An Approach to Reduce Network Effects in an Industrial Control and Edge Computing Scenario

Rômulo A. L. V. de Omena, Danilo F. S. Santos and Angelo Perkusich

Embedded Systems and Pervasive Computing Lab, Federal University of Campina Grande, 882 Aprígio Veloso St., 58429-970, Campina Grande, PB, Brazil

Keywords: IIoT, Industry 4.0, Edge Computing, 5G, Model Predictive Control, AGV.

Abstract: The cloud-based nature of Industry 4.0 enhances its flexibility and scalability features. To support time-sensitive and mission-critical applications, whereby low latency and fast response are essential requirements, usually cloud computing resources should be placed closer to the industry. The Edge Computing concept combined with next-generation networks, such as 5G, may fulfill those requirements. This paper presents an experimental system setup that combines a Model Predictive Control approach with a compensation strategy to mitigate network delay and packet loss. The experimental system has two sides, namely, the edge and the local side. The former executes the controller and connects to the local side through a network. The latter is attached to the application and has lower computing capabilities. In our setup, the application under control is a two-wheeled mobile robot, which could act as an Automated Guided Vehicle. We defined two control objectives, the Point Stabilization, and the Trajectory Tracking, which ran through distinct network conditions, including delays and packet losses. These control objectives are only validation scenarios of the proposed approach but could be replaced by a real case path planner. The obtained results suggest that the approach is valid.

1 INTRODUCTION

Driven by the Internet of Things (IoT) paradigm, the new era of computing is bringing out the Internet from the traditional desktop to many objects around us (Gubbi et al., 2013), based on ubiquitous computing, where smart environments recognize, identify objects, and retrieves information (Yaqoob et al., 2017). Applications are spread in several domains, such as smart homes, healthcare, industrial automation, agriculture, school, market, and vehicles (Al-Fuqaha et al., 2015).

The application of IoT in industries covers an area called the Industrial Internet of Things (IIoT). In an IIoT system, the industrial "things", such as sensors, controllers, actuators, production lines, and equipment, connect to the Internet. They can improve productivity, efficiency, safety, and intelligence of industrial factories and plants (Xu et al., 2018). Sisinni et al. (2018) considers the IIoT as a subset of IoT, and merging IIoT and Cyber-Physical Systems (CPS) results in the Industry 4.0. An essential feature of IIoT is the cloud-based nature, making it more flexible and scalable than conventional industrial systems (Wan et al., 2016).

To fulfill the requirements of mission-critical and time-sensitive systems, where low latency communication and fast response are essential, the physical distance to the cloud computing servers may not meet those requirements. To deal with cloud computing limitations, the "Edge Computing" concept has emerged, bringing cloud services closer to devices. It is also known as the "cloud closer to the ground" (Kaur et al., 2018). In an industrial environment, edge computing introduces an intermediate level, separating the field level domain from cloud-based services. The field devices perform tasks regarded to industrial automation and need to react faster; therefore, latency and reliability are real-time constraints and requirements of communication links (Pallasch et al., 2018). In the context of IIoT and Industry 4.0, edge computing provides high bandwidth, low latency, and low jitter, necessary to process urgent and complex tasks on time. Battery-powered devices and sensors, restricted in energy consumption, also take advantage of edge computing, as tasks demanding high processing are transferred to a higher layer (Aazam et al., 2018).

Qiu et al. (2020) highlights some advantages of

V. de Omena, R., Santos, D. and Perkusich, A.
An Approach to Reduce Network Effects in an Industrial Control and Edge Computing Scenario.
DOI: 10.5220/0010496502960303
In *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021)*, pages 296-303
ISBN: 978-989-758-510-4

the Edge Computing in IIoT: 1) improve system performance by reducing the overall delay of the system; 2) protect data security and privacy, reducing risks of data leakage; 3) reduce operational costs, as the data uploading volume to cloud and bandwidth consumption is reduced. Chen et al. (2018) discuss the cooperation mechanism between Cloud Computing and Edge Computing. On the one hand, Edge Computing better supports real-time processing. On the other hand, Cloud Computing focuses on analyzing big data and knowledge mining from the data obtained at the edge, playing an important role in periodic maintenance, decision support, and other tasks that do not need to be performed in real-time.

When the mobile network provides the edge computing services, it is called Multi-Acess Edge Computing (MEC) (Abbas et al., 2018), which delivers computing capabilities through the Radio Access Network (RAN), therefore reducing latency and improving the Quality of Service (QoS) (Kiani and Ansari, 2018). 5G mobile networks are expected to bring networks with ultra-low latency and ultra-high reliability (ULLRC). Thus, the merging of MEC and 5G is expected to attend mission-critical IoT demands and the Tactile Internet. For the latter, the requirement of end-to-end latency is about 1 ms (Rimal et al., 2017).

The control systems based on Cloud or Edge Computing inherits some features of the Networked Control Systems (NCS) field (Hespanha et al., 2007). However, differently, the legacy hardware controllers are replaced by software instances. The controllers' softwarization improves the operational efficiency and flexibility and can be provisioned on-demand and are easier to be upgraded (Zhao and Dán, 2018).

The IIoT and Industry 4.0 are paradigms that might be empowered by the new upcoming edge computing and 5G technologies. Motivated by this context, we present an experimental setup of a control architecture implemented using Edge Computing, taking a two-wheeled mobile robot as a use case. Inside the industry, we consider that the wheeled mobile robot can act as an Automated Guided Vehicle (AGV) transporting goods from A to B. The results from the initial stage of this research are presented in this paper, in which techniques to control the mobile robot through Edge Computing are under evaluation. Such an approach aims to reduce the effects of the communication network in the control system. For this purpose, the Model Predictive Control (MPC) with a delay and packet loss compensation method is applied to control the mobile robot at different network conditions. Two different control objectives are performed, the Point Stabilization and the Trajectory Tracking, ran through distinct network conditions, including delays and packet losses.

The remainder of this paper is organized as follows. A literature review, including the main concepts, is presented in Section 2. More details about the experimental setup and the algorithms are presented in Section 3. The validation and results discussion are in Section 4. Finally, the conclusions are presented in Section 5.

2 LITERATURE REVIEW

A networked predictive control has been proposed in Liu et al. (2007) to deal with the communication delay in both forward and feedback channels. For compensation of the random network delay, the controller can generate a control sequence within a prediction horizon, sent in a single packet to the plant. Without communication delay, only the control sequence's first signal is applied to the plant, and the remaining predicted control signals are discarded. When the network is subject to communication delay, the forward channel (controller-to-actuator) will be delayed or dropped. Therefore, the predicted control inputs from the last available sequence are applied to the plant until a new packet has arrived. To compensate for the feedback (sensor-to-controller) delay, a predictor is used to predict the current plant state $\hat{x}$.

Similarly, Findeisen and Varutti (2009) uses a nonlinear model predictive controller to compensate for the network nondeterminism. The timestamps of the packets coming from the plant side is used to estimate the τ_{sc} (sensor-to-controller delay). Given that the delays on the actuation side are stochastic, the τ_{ca} (controller-to-actuator delay) is assumed to be known. The state prediction can now be calculated, considering τ_{sc} and τ_{ca}, and the MPC optimization problem is triggered. Hence, the control packet timestamp is shifted to τ_{ca} and sent to the actuator. This one is tasked with applying the control input when the timestamp matches the actuator's inner clock. Other similar compensation strategies can be seen (Pin and Parisini, 2011), also including wireless networks (Ulusoy et al., 2011) and decentralized (Hojjat A. Izadi and Zhang, 2011) or distributed (Grüne et al., 2014) approaches.

When the industrial control is merged with the cloud computing, the "as a service" concept applied to control comes up. Esen et al. (2015) investigates the "Control as a Service" concept to control a car using cloud computing. Analogously, Hegazy and Hefeeda (2015) inserts the industrial automation in the cloud services paradigm. To attend the realtime requirements, the execution of industrial control in the edge computing has been investigated (Abdelzaher et al., 2020; Chen et al., 2020).

Relocating the MPC controller to the cloud or edge computing is already a research area. Skarin et al. (2018) combines the edge computing with a 5G network and performs tests involving some nodes at Lund, Sweden. They evaluate the MPC's performance as the controller of a ball and beam process, implemented in nodes with different processing capabilities, and hosted at different places. Skarin et al. (2019) runs the MPC in the edge, however, a Linear Quadratic Regulator (LQR) implemented locally assumes when the packets are delayed or lost. Vick et al. (2016) use a computer to represent a cloud service, run an MPC to control a robot and to compensate the network delays.

Based on the literature review, the delay and packet loss compensation methods with MPC is adopted here. Additionally, the future trajectory to be followed by the mobile robot is included in the MPC optimization problem for the Trajectory Tracking simulations. This procedure will make the controller preview the linear and angular velocities control sequence necessary to the robot follow the desired path during the prediction horizon. The actuator uses such control sequence to compensate for the delays and packet losses.

Up to now, this works involves only the interaction between the Edge Computing (named here as the edge side) and the lower level (named local side). The latter is the controlled plant or process, here, represented by the mobile robot or AGV. Next steps may offload more tasks to the edge side, as the path planning and collision avoidance procedures, to name a few. Additionally, the Cloud Computing may cooperate with the edge side processing massive data related to tasks that not requires fast response.

3 EXPERIMENTAL SETUP

The basic architecture of the experimental setup is shown in Figure 1. The evaluation of the system is carried out through two computers. The first represents the edge computing side and performs as a server. The second, located at the local side and attached to the robot, act as the client, sending state measurements and waiting for responses from the server containing the control sequence input. For both sides, the communication protocol used to deliver and receive packets is the UDP.

The robot used is the Pioneer 3-DX and is simulated in the CoppeliaSim (Rohmer et al., 2013). Through an Application Programming Interface

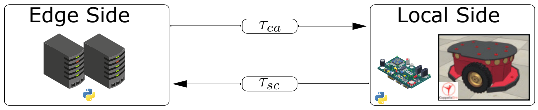

Figure 1: Basic architecture of the experimental setup.

(API), the CoppeliaSim is linked to the local side through a TCP/IP socket. Note that the CoppeliaSim runs in a computer over an operating system; however, since the tool is simulating the robot at the local side, the computer associated is considered to have limited computing capabilities, but can run the sensor and actuator routines, and can send/receive data by the communication network. Suppose that a physical robot would be used. In this case, the computer which simulates the robot in CoppeliaSim would be changed by a physical robot attached to a computer with limited resources (e.g., a Raspberry Pi) or even a microcontroller.

The flowcharts of Figure 2(a) and 2(b), respectively, describes how the edge code and the sensor thread of the local code works. Both software executed at the edge and local sides are written in Python and are referred to as the edge code and local code. The network delays are estimated through timestamps. τ_{sc} in the edge code, while τ_{ca} is estimated by the local code and is given back to the edge code. The edge and local clocks are assumed to be synchronized. τ_{mpc} is estimated during the MPC optimization running. With these delays, the predictor can finally predicts the robot's state and trigger the MPC optimization problem solved through CasADi (Andersson et al., 2019). The sensor sampling interval is T (equal to the time step of the MPC predictions).

The experimental procedure is facilitated by implementing the edge and local side on the same computer. However, the local code and CoppeliaSim runs in the native operating system, while the edge code runs in a Linux virtual machine. The virtualization software creates the virtual network interface between the native and virtualized operating system. The ERRANT (EmulatoR of Radio Access NeTworks) developed by Trevisan et al. (2020) and the Linux NetEm (Network Emulator) has been adopted to emulate some network profiles. The ERRANT is an open-source tool that emulates mobile networks based on a data set composed of real measurements and runs on top of NetEm. That tool was used in this work to emulate the 3G and 4G networks. The NetEm was also purely applied to emulate a network with packet loss and a 5G network. The network emulation was carried out in the virtual machine at the local side. More details about the overall system operation are discussed in Section 4.

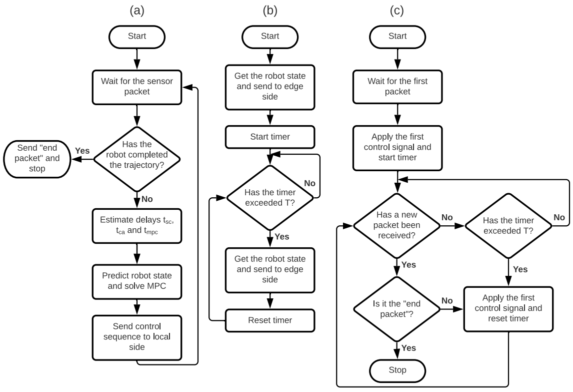

Figure 2: Flowchart describing the operation of: (a) Edge code, (b) and (c) the sensor thread and the compensation strategy of the local code, respectively.

3.1 Model Predictive Control

The MPC, sometimes referred to as Receding Horizon Control, obtains control actions by solving a finite horizon open-loop optimal control problem at each sampled state. The plant's measured state is taken as the initial state for the optimization problem, calculated within a prediction horizon N. This MPC procedure makes it different from conventional control, on what a pre-computed control law is adopted. The MPC can deal with constraints, defining limits for the plant state's control input and safety limits. Another essential feature is the capability to control a multivariable process (Mayne et al., 2000).

The optimization problem is determined to minimize the following cost function J:

$$\underset{u}{\text{minimize}}\ J = \sum_{k=0}^{N-1} x'(k)Qx(k) + u'(k)Ru(k) \qquad (1)$$

$$\text{subject to } g1 \leq G1 \text{ and } g2 = G2$$

where $x(k)$ and $u(k)$ denotes the state and the control input, respectively. The diagonal matrix Q penalizes the difference between the predicted state and the reference, while the matrix R does the same with the control input. G_1 and G_2 are, respectively, the inequality and equality constraints. The basic MPC (Bemporad and Morari, 1999) so runs repeatedly the following sequence:1) get the new state $x(k)$; 2) solve the optimization problem (1); 3) apply only $u(k) = u(k+0|k)$; 4) make $k \leftarrow k+1$ and go to step 1.

3.2 Compensation Strategy

The compensation strategy for the network delay and packet loss is deployed in the local code and is briefly represented by the flowchart of Figure 2(c). After the local code has started, it waits for the first packet coming from the edge side. When received, it applies the first control signal of the sequence generated by the MPC and start the timer. Whenever a new packet has arrived, the first control signal is applied and the timer is reset. If the packet is delayed or lost and the time exceeds T, the next control signal is applied and the timer is again reset. If the last control signal has been reached, it continues being applied until a new packet is received. The "end packet" is a packet sent by the edge side when the robot has completed the trajectory. It is sent several times to ensure that it will reach the other side in case of packet losses.

3.3 Mobile Robot

The differential driving mobile robot, classified as a non-holonomic mobile robot, may have two rear wheels and a front castor wheel, or a four-wheel configuration. The two wheels configuration is used here. Details of the kinematic equation can be seen in Dongbing Gu and Huosheng Hu (2006).

The state and control signal vectors are respectively denoted as $\mathbf{x} = [x\ y\ \theta]^T$ and $\mathbf{u} = [v\ \omega]^T$. The state variables x and y represents the Cartesian position in meters of the robot, and θ is the robot orientation in radians referenced from the x axis. The control variables are the linear speed v, in meters per second, and the angular speed ω, in radians per second. The robot's linear and angular speeds depends on the left and right wheel speeds and the distance between the wheels.

4 VALIDATION

The validation procedure was based on tests conducted by Dongbing Gu and Huosheng Hu (2006) and Mehrez et al. (2013), whose MPC is applied to control a wheeled mobile robot. The same control objectives are adopted here, which are Point Stabilization and Trajectory Tracking. The items included in the CoppeliaSim scene are the Pioneer 3-DX robot and a floor with dimensions 25x25 m, where x and y covers the intervals from -12.5 to 12.5 m.

It's important to note that, for now, the main idea is to validate the compensation strategy detailed in the previous section. The trajectories followed by the AGV during the validation are not necessarily a real case of the Industry 4.0 or any other application, but is enough to evaluate the controlling of the AGV at distinct network conditions, including favorable conditions, the case of a 5G network, or bad conditions, the case of a network with high delays and packet losses.

4.1 Point Stabilization

For the Point Stabilization simulations, the robot should leave from the initial state $\mathbf{x}_0$ and stabilize at the desired reference $\mathbf{x}_r$. The vectors were adjusted as $\mathbf{x}_0 = [-11.5\ -11.5\ 0]^T$ and $\mathbf{x}_r = [11.5\ 11.5\ \pi]^T$ for all the simulations. If the reference orientation angle is chosen π, the robot should maneuver near to the state constraints and, thus, the MPC performance in this circumstance can be evaluated. The controller time step was set as $T = 40$ ms and the prediction horizon size as $N = 40$, resulting in a prediction horizon of 1600 ms in time. The sensor sampling period is made equal to T. The state constraints imposed to the MPC are $-12.5 \leq x \leq 12.5$ m and $-12.5 \leq y \leq 12.5$ m, which coincides with the floor dimensions set in the CoppeliaSim scene. For the control signals, the constraints are $-1.2 \leq v \leq 1.2$ m/s and $-\pi/4 \leq \omega \leq \pi/4$ rad/s for the linear and angular speeds, respectively.

Four network profiles have been taken an account to evaluate the robot controlling in different network conditions. The 3G and 4G networks were emulated through the ERRANT. Since the tool emulates the network based on a data set of real measurements, there are different bandwidth and latency situations for a single network technology, which could be related to the signal quality, for example. In this case, for the 3G and 4G profiles, the network conditions were configured to change every 5 seconds. The NeTem purely was used to emulate the two other profiles. For the 5G profile, no changes have been applied to bandwidth, and the whole bandwidth of the Gigabit virtualized network interface is available. The uplink and downlink delay was adjusted to 0.5 ms with a normal distribution of 0.05 ms, resulting in a total latency of 1 ms. The other profile, called 200msPL50%, has 100 ms of delay per channel, with a normal distribution of 10 ms, totalizing a latency of 200 ms. The same profile has 50% of packet losses per channel. The weights of the MPC cost function are defined as diagonal matrixes and were adjusted to Q = diag(20,20,30) and R = diag(20,60). The two first elements of Q penalizes the states x and y, while the last does with θ. For R, the first and second elements penalize the linear and angular speeds control signals, respectively.

In Figure 3(a) is shown the trajectory of the robot obtained for all the network profiles. The red circle circumscribes the region where is the initial point, and the green circle, the final point. The blue circle delimits the region where the robot maneuver to stabilizes in the desired reference. The maneuver occurs because the reference angle is set as π, and, in this case, the robot should turn left and apply a reverse speed to reach the final point.

The control signals related to the linear and angular speeds, applied to the robot, for each network profile, are shown in Figure 3(b) to 3(e). The simulations were programmed to end when the absolute value of the difference between the reference and the state vectors' norms was less or equal to 0.2. This occurred after approximately 32 s.

About the compensation strategy implemented by the actuator in the local side, the biggest step in the control sequence (for a prediction horizon of $N = 40$) was to the 4$^{\text{th}}$ element (starting from zero) for the 5G profile. It means that, at one or more particular moments of the simulation, the local side remained four times the controller's time step (T) without receiving a new packet. Thus, the actuator has continued applying the control signals of the sequence generated by the MPC until the arrival of a new packet with a more recently calculated control sequence. For the 4G and 3G profiles, the advancing has been to the 7$^{\text{th}}$ and 15$^{\text{th}}$ elements, respectively. As expected, the control signal sequence steps would be bigger in the 200msPL50% profile, once a percentage of the packets are lost. In this case, the actuator advanced until the 25$^{\text{th}}$ element. Even with the delays and packet losses, the MPC combined with the compensation strategy can stabilize the robot at the desired reference.

4.2 Trajectory Tracking

In the Trajectory Tracking, all the simulations procedures and adjustments were the same, except for the reference. Now, the reference vector is not fixed, but is time varying. Two trajectories with different shapes were simulated, the circular-shape and eight-shape. The reference vector is given by $\mathbf{x}_r = [x(t)\ y(t)\ \theta(t)]^T$. For the circular-shape, $x(t)$ and $y(t)$ are calculated by $x_1(t)$ and $y_1(t)$ from (2). $x_2(t)$ and $y_2(t)$ from the same equation are used for the eight-shape. For both shapes, $\theta(t)$ is calculated by (3), where T is the controller time step.

$$\begin{array}{ll} x_1(t) = 10.0\sin(0.1t) & x_2(t) = 11.5\sin(0.1t) \\ y_1(t) = 10.0\cos(0.1t) & y_2(t) = 10.0\cos(0.05t) \end{array} \tag{2}$$

$$\theta(t) = \arctan\left[\frac{y(t+T)-y(t)}{x(t+T)-x(t)}\right] \tag{3}$$

Another difference for the Trajectory Tracking is how the MPC cost function is calculated. The cost function now takes an account not only a reference point but rather a vector containing N points (remember that N is the prediction horizon size). This approach so brings better results once the controller can "see"

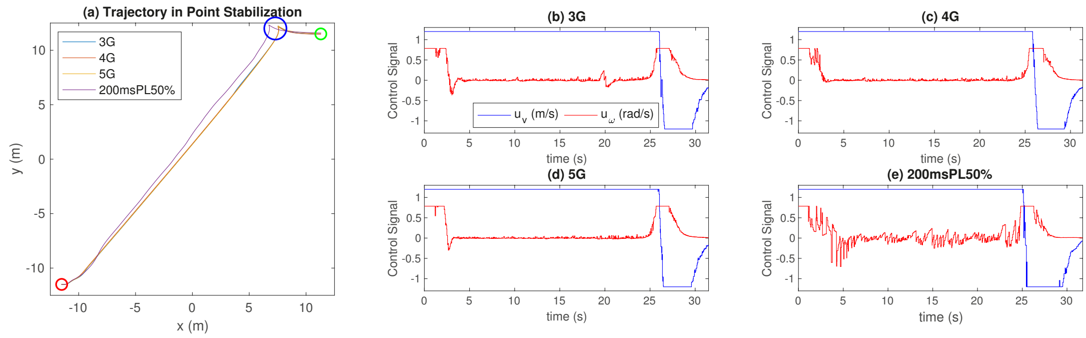

Figure 3: Point stabilization results for all the network profiles: (a) Trajectory of the robot, (b)–(e) Control signals.

the future trajectory and suitably calculate the control signal sequence. On the other hand, the processing time increases. The prediction horizon was reduced to $N = 30$, and the controller time step and sensor sampling period were increased to $T = 80$ ms to reduce the impact of that. One exception was considered for the simulation's prediction horizon size with the 5G profile, adjusted to $N = 20$, as the network conditions are much more favorable. In this case, the controller's advancing into the future can be smaller, reducing the MPC processing time.

For both trajectory shapes in all the network profiles, the Q weights were adjusted to Q = diag(30,30,0.2), except in the 200msPL50% profile, where it was adjusted to Q = diag(40,40,0.2). For the circular-shape, the R weight was set as R = diag(50,50) in all the simulations. In case of the eight-shape, the R diagonal matrix received the values R = diag(10,5) for the three first profiles and R = diag(20,20) for the 200msPL50% profile.

The trajectory of the robot for the circular-shape and all network profiles, can be seen in Figure 4(a), whereas the controls signals are depicted in Figure 4(b) to 4(e). The red circle delimits the region of the initial state, denoted by the vector $\mathbf{x}_0 = [0\ 10.5\ 0]^T$. The initial state is outside the circular perimeter; however, the controller quickly moves the robot to the reference trajectory, even more in the 5G profile. The robot completes the trajectory in the clockwise direction and starts a new lap, but immediately stops in the region delimited by the green circle. The simulation was programmed to end after 70 s, time enough to complete a lap, and complete the beginning of a new one. The same procedures were repeated with the eight-shape, and the results can be seen in Figure 5. In this case, the initial state is $\mathbf{x}_0 = [0\ 10\ 0]^T$ and the simulation time was adjusted to 130 s.

The circular-shape trajectory is smoother and easier to follow, as maintaining constant the linear and angular speeds are enough to keep the robot in the reference trajectory. Oppositely, in the eight-shape, the controller should continuously make changes in the linear and angular speeds. Nevertheless, even in bad network conditions, the trajectory is successfully followed and are very similar for all the network profiles. As expected, Trajectory Tracking has better results because the controller considers the future trajectory to generate the control sequence. Therefore, even if the packets forwarded to the local side are delayed or lost, the MPC's predictive nature can generate a control sequence exploited by the actuator to keep the robot in the desired reference.

If a path planner is adopted, the planned path can be provided to the MPC and the Trajectory Tracking method can be used. In the same way, the actuator in the local side will be able to make the compensations when necessary and track the robot in the planed path.

5 CONCLUSIONS

This paper introduced an experimental setup of a system for compensation of delays and packet loss in communication networks. In the proposed system, Edge computing supports the MPC execution to control a mobile robot. The actuator uses the control sequence generated by the MPC, within the prediction horizon to compensate. We implemented two control objectives: Point Stabilization and Trajectory Tracking. In the first, the robot should leave from an initial state and reach the reference. The second represents this work's main contribution since the future trajectory points are included in the MPC optimization problem. The control sequence is, in this case, generated according to the future trajectory. The actuator takes advantage of the last received control sequence when the packets from the edge are delayed or lost. Thus the control signals are continuously applied to the robot. The favorable conditions of the 5G network allow the reduction of the prediction horizon, and consequently, the MPC has a faster response.

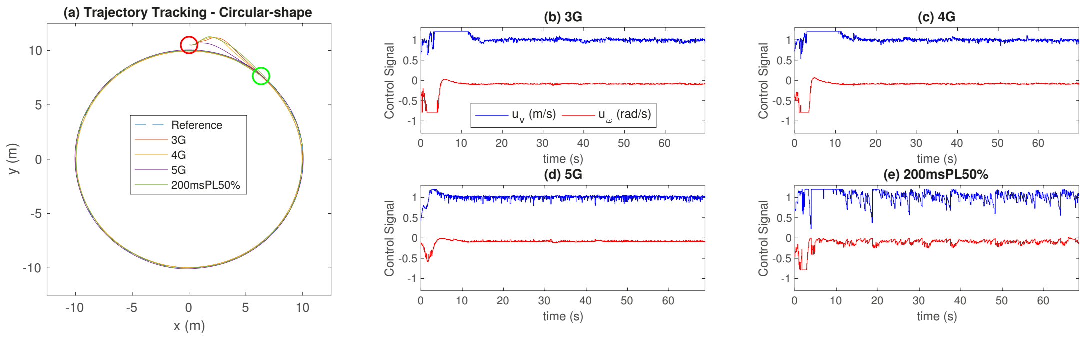

Figure 4: Circular-shape results for all the network profiles: (a) Trajectory of the robot, (b)–(e) Control signals.

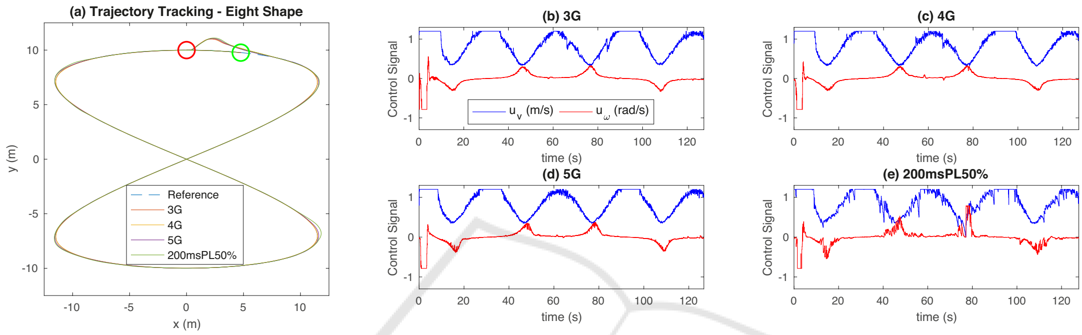

Figure 5: Eight-shape results for all the network profiles: (a) Trajectory of the robot, (b)–(e) Control signals.

The next steps may comprise a more realistic scenario of Industry 4.0, including multiple AGV's using path planning, including autonomous interactions between AGV's and other actors of the manufacturing process. The use of Edge Computing may support those mentioned tasks, but also other relevant ones, like, change the parameters of the controller according to the load characteristics (e.g., dimensions, weight, and fragility), promote the cooperation among multiple AGV's to transport a heavier load, use Artificial Intelligence to help on the decision-making and other tasks that requires more computing power. Edge Computing also helps to reduce the cost with the robots and all devices and machinery related to the industrial environment, given that the computing power will be placed at the edge, besides, the battery consumption of the mobile robots may be reduced.

ACKNOWLEDGEMENTS

The authors thank the support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES), and Programa de Pós-Graduação em Engenharia Elétrica (COPELE), Federal University of Campina Grande.

REFERENCES

Aazam, M., Zeadally, S., and Harras, K. A. (2018). Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(10):4674–4682.

Abbas, N., Zhang, Y., Taherkordi, A., and Skeie, T. (2018). Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465.

Abdelzaher, T., Hao, Y., Jayarajah, K., Misra, A., Skarin, P., Yao, S., Weerakoon, D., and Årzén, K.-E. (2020). Five challenges in cloud-enabled intelligence and control. *ACM Trans. Internet Technol.*, 20(1).

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

Bemporad, A. and Morari, M. (1999). Robust model predictive control: A survey. In Garulli, A. and Tesi, A., editors, *Robustness in identification and control*, pages 207–226, London. Springer London.

Chen, B., Wan, J., Celesti, A., Li, D., Abbas, H., and Zhang, Q. (2018). Edge computing in iot-based manufacturing. *IEEE Communications Magazine*, 56(9):103–109.

Chen, C., Lyu, L., Zhu, S., and Guan, X. (2020). On-demand transmission for edge-assisted remote control in industrial network systems. *IEEE Transactions on Industrial Informatics*, 16(7):4842–4854.

Dongbing Gu and Huosheng Hu (2006). Receding horizon tracking control of wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(4):743–749.

Esen, H., Adachi, M., Bernardini, D., Bemporad, A., Rost, D., and Knodel, J. (2015). Control as a service (caas): Cloud-based software architecture for automotive control applications. In *Proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud*, page 13–18, New York, NY, USA.

Findeisen, R. and Varutti, P. (2009). *Stabilizing Nonlinear Predictive Control over Nondeterministic Communication Networks*, pages 167–179. Springer Berlin Heidelberg, Berlin, Heidelberg.

Grüne, L., Allgöwer, F., Findeisen, R., Fischer, J., Groß, D., Hanebeck, U. D., Kern, B., Müller, M. A., Pannek, J., Reble, M., Stursberg, O., Varutti, P., and Worthmann, K. (2014). *Distributed and Networked Model Predictive Control*, pages 111–167. Springer International Publishing, Heidelberg.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660.

Hegazy, T. and Hefeeda, M. (2015). Industrial automation as a cloud service. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2750–2763.

Hespanha, J. P., Naghshtabrizi, P., and Xu, Y. (2007). A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162.

Hojjat A. Izadi, B. W. G. and Zhang, Y. (2011). Decentralized model predictive control for cooperative multiple vehicles subject to communication loss. *International Journal of Aerospace Engineering*, 2011:1–13.

Kaur, K., Garg, S., Aujla, G. S., Kumar, N., Rodrigues, J. J. P. C., and Guizani, M. (2018). Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Communications Magazine*, 56(2):44–51.

Kiani, A. and Ansari, N. (2018). Edge computing aware noma for 5g networks. *IEEE Internet of Things Journal*, 5(2):1299–1306.

Liu, G., Xia, Y., Chen, J., Rees, D., and Hu, W. (2007). Networked predictive control of systems with random network delays in both forward and feedback channels. *IEEE Transactions on Industrial Electronics*, 54(3):1282–1297.

Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814.

Mehrez, M. W., Mann, G. K. I., and Gosine, R. G. (2013). Stabilizing nmpc of wheeled mobile robots using open-source real-time software. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6.

Pallasch, C., Wein, S., Hoffmann, N., Obdenbusch, M., Buchner, T., Waltl, J., and Brecher, C. (2018). Edge powered industrial control: Concept for combining cloud and automation technologies. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 130–134.

Pin, G. and Parisini, T. (2011). Networked predictive control of uncertain constrained nonlinear systems: Recursive feasibility and input-to-state stability analysis. *IEEE Transactions on Automatic Control*, 56(1):72–87.

Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M., and Wu, D. O. (2020). Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys Tutorials*, 22(4):2462–2488.

Rimal, B. P., Van, D. P., and Maier, M. (2017). Mobile edge computing empowered fiber-wireless access networks in the 5g era. *IEEE Communications Magazine*, 55(2):192–200.

Rohmer, E., Singh, S. P. N., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326.

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., and Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734.

Skarin, P., Eker, J., Kihl, M., and Årzén, K.-E. (2019). Cloud-assisted model predictive control. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 110–112.

Skarin, P., Tärneberg, W., Årzen, K., and Kihl, M. (2018). Towards mission-critical control at the edge and over 5g. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 50–57.

Trevisan, M., Safari Khatouni, A., and Giordano, D. (2020). ERRANT: Realistic emulation of radio access networks. *Computer Networks*, 176:107289.

Ulusoy, A., Gurbuz, O., and Onat, A. (2011). Wireless model-based predictive networked control system over cooperative wireless network. *IEEE Transactions on Industrial Informatics*, 7(1):41–51.

Vick, A., Guhl, J., and Krüger, J. (2016). Model predictive control as a service — concept and architecture for use in cloud-based robot control. In *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 607–612.

Wan, J., Tang, S., Shu, Z., Li, D., Wang, S., Imran, M., and Vasilakos, A. V. (2016). Software-defined industrial internet of things in the context of industry 4.0. *IEEE Sensors Journal*, 16(20):7373–7380.

Xu, H., Yu, W., Griffith, D., and Golmie, N. (2018). A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access*, 6:78238–78259.

Yaqoob, I., Ahmed, E., Hashem, I. A. T., Ahmed, A. I. A., Gani, A., Imran, M., and Guizani, M. (2017). Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications*, 24(3):10–16.

Zhao, P. and Dán, G. (2018). A benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds. *IEEE Transactions on Network and Service Management*, 15(4):1460–1472.