorithm linearly adapts the robot's speed to the Wi-Fi signal level (e.g., 0.425 m/s with -76 dBm).

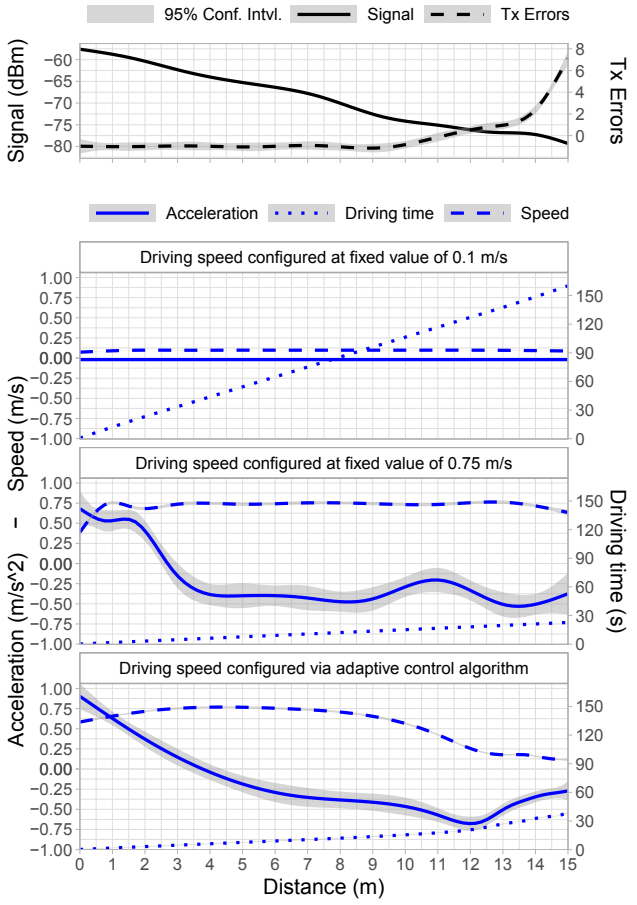

Figure 4: Speed, acceleration, and driving time

5 EXPERIMENTAL EVALUATION

This section evaluates the adaptive speed control algorithm proposed in Sec. 4 and compares it with scenarios not making use of any context information. The following three scenarios are evaluated: (*i*) the robot drives at minimum speed (0.1 m/s), (*ii*) the robot drives at maximum speed (0.75 m/s), and (*iii*) the robot uses our control algorithm to drive at adaptive speed.

Following the experimental methodology described in Sec. 3.1, we performed 10 experiment runs for each scenario (minimum speed, maximum speed, adaptive speed). In addition to the Wi-Fi information recorded in the Robot brain, we record the odometry sensor data directly in the robot itself. This is because the data from the odometry sensors is not timestamped, and sending it over the Wi-Fi channel would not be suitable for measuring the speed and acceleration experienced by the robot (due to risk of transmission failures over Wi-Fi).

The measured data is aggregated and analyzed to produce the results on Fig. 4. The figure has four different graphs. On each graph

the x-axis is the distance traveled during the experiment, from the start (0 m) to the end (15 m). The first subgraph from the top presents the Wi-Fi signal level (y-axis on the left) and the transmission errors over the robot driving path (y-axis on the right). As it can be noticed, there is a significant decay on the Wi-Fi signal quality in the last 5 meters of the driving path reflected by an exponential increase of the transmission errors. The remaining three graphs of Fig. 4 present – for each evaluated scenario – the speed, the acceleration, and the driving time as measured via the odometry sensor data. Despite the acceleration and the speed having different units (m/s and m/s^2, respectively), they share the same y-axis on the left since they present the same range of values. The y-axis on the right represents the elapsed driving time since the start of the experiment run.

In the minimum speed experiment, the robot speed is set constant to 0.1 m/s from the start to the end. Similarly, the acceleration presents a constant value in the order of few cm/s^2. Driving such a low speed results in a smooth run that is not affected by the degradation of the Wi-Fi channel in the last segment of the path, since the slowness of the movement allows for more time to recover from possible transmission errors and retransmissions. As a drawback, the robot requires $\sim$ 160 seconds to complete each experiment run. On contrary, the maximum speed experiment is the one requiring less time ($\sim$ 27 seconds).The impact of the decreasing Wi-Fi signal quality can be seen in the acceleration curve (notably in the last 5 meters of the path) where the acceleration fluctuates due to increased packet delay or delayed reaction, resulting in a stop-drive effect of frequent braking and spurring acceleration to full-speed. Effect of the stop-drive behavior, the driving direction is deviating from the straight driving path.

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

6 CONCLUSIONS AND FUTURE WORK

This paper first presents the challenges of today's Cloud robotics systems and the opportunities offered by Edge computing. One of the key differentiating features of Edge computing is the possibility for applications running at the Edge to consume context information about the network. This can be used to optimize the robotics systems operations in ways otherwise impossible in the Cloud. Following the Edge computing concept, we have designed an Edge robotics system blending together the Robot Operating System (ROS) – which offers a common development framework for robotics applications – and the ETSI MEC architecture, which defines a common framework for Edge computing. An experimental environment is deployed in the 5TONIC laboratory where one mobile robot is employed. We first perform a set of experiments to characterize the relation between the robot control delay and the Wi-Fi signal strength. The resulting characterization has been used as a baseline for designing, implementing and experimentally evaluating a control algorithm which consumes context information about the Wi-Fi signal and adapts the robot's speed for a smoother driving. Our experimental results show that adapting the robot's speed based on the Wi-Fi signal provided by the MEC information service can effectively produce a smoother driving at high speeds. This improvement allows the robot to operate faster compared to the case of not consuming any context information. Future work is therefore expected in the designing of more advanced control algorithms and its experimental evaluation under several different conditions. As an example, we are currently looking into how consuming a wider source of context information can make robotics applications adaptable to larger and dynamic environments.

REFERENCES

[1] 3GPP. 2018. *Service requirements for next generation new services and markets.* Technical Specification (TS) 22.261 v16.4.0. 3rd Generation Partnership Project (3GPP).

[2] 5G-CORAL. 2018. *5G-CORAL initial system design, use cases, and requirements.* Deliverable 1.1.

[3] 5TONIC. 2018. *5TONIC: Open-research and innovation laboratory for 5G technologies.* [Online]. https://www.5tonic.org/ [Accessed 13 Sep. 2018].

[4] ETSI. 2016. *Mobile Edge Computing (MEC); Framework and Reference Architecture.* Group Specification (GS) 003 v1.1.1. European Telecommunications Standards Institute (ETSI).

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

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

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

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

[9] A. Hammad, S. S. Ali, and A. S. T. Eldien. 2017. A novel implementation for FastSLAM 2.0 algorithm based on cloud robotics. In *2017 13th International Computer Engineering Conference (ICENCO).* 184–189. https://doi.org/10.1109/ICENCO.2017.8289785

[10] J. Karjee, S. Behera, H. Kumar Rath, and A. Simha. 2017. Distributed Cooperative Communication and Link Prediction in Cloud Robotics. In *2017 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops).* 1–7. https://doi.org/10.1109/SECONW.2017.8011039

[11] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering* 12, 2 (April 2015), 398–409. https://doi.org/10.1109/TASE.2014.2376492

[12] R. Rahimi, C. Shao, M. Veeraraghavan, A. Fumagalli, J. Nicho, J. Meyer, S. Edwards, C. Flannigan, and P. Evans. 2017. An Industrial Robotics Application with Cloud Computing and High-Speed Networking. In *2017 First IEEE International Conference on Robotic Computing (IRC).* 44–51. https://doi.org/10.1109/IRC.2017.39

[13] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret. 2018. Context-Aware Cloud Robotics for Material Handling in Cognitive Industrial Internet of Things. *IEEE Internet of Things Journal* 5, 4 (Aug 2018), 2272–2281. https://doi.org/10.1109/JIOT.2017.2728722

[14] X.V. Wang, L. Wang, A. Mohammed, and M. Givehchi. 2017. Ubiquitous manufacturing system based on Cloud: A robotics application. *Robotics and Computer-Integrated Manufacturing* 45 (2017), 116–125. https://doi.org/10.1016/j.rcim.2016.01.007 Special Issue on Ubiquitous Manufacturing (UbiM).