FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators

Journal: IEEE Transactions on network and service management

DOI: 10.1109/TNSM.2022.3173436

Date of Publication: 9 May 2022

Authors: Milan Groshev (1), Jorge Martin-Perez (1), Carlos Guimaraes (2), Antonio de la Oliva (1), Carlos J. Bernardos (1)

(1) University Carlos III of Madrid,

(2) ZettaScale Technology SARL

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management

FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators

Milan Groshev*, Jorge Martín-Pérez*, Carlos Guimarães†, Antonio de la Oliva* and Carlos J. Bernardos*
*Universidad Carlos III de Madrid, Spain
†ZettaScale Technology SARL, France

***Abstract*—Wireless communications represent a game changer for future manufacturing plants, enabling flexible production chains, as machinery and other components not to be restricted to a location by the rigid wired connections on the factory floor. However, the presence of electromagnetic interference in the wireless spectrum may result in packet loss and delay, making it a challenging environment to meet the extreme reliability requirements of industrial applications. In such conditions, achieving real-time remote control, either from the Edge or Cloud, becomes complex. In this paper, we investigate a forecast-based recovery mechanism for real-time remote control of robotic manipulators (FoReCo) that uses Machine Learning (ML) to infer lost commands caused by interference in the wireless channel. FoReCo is evaluated through both simulation and experimentation in interference prone IEEE 802.11 wireless links, and using a commercial research robot that performs pick-and-place tasks. Results show that upon interference FoReCo reduces the trajectory error by more than a 34.35% in both simulation, and experimentation. We also show that FoReCo is sufficiently lightweight to be deployed in existing hardware.**

***Index Terms*—Robotic Manipulator, Wireless Remote Control, Machine Learning, IEEE 802.11, Interference.**

I. Introduction

Real-time remote control and coordination of robot manipulators over a wireless network are seen as the key enabler for future industrial applications [1], where a high level of flexibility, accuracy, data sharing, and cost reduction are desired. While wireless is a must for mobile robots like Autonomous Guided Vehicles (AGVs), the implementation of wireless connections for robots manipulators also has many advantages such as greater flexibility, reduction of installation and maintenance costs, ease of scale, and less personnel exposure to hazardous situations [2]. Industry 4.0 scenarios will decide whether to use wireless technologies in the licensed spectrum, such as 5G New Radio [3]; or in the unlicensed spectrum, such as IEEE 802.11 [4].

Nowadays, industrial verticals implement IEEE 802.11 technologies for factory automation through commercial solutions such as Industrial WLAN (IWLAN) developed by Siemens. The low cost, good performance (e.g., low latency, high throughput), and extensive implementation in commercial equipment make IEEE 802.11 a suitable candidate to fulfill the tight timing constraints of industrial automation. However, achieving the reliability, transparency, and stability for real-time remote control required in many applications remains a critical challenge in IEEE 802.11. due to the highly unpredictable, unreliable, and interference prone wireless channel,

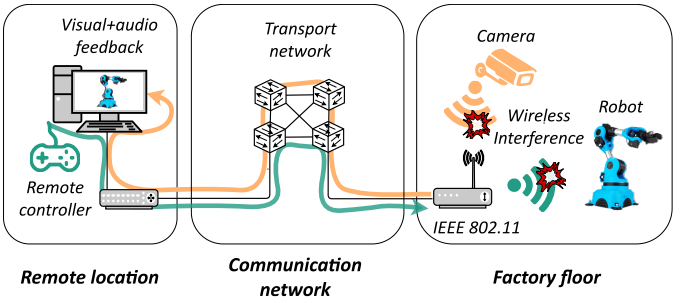


Fig. 1: Diagram of an industrial robotic remote control.

which introduces delays, packet loss, jitter, throughput bottlenecks, and even loss of connectivity [5]. The presence of packet collisions and electromagnetic (EM) interference in the shared medium results in delayed or even lost control commands. While the delayed delivery of control commands to the robot breaks the transparency of the remote control system (e.g., lag between the executed remote control commands and the robot movements), the lost commands directly influence the stability resulting in a deviation from the desired trajectory.

Mitigating the effect of delayed or lost commands in real-time remote control robots are not new, as it has been tackled through the lens of control theory in the robotics field [6]. However, most of these works do not consider the presence of EM interference, and assume that commands' delays are constant, following sums of normal distributions, or following first order Markov Processes. These assumptions do not consider the specifics of IEEE 802.11 Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) based Medium Access Control, as they do not capture how the underlying back-off, and re-transmissions impact the packet latency, nor the interference. Rather than making assumptions about the commands' delay and designing a delay-tolerant control loop, this paper shifts the focus and proposes a predictive control loop that infers the delayed or lost commands, and feeds them to the robot control loop.

In this paper, we propose FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators. FoReCo is suitable for autonomous or human-assisted remote control of robot manipulators that perform repetitive tasks such as welding, materials handling, picking, and packing, or assembly. In case the robot does not receive a remote control command on time due to IEEE 802.11 collisions or EM interference, FoReCo (*i*) infers the delayed

command; and (*ii*) injects it in the robot driver loop so the operator does not perceive misbehaviour in the remote control process.

This paper contributes to the state-of-the-art as follows:

- we formulate an optimization problem to minimize the trajectory error due to delayed and lost real-time remote control commands;
- we propose FoReCo to infer delayed and lost commands using Machine Learning (ML) algorithms;
- we validate FoReCo via simulation using an IEEE 802.11 analytical model [7] that accounts for an interference source, and packet collisions due to channel neighbors; and
- we show experimentally that FoReCo works in a commercial research robotic arm, and mitigates the effects of electromagnetic interference created with a real jammer.

In the remainder of this paper, we review the related work in §II, and formulate the problem statement in §III. Then, in §IV we present how FoReCo infers delayed/lost commands through ML. In §V we explain the analytical model of IEEE 802.11 that we use to test FoReCo in simulated scenarios with wireless interference. Later in §VI, FoReCo is validated via simulation and real experiments. Finally, §VII discusses the main insights from the obtained results, followed by conclusions and future directions in §VIII.

II. Related work

There is a rising interest in the networking community for providing support to Industry 4.0 cases in commercial deployments. The goal is always to meet the reliability that industrial processes require as it is in the case of remotely controlled robotic manipulators. Under the umbrella of the 5G-Public-Private-Partnership (5G-PPP) and the European Research Council (ERC), the European networking community has been focusing on how 5G and beyond 5G architectures can support Industry 4.0. 5G-DIVE [8] and 5Growth [9], are two examples of platforms that manage the adequate resource provisioning and allocation of services like a remote control in Industry 4.0. More recent European research projects [10] as Daemon [11] and Hexa-X [12] also provide support to Industry 4.0 applications by bringing intelligence to the network in order to meet strict constraints such as reliability.

In the case of teleoperation applications (i.e., applications providing remote control of systems), there is a plethora of work in the robotic and mechatronic literature on how to operate robotic manipulators. Reference [13] presents a prototype designed for the remote operation of an industrial robot manipulator using augmented reality, and [14] studies how to overcome, with the help of predictions, the collision of a robotic manipulator with objects due to remote operator errors. Works as [15] and [16] propose teleoperated robotic manipulators that assess complex and high precision tasks with the help of ML assisted solutions, and gravity compensation approaches, respectively. However, the robotic and mechatronic literature many times fails to consider the latency induced by the network in teleoperated/remotely-controlled systems. Indeed, none of the aforementioned works account for the network latency in the problem formulation, nor in the experimental stage, as authors control the robotic manipulator with a computer directly attached to the robot.

Works as [13]–[16] introduce errors in remotely-controlled robotic manipulators when applied in networks that suffer from high latencies or packet losses. Also, they cannot rely on network platforms like [8], [17], [11] and [12] to overcome such problems, as these platforms make a best-effort approach by means of network resource allocation and life-cycle management. That is, platforms as 5Growth [17] make the best to allocate network and computing resources for applications as [16], but they do not assist the remote-control application to recover when control commands are delayed or lost in the network.

Therefore, it is up to the remote-control application to decide how to react when control commands are delayed or lost. Mainly the robotic/mechatronic literature relies on control theory to overcome issues produced by delayed control commands. Works as [18]–[21] propose time domain passivity-based approaches, in particular [18] proposes to use a two-layer and a switching passivity-based [22] approach to deal with delays in the network. Other solutions [23]–[25] cope with the delay in remote-control using wave variable passivity-based approaches [26]. Another option is to resort to adaptive and robust control mechanisms to ensure the remotely-controlled robot stability upon delayed commands, as done in [27]–[30]. For example, [30] uses a Radial Basis Function Neural Network [31] based on Proportional Differential (PD) control [32] to mitigate the effect of external uncertainties and delayed commands in the robotic manipulator. However, the cited control-theory solutions in robotic/mechatronic literature takes unrealistic assumptions about the remote control commands' delays. In particular, [20] and [28] assume that the delay in the network is constant; [19], [21], [30], [23] and [24] assume that delays are constant or with small variations; and [18], [29], [25], and [27] take the causality assumption for the delay, i.e., the network delay cannot increase faster than time[1]. All of the aforementioned assumptions on network delay are not suitable for IEEE 802.11 wireless networks.

In this paper, we aim to improve the application and communication reliability by solving the problem from the networking perspective. We use command predictions in order to recover from the loss or delay of control packets. There are also works in the state of the art that follows a similar approach, in particular, [33] presents a control communication protocol that takes into account the wireless Signal to Noise Ratio (SNR) and uses a reinforcement learning [34] approach [35] to find the optimal speed of an AGV; and [36] proposes an AGV path tracking application using a Kalman filter to provide delay estimations for successful operation. However, [33] assumes that the success of the wireless transmission is captured by a first-order Markov process [37], and [36] assumes that the command delay in an IEEE 802.11 networks follows a Gamma distribution. Both assumptions neglect the presence of EM interference in the wireless channel, as well as the back-

[1]Following our notation, the causality assumption is expressed as: $|\Delta(c_{i+1}) - \Delta(c_i)| \leq |g(c_{i+1}) - g(c_i)|$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management

off and re-transmission mechanisms of IEEE 802.11 wireless channels that we investigate in this paper. Moreover, both [33] and [36] are solutions to enhance the reliability of real-time remotely controlled AGVs in wireless networks, rather than robotic manipulators.

The potential advantages from the wireless remote control of a robot manipulator are significant, but realizing such systems over an IEEE 802.11 network remains a challenging task. To the best of our knowledge, the state of the art disregards the presence of EM interference in real-time remote control and makes assumptions about the remote control commands' delay that do not apply to IEEE 802.11 wireless networks. To fill these gaps, we propose FoReCo, an ML-based solution that aims to minimize the robot trajectory error by predicting the missing remote control commands without making assumptions about the commands' delay.

III. Problem Statement

A remote control system in industrial environments generally consists of three main parts: *(i)* remote location; *(ii)* communication network; and *(iii)* factory floor (see Fig. 1). The remote site resides away from the factory floor, where a remote controller (fully autonomous or human-assisted) sends control commands to the factory robot in an open-loop fashion following a given frequency. Control commands which are sent every Ω [ms] have to transverse the transport network and wireless link to reach the robot. Additionally, the remote site contains visual and audio feedback that is streamed back to the remote location in order to provide similar conditions as those on the factory floor. While the control commands are very sensitive to packet losses and jitter, the visual and audio feedback can provide good quality of experience with up to 1 % of packet loss and 30 ms of jitter. Studying the visual and audio feedback over the inconsistent wireless channel and how it can impact the remote control of the robot is an interesting topic that is out of the scope of this work.

In this work considers an IEEE 802.11 wireless link, as its low price makes it an appealing solution for Industry 4.0. However, the unlicensed IEEE 802.11 spectrum leads to packet collisions, backoff times, and re-transmissions that introduce delay in the control commands. That is, since the moment a control command c_i is generated $g(c_i)$, up until the moment it is delivered to the robot $a(c_i)$, the transport network and wireless link introduce a delay $\Delta(c_i) = a(c_i) - g(c_i)$. Note that the latter is the addition of the delay introduced by the transport network $\Delta_T(c_i)$, and the delay introduced by the wireless link $\Delta_W(c_i)$, i.e.: $\Delta(c_i) = \Delta_T(c_i) + \Delta_W(c_i)$. In this paper, we make the following assumption on the transport network delay:

Assumption 1. *The delay introduced by the transport network $\Delta_T(c_i)$ is upper bounded by a constant D:*

$$\Delta_T(c_i) \leq D, \quad \forall i \tag{1}$$

Since each network entity (e.g., switch or router) in the transport network has finite queue sizes, the transport network can be modeled as a Jackson network [38]. Thus, we choose D as a constant higher than the summation of waiting times and processing time at each queue within the remote control path.

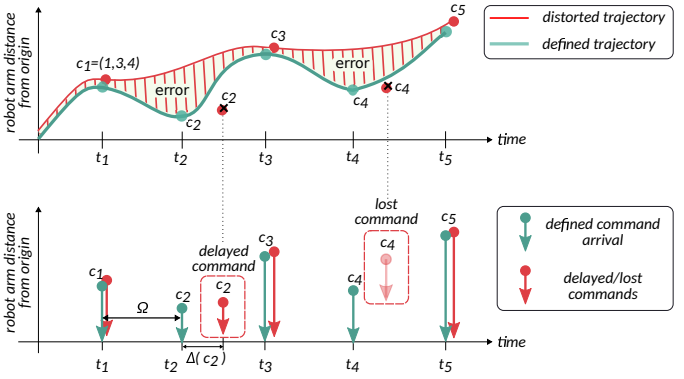


Fig. 2: Impact of delayed and lost commands in the robot trajectory.

However, even if D is very small, the delay introduced by the IEEE 802.11 link $\Delta_W(c_i)$ might lead to a laggy behavior in the remotely controlled robot. This means that the remote controller will experience a lag in between the time it moves the controller, and the time the robot moves. Since remotely controlled robots can only tolerate waiting for τ milliseconds to receive the next command, if $\Delta(c_i) > \tau$ the command c_i will exceed the tolerated delay, and the robot will not execute it. Thus, it is necessary that the control commands delays satisfy $\Delta(c_i) \leq \tau$.

Control commands are sent every Ω ms and the robot expects to receive those commands in the same interval (Ω ms) for smooth operation. However, due to the network delay the next control command c_{i+1} might not arrive until $\Omega + \Delta(c_{i+1})$ ms have passed.

Overall, the robot will not execute commands that arrive out of time $\Delta(c_i) > \tau$, or are lost $\Delta(c_i) \to \infty$. Upon any of these situations, the command is not executed, resulting in a deviation from the ideal trajectory that the robot should follow (see Fig. 2). Note that the remote controller will notice this error in the real trajectory via the visual feedback that it receives from the factory floor (see Fig. 1). Thus, it is necessary to recover the discarded packets to minimize the error in the real trajectory.

Problem 1. *Given the random variables $\Delta(c_i)$, a distance $d : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, a tolerance τ, and a record of the last R commands; find $f : \mathbb{R}^d \times \underbrace{\ldots}_{R-2} \times \mathbb{R}^d \mapsto \mathbb{R}^d$ to solve*

$$\min_{\hat{c}_N} \quad \lim_{N\to\infty} \frac{1}{N} \sum_{i}^{N} d(\hat{c}_i, c_i) \tag{2}$$

$$s.t. \quad \hat{c}_i = f\left(\{\hat{c}_j\}_{i-R}^{i-1}\right) \mathbf{1}_{\Delta(c_i)>\tau} + c_i \left[1 - \mathbf{1}_{\Delta(c_i)>\tau}\right], \forall i \tag{3}$$

With $\mathbf{1}_{\Delta(c_i)>\tau} = 1$ if command c_i is delayed more than τ ms, and zero otherwise. Problem 1 targets to find a function f to derive those commands that did not arrive on time. Additionally, the derived commands $\hat{c}_i$ should minimize the error (i.e., the distance $d(\hat{c}_i, c_i)$) with respect to the original command

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management

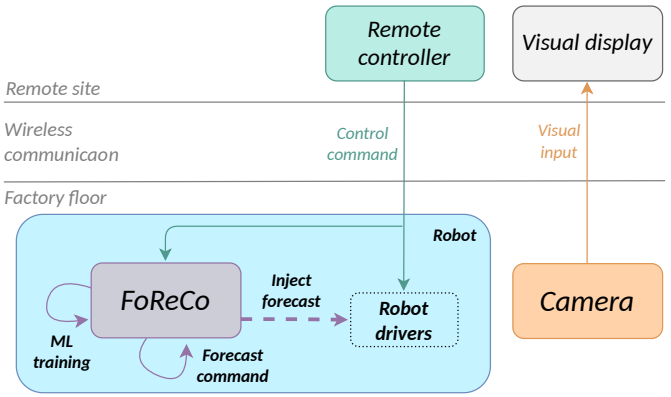


Fig. 3: FoReCo building block and remote control system.

c_i sent by the remote controller. That is, the objective of (2) is to minimize the dashed error region in Fig. 2 by using forecasts $f\left(\{\hat{c}_j\}_{i-R}^{i-1}\right)$ when a command c_i does not arrive on time $\Delta(c_i) > \tau$ (see (3)).

IV. Forecast-Assisted Remote Control (FoReCo)

In this section, we present FoReCo as a forecast-based recovery mechanism to minimize the trajectory error of remotely controlled robots via wireless connectivity.

A. The FoReCo Building Block

As discussed in §III, whenever the command delay exceeds the tolerance $\Delta(c_i) > \tau$, the robot considers the command c_i to be outdated and does not execute it. Depending on the robot, the absence of the command c_i may result in the robot stops, or keep feeding the prior command c_{i-1} to the robot control loop, which is implemented with solutions as Proportional-Integral-Derivative (PID) controllers (see [39]). Either way, the command c_i will not be executed and the robot trajectory will deviate from the expected, i.e., the trajectory executed by the remote controller. It is at this point that FoReCo predicts the command c_i that has not arrived on time and transparently triggers its execution into the robot. Hence, FoReCo stands as a complementary solution for any remotely controlled robot using a wireless link, while being agnostic to the implemented robot controller (control theory-based or not).

To predict control commands out of time, FoReCo follows an ML based approach, which has been proven to be effective with intention prediction and estimation of future trajectories of objects, such as vehicles, bikes, and humans. The learning model consist of predicting incoming control commands $c_i, c_{i+1}, c_{i+2}, \ldots$ with the help of the prior $c_{i-1}, c_{i-2}, \ldots$ commands. To do so, we advocate for an ML based methodology due to (*i*) the repetitive nature of the industrial tasks performed by remotely operated robots; and (*ii*) the difficulty to solve this problem with traditional dynamic programming algorithms.

Fig. 3 shows the conceptual components of the network control system we use to remotely control a robot (in-line with Fig. 1). The system shows the details of the interactions between the remote site and the factory floor over a communication channel. First, a real-time video stream of the robot is presented to a visual display over a wired communication channel. For simplicity, we assume that the uplink channel is error and delay-free and the video input is delivered to the remote operator immediately. The remote controller, with the help of the visual input, sends control commands over the wireless communication channel, and the commands are received by both the robot and FoReCo. With the received commands, FoReCo performs two actions:

1) **ML training**: to solve Problem 1, FoReCo resorts to ML to derive $f(\{c_j\}, \vec{w})$, with $\vec{w}$ being the weights to learn (see §IV-B). To obtain $\vec{w}$, FoReCo creates a dataset (see Fig. 6) with the commands it receives from the remote controller. The dataset contains a history of H commands, and FoReCo uses αH of them for training, and βH for testing; with $\alpha + \beta = 1$. As in Problem 1, the training procedure aims to minimize the distance between predicted commands $\hat{c}_i$, and the ones sent by the remote operator c_i. Hence, FoReCo trains its ML solution $f(\{c_j\}, \vec{w})$ s.t.:

$$\min_{\vec{w}} \frac{1}{\alpha H} \sum_{i}^{\alpha H} d\left(c_i, f\left(\{c_j\}_{i-R}^{i-1}, \vec{w}\right)\right) \tag{4}$$

   With the obtained weights $\vec{w}$, FoReCo tests the ML predictions accuracy in the testing set βH.
2) **Command forecast, validation and injection**: FoReCo awaits a control command c_i each Ω ms, and it triggers the forecasting if the next command c_{i+1} arrives latter than $a(c_i) + \Omega + \tau$. In this case, FoReCo will forecast the next command as $\hat{c}_{i+1} = f\left(\{\hat{c}_j\}_{i-R}^{i}, \vec{w}\right)$, i.e., using the ML solution f and the weights $\vec{w}$ obtained from the training stage. Next, FoReCo will validate the forecast by checking if the forcasted command offset is within the acceptable boundaries with respect to the current position of the robot. This validation is performed by FoReCo in order to prevent forecasts that can lead to an accident, malfunction, or robot misuse. The valid forecast command $\hat{c}_{i+1}$ is then injected in the robot drivers (as illustrated in Fig. 3) with the latter assuming that it received a command on time. In the case a command arrives on time $a(c_{i+1}) \leq a(c_i) + \Omega + \tau$, FoReCo will just store the command in the dataset and later use it for training and forecasting purposes. Note that we refer to $\hat{c}_i = c_i$ if the command arrived on time $\Delta(c_i) \leq \tau$, so it satisfies the constraint stated in (3). Thus, the forecasting receives as input $\{\hat{c}_j\}_{j-R}^{i}$ commands that arrived on time, and the forecasts of previous commands that did not arrive on time.

B. Studied ML Forecasting Algorithms

In the following, the selected ML algorithms used to implement FoReCo are described, as well as how they perform the command forecasting.

FoReCo is designed to forecast commands of remotely controlled robotic arms as the one in Fig. 1. Each command consists of d joints (remember $c_i \in \mathbb{R}^d$) that move together to shift the arm manipulator position, so the latter reaches the object of interest. Each command coordinate $c_i = (c_i^1, \ldots, c_i^d)$

represents the rotation angle or shift of a joint, depending on the joint nature.

The multivariate nature of the robotic arm suggests the use of vector autoregressive techniques as VAR. Appendix A details how the Box-Jenkins methodology points out that VAR is the most adequate model to infer future commands $\hat{c}_{i+n}$ of the robotic arm. On top of VAR, we have also investigated a seq2seq model to check if it can outperform the well-established Box-Jenkins methodology. Both solutions are compared against a moving average that serves as a benchmark for the command inference accuracy.

We acknowledge that not all ML forecasting algorithms are covered in this section. Nevertheless, in this paper we focus on some algorithms that are adequate and sufficient to outline the benefits of FoReCo, namely, we focus on the following algorithms:

- **Vector Autoregression (VAR)**: this regression solution is designed to predict multi-dimensional time-series with correlation across dimensions. This is the case of robotic arms, whose joint coordinates typically present correlation $\forall i, \exists k, m : c_i^k \sim c_i^{k+m}$; as they have to move together to reach and grab an object. VAR derives the prediction of command as follows:

$$\hat{c}_{i+1}^k = f^k\left(\{\hat{c}_j\}_{i-R}^i, \vec{w}\right) = b^k + \sum_{l=1}^{d}\sum_{j=i-R}^{i} w_{i,j}^l \cdot \hat{c}_j^l, \quad k \leq d \tag{5}$$

  with b^k being the bias for the k^{th} coordinate, $w_{i,j}^l$ the regression weights, and $f^k(\cdot)$ denoting the k^{th} coordinate of the resulting prediction. Both $b^k, w_{i,j}^l$ elements lie within the weight vector $\vec{w}$.
- **Sequence to sequence (seq2seq)** [40]: this ML solution is based on a Neural Network (NN) that receives as input a sequence and produces an output, that in our case is just a single output. These seq2seq models are known as many-to-one, as we feed it with a sequence of past commands $\{\hat{c}_j\}_{i-R}^i$ to produce a single one $\hat{c}_i$. The seq2seq architecture we use has: (*i*) an encoder layer of 200 Long Short-Term Memory (LSTM) neurons with Rectifier Linear Unit (ReLU) activations; (*ii*) and a decoder layer of 30 LSTM neurons, also with ReLU activations. The motivation behind the use of a seq2seq solution is to learn and encode the characteristics of robot movements, so the decoder layer "interprets" the encoded characteristics and guess the next command $\hat{c}_{i+1}$. Analytically, the seq2seq encoder and decoder layers are represented as follows:

$$e_i^k = \phi^k\left(W_0\hat{c}_i + W_1 a_{i-1} + W_2 m_i\right), \quad k \leq d \tag{6}$$

$$\hat{c}_{i+1}^k = f^k\left(\{\hat{c}_j\}_{i-R}^i, \vec{w}\right) = \\ = \phi^k\left(W_4 e_i^k + W_5 a'_{i-1} + W_6 m'_i\right), \quad k \leq d \tag{7}$$

  with $\phi(x)$ denoting the ReLU function[2], W_i denoting weight matrices (whose values are unrolled to derive the weight vector $\vec{w}$), a_{i-1}, a'_{i-1} being the output of the LSTM activation units, and m_i, m'_i referring to the memory cells of the encoder and decoder; respectively.
- **Moving Average (MA)**: we resort to this algorithm to have a benchmark for VAR and seq2seq. The MA derives the command prediction using:

$$\hat{c}_{i+1} = f^k\left(\{\hat{c}_j\}_{i-R}^i, \vec{w}\right) = \frac{1}{R}\sum_{j=i-R}^{i}\hat{c}_j \tag{8}$$

Note that FoReCo is flexible to support other forecasting algorithms, which can be integrated in a modular fashion.

C. Training of Selected ML Forecasting Algorithms

Next, we detail how FoReCo trains the selected ML algorithms described above.

- **VAR training**: to train the VAR algorithm, we resort to Ordinary Least Squares (OLS), i.e., the weights are computed as follows:

$$\vec{w} = \underset{\vec{w}}{\text{argmin}} \sum_{i}^{\alpha H}\sum_{k}^{d}\left(c_i^k - f^k(\{c_j\}_{i-R}^{i-1}, \vec{w})\right)^2 \tag{9}$$

  over the training portion of the dataset αH. Here $f^k(\cdot)$ refers to (5). Note that minimizing the summation in (9) is equivalent to minimizing the expression in (4), taking as distance $d(c_i, \hat{c}_i) = \sum_k \left(c_i^k - \hat{c}_i^k\right)^2$.
- **seq2seq training**: we train the seq2seq solution using Adam, a stochastic optimization method invariant to small gradients, as it is the case of our experimental study, e.g., a the 3rd robot joint takes values like $c_i^3 = 0.001$. We use Adam to iteratively minimize the error of the forecasts within a batch B_i of commands from the training set, i.e., $B_i < \alpha H$. Hence, at each training step Adam minimizes:

$$\min_{\vec{w}_t} \; l\left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t\right) = \sum_{i}^{\alpha H}\sum_{k}^{d}\frac{\left(c_i^k - f^k(\{c_j\}_{i-R}^{i-1}, \vec{w}_t)\right)^2}{B_i} \tag{10}$$

  with $l(\cdot)$ denoting the loss function, and $\vec{w}_t$ the updated weight vector at the step t of the training stage. Note that minimizing (10) is equivalent to minimizing (4) over the batch B_i, rather than the whole training dataset αH, taking the sum of squared distances. At each step the weights are updated as follows:

$$\vec{w}_{t+1} = \vec{w}_t - \eta\frac{m_{t+1}}{1-\beta_1^{\alpha H}}\frac{1}{\sqrt{\frac{v_i}{1-\beta_2^{\alpha H}}} + \varepsilon} \tag{11}$$

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla_{\vec{w}} l\left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t\right) \tag{12}$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\left[\nabla_{\vec{w}} l\left(\{c_j\}_{j=0}^{B_i}, \vec{w}_t\right)\right]^2 \tag{13}$$

  with m_t, v_t being the estimates of the first and second moment of the loss function gradient $\nabla_{\vec{w}} l(\cdot)$, η the step size, and $\beta_1, \beta_2, \varepsilon$ other hyper-parameters.

V. IEEE 802.11 with Electromagnetic Interference

So far we have discussed how FoReCo works in §IV-A, and the ML solutions that we consider to assess the command

[2] $\phi(x) = 0, x \leq 0$ and $\phi(x) = x$ otherwise

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management

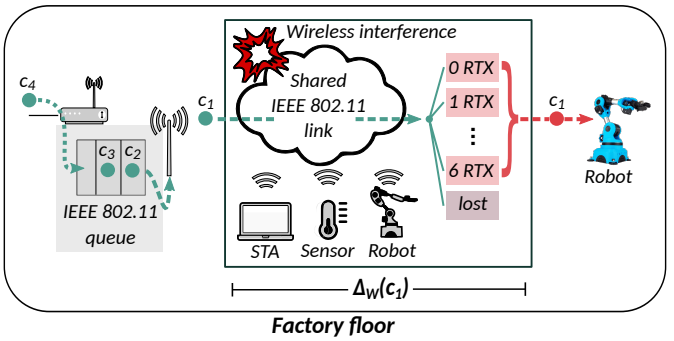


Fig. 4: Impact of wireless interference, retransmissions (RTX), and factory devices in the delay $\Delta_W(c_i)$ that control commands experience in an IEEE 802.11 link.

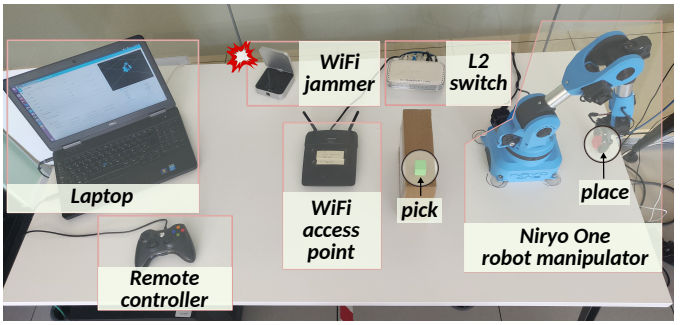


Fig. 5: Testbed setup.

forecasting in §IV-B and §IV-C, respectively. In this section we explain the analytical model we consider to derive the delay that control commands experiment $\Delta_W(c_i)$ in IEEE 802.11 wireless links under EM interference. The analytical model is latter used in §VI-C to derive the $\Delta_W(c_i)$, and assess the performance of FoReCo in a simulated scenario as close as possible to real IEEE 802.11-based real-time remote control.

In this paper, we resort to the analytical model presented in [7] to derive wireless delays. This work models the MAC layer of IEEE 802.11 with CSMA/CA, and studies how neighboring nodes and non-IEEE 802.11 interfering sources impact the wireless delay. The work is based on a refinement [41] of Bianchi's characterization of IEEE 802.11 [42]. The particularity is that [5] extends the underlying Markov chain to also capture the presence of an interference source that is active during T_{if} transmission slots, and emits with a probability p_{if}. The proposed model also captures both the back-off mechanisms and re-transmissions (RTX) of frames upon collision in the IEEE 802.11 wireless link.

With the aforementioned model, [7] obtains the steady-state vector of each state, in particular, they derive the probability that a frame has to be transmitted after j unsuccessful re-transmissions, which is denoted as a_j. Moreover, [7] also derives $\mathbb{E}_j[\Delta_W(c_i)]$, that is, average delay that the command c_i experiences in the wireless transmission after j unsuccessful re-transmissions. Based on such expression, we derive in the Appendix some theoretical results around the analytical model given in [7], that give some insights about the delay of control commands. In particular, the theoretical results in the Appendix conclude that in the considered IEEE 802.11 scenario:

- **i)** $\Delta(c_i)$ is only bounded on average, but not always (see Lemma 1);
- **ii)** $\Delta(c_i)$ diverges (see Corollary 1); and
- **iii)** the causality assumption does not apply (see Corollary 2).

Hence, the delay assumptions taken in the solutions presented in §II do not hold. In other words, we cannot bound the delays that the remote control commands c_i are experiencing. Still, we resort to the analytical model presented in [7], as such unbounded delay behaviors are realistic in IEEE 802.11 scenarios upon the presence of interference sources.

To derive the value of $\Delta_W(c_i)$ we follow [7] and model the transmission of control commands c_i over IEEE 802.11 wireless links as a queuing model. From the problem statement formulation presented in §III, we know that control commands have an arrival rate $\frac{1}{\Omega}$. These commands are queued in the IEEE 802.11 access point before they are transmitted to the shared wireless link. Following the IEEE 802.11 standard, a frame is re-transmitted up to 7 times. After this threshold is exceeded, the frame (and therefore, the control command) is assumed to be lost and no further re-transmission is executed (see Fig. 4).

Depending on the number of RTX, the control command delay $\Delta_W(c_i)$ will be higher or lower. This system behaves as an $G/HEXP/1/Q$ queuing model, with Q being the length of the access point queue, and the service rates of the hyperexponential distribution corresponding to the average delay that control commands see after j RTX, i.e., $\frac{1}{\mathbb{E}_j[\Delta_W(c_i)]}$.

Given this $G/HEXP/1/Q$ queuing model, we can derive $\Delta_W(c_i)$ in the desired IEEE 802.11 wireless scenario accounting for the number of transmitting devices and the probability and time that the wireless interference is active. These are the delay values used in the simulation scenarios in §VI-C, and we derive them using the CIW discrete event simulation library [43].

VI. Results

In order to evaluate FoReCo, we consider a realistic industrial application where a robot manipulator is remotely controlled to perform a pick and place task. This remote control application allows us to select the most suitable forecasting algorithm for FoReCo (see §IV-B), and to evaluate a prototype implementation of FoReCo under simulation (see §VI-C) and experimental scenarios (see §VI-D). It is worth mentioning that in this section we compare the performance of FoReCo with the baseline Niryo controller mainly because of the lack of state-of-the-art remote controllers for robot manipulators. We leave the comparison to more recent AVG approaches, e.g., learning-based AVG controllers [33], for follow-up work, where the AVG controllers need to be adapted for robot manipulators in order to employ their benefits.

A. Testbed setup and dataset collection

Fig. 5 shows the experimental testbed, built in the 5TONIC laboratory[3] that is composed of: a 6-axis Niryo One robotic

[3]https://www.5tonic.org

manipulator, a 2.4 GHz IEEE 802.11 access point (AP), an L2 switch, a 2.4 GHz Silvercrest Wireless Transmitter that is used as WiFi jammer, and a joystick that is connected to a laptop with 8GB RAM and 4CPU@2.4GHz via USB. The communication between the robot and the joystick is comprised of a wireless link from the robot to the AP and an Ethernet link from the AP to the laptop using the L2 switch. The Niryo One robotic arm is equipped with a Raspberry Pi 3 with a 1.2 GHz 64-bit CPU, 1GB RAM, and an IEEE 802.11n interface. The robot maximum speed is 0.4 m/s for the steeper axes and 90°/s for the servo axis with a configured Robot Operating System (ROS) control command frequency of 50Hz (i.e., $\Omega = 20$ ms) and command moving offset of 0.04 rad. The remote control and robot system is ROS version 1. The Niryo One ROS stack expects to receive control commands each Ω ms and considers that a packet did not arrive on time otherwise, i.e., the tolerance is $\tau = 0$. Niryo One ROS stack uses the prior command $\hat{c}_{i+1} = c_i$ in case $\Delta(c_{i+1}) > \Omega$. Every received command is passed to the Niryo motion planning layer (MoveIt), which uses Proportional–Integral–Derivative (PID) control.

Fig. 6 shows part of the dataset created by performing pick and place actions. The pick and place actions were manually repeated 100 times by two different human operators, an experienced and inexperienced human operator resulting in the creation of two separate datasets. To do so, they used the joystick as a remote controller, issuing a new control command every 20 ms. The inexperienced/experienced operators' datasets' contain $H = 187109$ commands. Both datasets store the joint states c_i of the robot manipulator under ideal network conditions, i.e., low latencies and absence of packet collision. To achieve such conditions, the datasets were obtained using Ethernet to send the remote controller commands. The experienced dataset was used to train the ML models while the inexperienced data was used for remote control and testing. In this way, we ensure that the trained ML model operates on data that is tightly related but not the same as the training data.

B. Forecasting accuracy

We now evaluate which of the selected forecasting algorithms achieves the highest forecasting accuracy in the collected datasets. The VAR algorithm was implemented using statsmodel v0.12.1, and seq2seq using Tensorflow 2.1.0. Fig. 7 shows the RMSE accuracy of each algorithm as we increase the forecasting window, i.e., how many consecutive commands are forecasted (a command is sent each $\Omega = 20$ ms). For every algorithm, we considered a record of the last $R = 1, \ldots, 20$ commands, and Fig. 7 plots the best-performing R parameter for each algorithm. For the training stage (see §IV-C), we used the $\alpha = 80\%$ of the experienced human operator data for training, and a $\beta = 20\%$ of the inexperienced operator data for testing. It is worth mentioning that we performed the 80%-20% split of the datasets only for testing purposes, mainly because of the repetitive nature of the movements, whereby using 20% of inexperienced operator data is sufficient to give us insides of the most accurate model.

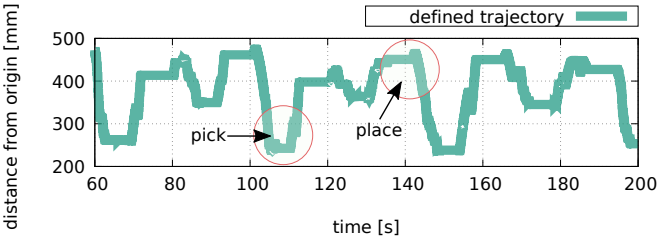


Fig. 6: Robot trajectory dataset with pick and actions of an inexperienced operator.

Later, when we evaluate the models in simulation and experimentation, we use 100% of the experienced human operator data for training, and 100% of the inexperienced operator data for validation. For seq2seq we resort to the standard hyper-parameter selection: $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-07$.

Results show that VAR has slightly better accuracy than MA, while seq2seq has the worst performance. It was expected that VAR would outperform MA since it is designed for correlated time-series – like the 6-axis time-series of the Niryo One robotic arm. However, seq2seq yielded worse accuracy than MA due to the vast number of weights $|\vec{w}| = 163803$ to learn, thus, it did not converge to an optimal solution. Given the results in Fig. 7, we use the MA and trained VAR solution as forecasting techniques in our simulation analysis presented in §VI-C.

C. Simulation evaluation

In the following, we evaluate how FoReCo behaves under a simulated environment with wireless interference. We consider a transport network with negligible transport delay, i.e., $D \simeq 0$ ms in *Assumption* 1, thus, commands' delays are dominated by the wireless delay $\Delta(c_i) \simeq \Delta_W(c_i)$. To derive $\Delta_W(c_i)$, we resort to an analytical model of IEEE 802.11 with non-IEEE interfering sources [7], and use the parameters reported in [7, Table 2]. The goal of the simulation validation is two-folded: (*i*) evaluate the precision of the forecasted commands by FoReCo, and (*ii*) assess the scalability with up to 25 robotic arms sharing a wireless medium with interferences. All the details about the simulation implementation of FoReCo and

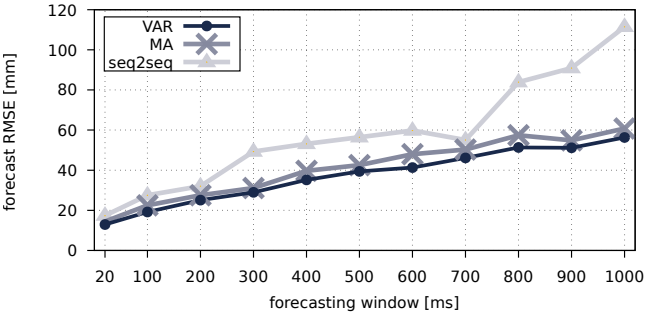
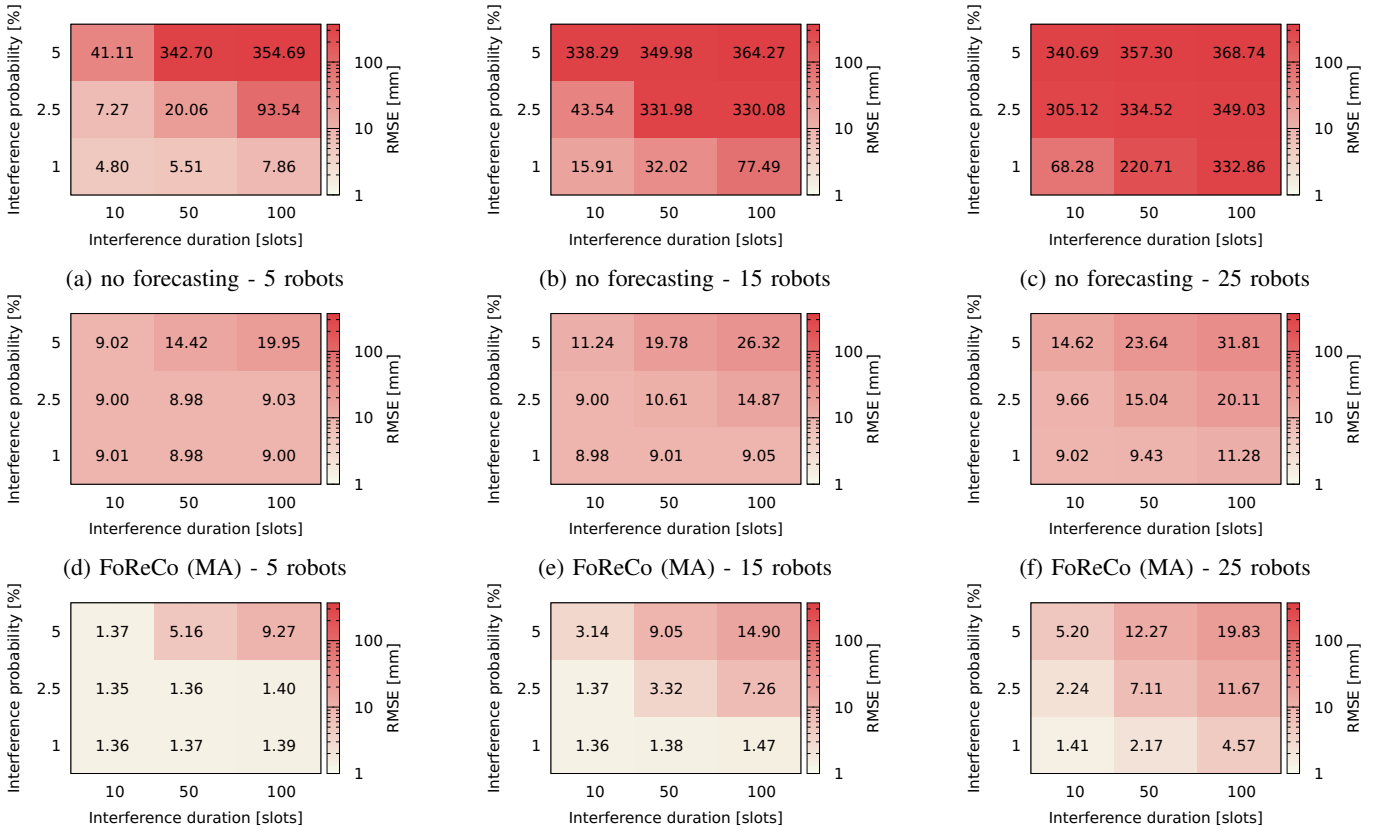


Fig. 7: Forecast accuracy for different forecasting windows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management



(a) no forecasting - 5 robots (b) no forecasting - 15 robots (c) no forecasting - 25 robots

(d) FoReCo (MA) - 5 robots (e) FoReCo (MA) - 15 robots (f) FoReCo (MA) - 25 robots

(g) FoReCo (VAR) - 5 robots (h) FoReCo (VAR) - 15 robots (i) FoReCo (VAR) - 25 robots

Fig. 8: Robot trajectory error upon interference without forecasting (top), with FoReCo using MA (middle), and FoReCo using VAR (bottom).

the IEEE 802.11 analytical model can be found in our publicly available git repository[4].

Each simulation issues the commands of an inexperienced human operator and introduces command delays $\Delta_W(c_i)$ following [7]. Fig. 8 shows the error experienced by the robot trajectory. Fig. 8 (top) shows the results using the state-of-the-art solution, i.e., repeating the prior command $\hat{c}_{i+1} = \hat{c}_i$ upon delays. Fig. 8 (middle) shows the results when FoReCo recovers packets using the MA solution specified in (8), and Fig. 8 (bottom) shows the results when FoReCo uses the VAR solution to recover packets – as specified in (5). Since the introduced wireless delay $\Delta_W(c_i)$ is a random variable, we repeat each simulation 40 times. Note that, in each simulation, we vary the time and probability of the active interference. Each square in the Fig. 8 heatmap illustrates the averaged RMSE of the 40 simulations done for every pair of interference duration, and probability. The RMSE is computed over the entire robot trajectory induced by the inexperienced human operator, and it considers commands arriving on time $\Delta(c_i) \leq \tau$ and out of time $\Delta c_i > \tau$, without using control command forecasting (top in Fig. 8), and with FoReCo using MA and VAR (middle, and bottom rows in Fig. 8; respectively).

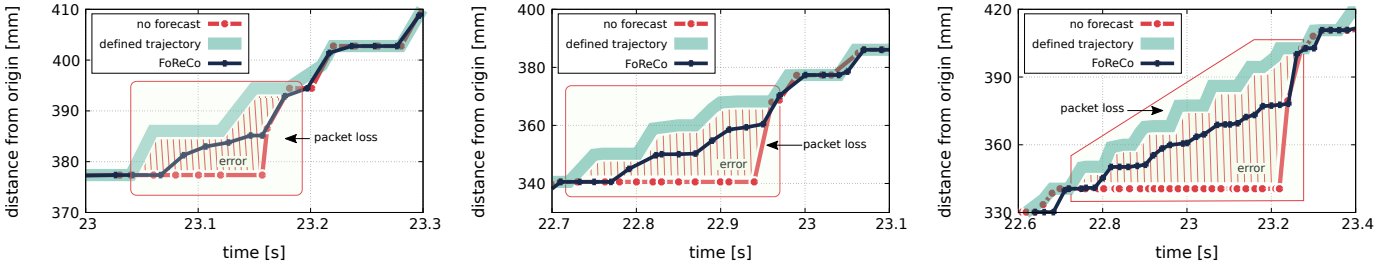
The RMSE error in Fig. 8 is represented in logarithmic scale, and we can appreciate that FoReCo command recovery constrained the robot trajectory error below 19.95 mm, 26.32 mm and 31.81 mm using MA (middle row) and 9.27 mm, 14.90 mm and 19.83 mm using VAR (bottom row) for 5, 15 and 25 robots on the factory floor, respectively. Fig. 8 shows that the VAR solution outperforms the MA solution in every simulation scenario for approximately 10 mm. On the other hand, the no forecasting solution resulted in an RMSE in the order of ~ 350 mm in the worst cases, no matter the number of robots. Thus, simulations indicate that (*i*) the VAR solution outperforms the MA solution by minimizing the error for additional 10 mm; (*ii*) FoReCo based on VAR will not exceed errors of 20 mm; and (*iii*) FoReCo reduces the experienced error by more than one order of magnitude. In particular, FoReCo using VAR reduces by more than a 94.4% (368.74 mm with no forecasting and 19.83 mm with FoReCo (VAR)) the experienced error in factory floors of 25 robots

D. Experimental evaluation

Motivated by the simulation results, we implemented and integrated a prototype of FoReCo that is based on VAR within the real remote control system presented in Fig. 5. The prototype is following the principles defined in §IV, and uses ROS to interact with the remotely controlled Niryo One robotic arm. All the details about the prototype implementation of FoReCo can be found in our publicly available git repository[4].

[4]https://gitlab.it.uc3m.es/5g-team/FoReCo

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3173436, IEEE Transactions on Network and Service Management



(a) 5 consecutive losses (b) 10 consecutive losses (c) 25 consecutive losses

Fig. 9: Robot trajectory with controlled packet losses using no forecasts, and FoReCo.

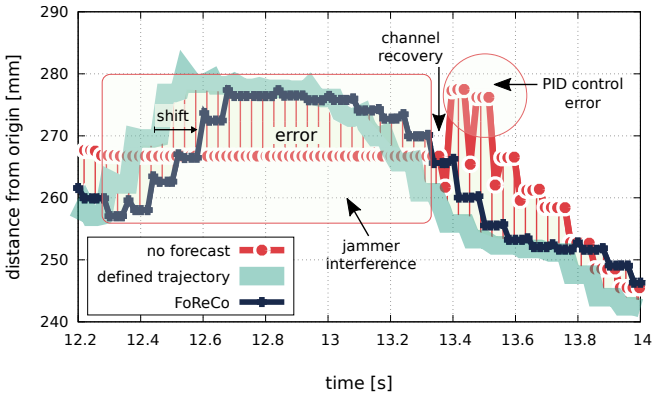


Fig. 10: Robot trajectory upon IEEE 802.11 jammer interference

1) Controlled experimental evaluation: In our first experimental analysis, we manually introduced loss of control commands in order to evaluate the improvements that the prototype offers under a controlled environment. The complete dataset from the inexperienced user was used to develop a remote controller that was randomly dropping consecutive control commands. Every time consecutive control commands were lost, FoReCo injected predictions from the VAR model. We executed 3 different sets of experiments, where the remote controller randomly introduced 5, 10, or 25 consecutive losses. Each experiment run was 30 seconds, with the robot joint states being recorded in the robot itself.

Fig. 9 shows the trajectories followed by the robot arm when consecutive control commands were lost for the case of no forecasts, and the FoReCo solution. The results show that FoReCo mitigates the negative effects of lost commands reducing the trajectory error. Namely, FoReCo reduces the RMSE a 34.35%, 52.74%, and 56.31% upon 5, 10, and 25 consecutive losses, respectively. Note that the benefits are more noticeable as the number of consecutive losses increase, since FoReCo prevents the robot from being still for long periods. However, Fig. 9(c) also shows that FoReCo deviates more and more from the defined trajectory as the number of consecutive losses increase (in between second 23 and 23.2), since VAR builds its forecasts $\hat{c}_{i+1}$ using prior forecasted commands $\hat{c}_i, \hat{c}_{i-1}, \ldots, \hat{c}_{i-R}$ Thus, the prediction error propagates – see (5).

2) Jammed experimental evaluation: In our second experimental analysis, we emulate a realistic interference scenario where the Silvercrest Wireless device is used to transmit synchronized radio waves in the same frequency as the robot, introducing unpredictable network delays $\Delta(c_i)$ and packet losses – same as the analytical model [7] used in prior simulations in §VI-C. The dataset from the inexperienced user was used in order to develop a remote controller that sent the control commands over the jammed wireless network. Every time control command(s) were lost or delayed due to the interference $\Delta(c_i) > \tau$, FoReCo injected predictions from the VAR model. The experiment was executed for 30 seconds and in the robot, we recorded the robot joint states.

Fig. 10 presents the trajectories followed by the robot arm when the wireless channel was interfered by the jammer. The results show how FoReCo reduces by more than 50% the trajectory error RMSE (from 18.91 mm to 8.72 mm) – compared to the bare Nyrio One solution without forecasting. In the interval between 12.4 and 12.6 seconds, FoReCo starts to deviate (shift) from the defined trajectory in an identical way to the controlled experiments described above. In addition, an interesting observation is how the robot trajectory behaves upon channel recovery. In the latter case, the Niryo One ROS MoveIt PID controller takes around 400 ms to stabilize the trajectory (from second 13.5 to 13.8), since the PID controller receives repeated commands $c_{i+1} = c_i, \forall i \in [12.4, 13.4]$ during more than a second, resulting in the highlighted error in Fig. 10.

3) Training and inference times: Since Niryo One is equipped with a Raspberry Pi3, which is limited in terms of computing power, we executed a set of experiments to measure both training and inference times of VAR whenever executed in the robot itself. While training takes around 5.99±0.06 min to train using the experienced human operator dataset, the inference (i.e., prediction of the commands) only takes 1.60±0.16 ms. These results show that the current hardware of the robot is sufficient to not only accommodate FoReCo within the constraints of our prototype but also to support the implementation of more stringent applications control loops. Note that training is only required when we need to build or update the model, which time profiling is presented in Table I.

For comparison, Table II also presents training and inference times in different equipment: *(i)* NVIDIA Jetson Nano which can be co-located with Niryo One; *(ii)* a laptop equipped with a 2nd gen Intel Core i7 and 6GB RAM representing the user equipment; and *(iii)* a local server with two Intel(R) Xeon(R) CPU E5-2620 v4@2.10GHz and 64GB RAM

TABLE I: Time profiling of FoReCo training in Niryo One

	Load Data (s)	Down Sampling (s)	Check Quality (s)	Training Model (s)
Raspberry Pi3 (Robot)	1.95 ± 0.02	0.26 ± 0.007	306.38 ± 3.15	50.98 ± 0.54

representing the Edge to offload training and inference tasks.

TABLE II: Training and inference times in different equipment

	Training (min)	Inference (ms)
Raspberry Pi3 (Robot)	5.99 ± 0.06	1.60 ± 0.16
NVIDIA Jetson Nano (Robot)	1.31 ± 0.01	0.61 ± 0.28
Laptop (UE)	0.36 ± 0.01	0.22 ± 0.10
Local Server (Edge)	0.23 ± 0.007	0.0001 ± 0.00003

VII. Discussion

In this section we discuss the results, analyzing how they can be interpreted from the perspective of real-time wireless networked control systems. In addition, possible future directions are identified.

A. Overall performance and applicability

In experimental scenarios §VI-D with a single robot FoReCo reduces by a 34.35% the trajectory error upon the presence of up to 25 slots of interference. Moreover, the simulation results §VI-C show that FoReCo reduces the trajectory error up to a 94.4% with 25 robots sharing the IEEE 802.11 channel, and up to 100 consecutive interference slots. That is, FoReCo's benefits are more evident upon worse wireless conditions, either due to simultaneous robots transmitting in the wireless medium, or due to longer interference periods. The reason is that packet collisions and delayed/lost commands increase with the number of robots in the channel, and the robotic arm remains still for longer periods as the consecutive losses increase. Thus, FoReCo recovers more delayed/lost commands and prevents the robot from stopping.

Results show that FoReCo can achieve high precision and suggest that industrial manipulation applications (e.g., assembly, pick-and-place) can benefit from adopting the proposed method. Results also demonstrate that the prototype of FoReCo can be easily attached to robot manipulators without robot-specific modifications. Furthermore, results show how predictive control can improve the reliability of real-time remote control over the IEEE 802.11 wireless network.

B. Coexistence with emerging wireless technologies

Existing industrial wireless technologies cannot meet all of the remote control requirements (such as 99.999% of reliability, 2-20ms latency, 100-200 Mbps data rate). Although emerging technologies such as WiFi 6E and 5G are expected to be a key enabler in this regard by increasing the levels of reliability in industrial wireless, the uncertain and time varying wireless channel continues to be an issue for wireless technologies. The fact that FoReCo does not make any assumptions about the wireless channel and the network delays makes it applicable also to emerging wireless such as mmWave or 5G. In addition, it is very likely that legacy systems will leverage previous technologies which will operate for years to come, and FoReCo can offer them the needed reliability for running remote control applications.

C. Real-time Path Tracking Predictions

Concerning the ML algorithm for repetitive reference trajectories, given the performance of VAR, our future work will consider other forecasting algorithms as exponential smoothing methods. We will also investigate other ML approaches, such as classification algorithms, to infer the type of command that comes next. It is worth noting that, although VAR performed well on the pick-and-place dataset, a deviation from the defined trajectory is witnessed when the number of consecutive commands losses increases. Future versions of FoReCo may mitigate large periods of consecutive losses by incorporating delayed commands that did not arrive on time, i.e., commands c_i with $\Delta(c_i) > \tau$ could be used instead of the predicted one $\hat{c}_i$, to infer commands after $n\Omega$ ms. In other words, based on (3) we could use $\hat{c}_n = f(\hat{c}_{n-1}, \ldots, c_i, \ldots)$.

D. Predictions at the edge of the network

In addition to improving the model accuracy and precision, an edge-based version of FoReCo can be considered where the VAR algorithm will always base its predictions on the real control commands $\hat{c}_i = f(\{c_j\}_{i-R}^{i-1})$. However, this approach is a bit more disruptive because an edge-based version of FoReCo indicates that the predictions will need to traverse the interfered wireless channel. Hence, FoReCo will need to piggyback the predictions together with real-time control commands. Piggybacking predictions require modifications of the robot drivers to use them whenever a control command does not arrive on time. We leave such an option for future work.

E. An extra resilience layer

As stated in §II, FoReCo targets a solution from the network perspective, in opposition to already existing related work. However, the two approaches are not mutually exclusive and, consequently, they can be used together to provide even higher levels of reliability and precision when performing remote control of robotic manipulators. In doing so, in a full-fledged solution, several recovery mechanisms can be envisioned as a resilience stack where all layers work together in a fail-over fashion.

F. Extension to other robotic systems

Although FoReCo explicitly targeted robotic manipulators, the solution described in this work can be easily extended to many other robotic systems that implement open- or close-loop between themselves and the remote controlling system. For example, remotely controlled AGVs within the manufacturing plant. Similar to the robotic manipulators, missing commands are predicted and injected into the AGV so that it can smoothly follow its trajectory. Still, FoReCo requires that performed

tasks are somehow periodic so that the corresponding dataset can be created, and training and inference tasks executed with acceptable levels of precision.

VIII. Conclusions

This paper presents FoReCo, a forecast-based recovery mechanism for real-time robot remote control. The FoReCo prototype uses VAR and ROS, and its performance has been assessed in a commercial research robotic arm remotely controlled over IEEE 802.11 wireless channels under the presence of interference. We also validate FoReCo through simulation and assess its performance in a real testbed. Results show that FoReCo provides high precision by achieving a trajectory error below 19.83 mm in simulation, and of 8.72 mm in experimentation with a commercial research robotic arm and jammed interference. Overall, FoReCo reduces the trajectory error by more than a 34.35% in every experimental and simulated scenario. Moreover, FoReCo is lightweight and can be deployed in the hardware already available in several solutions.

As follow-up work, we plan to (*i*) integrate more ML forecasting algorithms; and (*ii*) adapt solutions of wireless controlled AGVs [36] [33] to remotely controlled robotic arms, and compare their performance against FoReCo in IEEE 802.11 wireless channels.

Acknowledgment

This work has been partially funded by European Union's Horizon 2020 research and innovation programme under grant agreement No 101015956, and the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-EDGEDT and 6G-DATADRIVEN

Appendix

A. Box-Jenkins methodology

In this appendix we show how we applied the Box-Jenkings methodology [44] to decide that VAR with autoregression (AR) order of 10 was the most appropriate model to infer future robot commands $\hat{c}_{i+n}$. In the following, we apply the three stages of the Box-Jenkins methodology in our problem:

1) **Model identification**: the robot trajectory time series is *i*) stationary; *ii*) non-seasonal; *iii*) with causality; and *iv*) co-integration across the robot axis. Hence, Vector Autoregressive Moving Average (VARMA) is an adequate method to infer future commands – see [45]. Below we detail why properties *i*)-*iv*) are satisfied:
   i) the time series is stationary because the null-hypothesis (non-stationary) of the Augmented Dickey-Fuller test [46] is rejected with significance level $\alpha = 0.05$. Actually, the test yielded p-values in the range of $p \in [-3.71 \cdot 10^{-13}, -1.52 \cdot 10^{-26}]$ for every robot axis;
   ii) the time series is not seasonal, for the robot operator sometimes takes more time to repeat the pick and place tasks, and all the actions are not exactly the same, i.e., $\nexists T : c_i = c_{i+nT},\ \forall n \in \mathbb{N}$. Moreover, the Partial Autocorrelation Function (PACF) values – see Fig. 11 – drastically drop to zero and do not fall outside the 95% confidence interval. Hence, there is no evidence of seasonality;
   iii) each robot axis Granger-causes [47] the others $c_i^l \sim c_i^{l'},\ \forall l, l'$ because the null-hypothesis (no Granger-causality) of the Granger-causality test was rejected with significance level $\alpha = 0.05$. Indeed, the test yielded p-values $p \leq 0.0144$ when it checked Granger-causality between every pair of axis;
   iv) the robot-axis time series are co-integrated, i.e.:

$$\forall l, \exists \{a_i^l\}_i^{T_l} : \sum_l \sum_i^{T_l} c_i^l a_i^l \sim I(m), \quad m \leq \min_i \{d : c_i^l \sim I(d)\} \tag{14}$$

   with $c_i^l \sim I(d)$ meaning that time series of axis l has differentiation order d. In other words, there is a linear combination of the robot axis time series with order of integration below the order of integration of each robot axis individually. In our experiments we used the co-integration Johanson's test [45], and proved that (14) holds because we sequentially rejected the null hypotheses of having $k = 0, k \leq 1, \ldots, k \leq 5$ co-integrating vectors with significance level $\alpha = 0.05$.
2) **Parameter estimation**: after identifying VARMA as a suitable model, we have to estimate its parameters for our robot dataset. VARMA considers that future commands $\hat{c}_{i+n}$ are expressed using prior commands $c_{i-n},\ n \in \mathbb{N}$; and that regression residuals ε_{i+1} equal a moving average of prior residuals $\varepsilon_{i-n},\ n \in \mathbb{N}$. Following §IV-B notation,

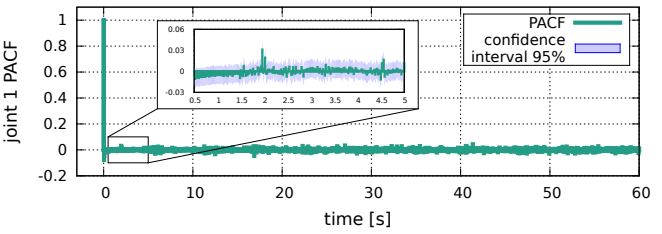


Fig. 11: PACF of joint 1 over time. Instants outside the confidence interval suggest possible seasonality.

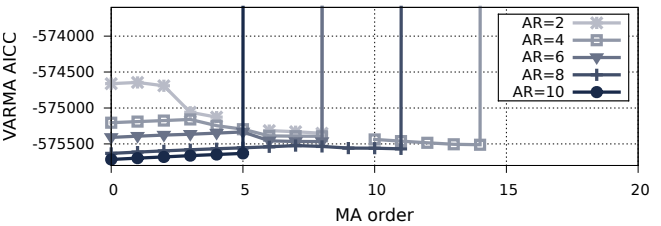


Fig. 12: VARMA accuracy using different AR and MA orders.

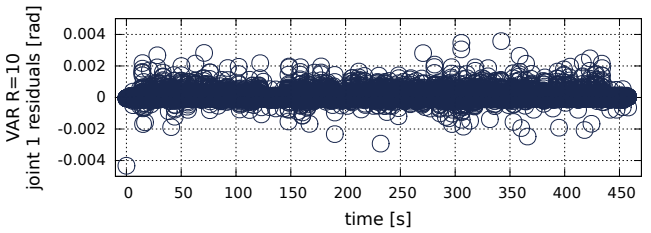


Fig. 13: Joint 1 residuals for VAR with AR order $R = 10$.

VARMA looks as follows:

$$\hat{c}^k_{i+1} = f^k\left(\{\hat{c}_j\}^i_{i-R}, \vec{w}\right) = b^k + \sum_{l=1}^{d}\sum_{j=i-R}^{i} w^l_{i,j}\cdot\hat{c}^l_j + \varepsilon^k_{i+1} + \sum_{l=1}^{d}\sum_{j=i-M}^{i} w'^l_{i,j}\cdot\varepsilon^l_j, \quad k \le d \tag{15}$$

with $\vec{w} = \{\{w^l_{i,j}\}_{i,j,l}; \{w'^l_{i,j}\}_{i,j,l}\}$ the weights for the VARMA model, R the autoregression (AR) order, $w'^l_{i,j}$ the MA weights, and M the MA order.

To select the best parameters (R, M), we have trained different VARMA models with $1 \le R \le 10$, $0 \le M \le 10$; and compared their qualities using the Akaike Information Criterion Corrected (AICC) [48], as authors of [49] do. Figure 12 illustrates the AICC achieved with different combinations of the AR and MA order. Results show that only the least accurate models benefit from increasing their MA order. Whilst the best models, with higher AR order, only get worse as the MA order increases.

From Fig. 12 we conclude that $(R = 10, M = 0)$ is the most accurate model, i.e., the VAR model that we use throughout the paper.

3) **Statistical model checking**: to check the validity of VAR with $R = 10$ we run the Ljung-Box test [45], with null-hypothesis of no correlation across the residuals of the fitted VAR model. We could not rejected the null-hypothesis because the obtained p-values where $p > 0.99$ for every axis. Therefore, the residuals of VAR with $R = 10$ are not correlated. Moreover, despite some outliers, the residuals of all axis maintain a constant mean and variance over time – see[4] Figure 13. Hence, Box-Jenkins methodology [44] suggests that VAR with AR order $R = 10$ is an adequate estimator of future commands $\hat{c}_{i+n},\ n \in \mathbb{N}$.

B. IEEE 802.11 analytical insights

In the following, we present some theoretical results about the expected delay of a command, and the causality assumption in the IEEE 802.11 scenario considered in this paper.

Lemma 1. *A control command c_i traversing a transport network, and an IEEE 802.11 wireless link under interference will experience an average delay satisfying*

$$\mathbb{E}[\Delta(c_i)] \le D + \frac{1}{1-a_{m+2}}\sum_{j=0}^{m+1} a_j \cdot \mathbb{E}_j[\Delta_W(c_i)], \quad \forall c_i \tag{16}$$

with probability $1 - a_{m+2}$, and

$$\mathbb{E}[\Delta(c_i)] = \infty, \quad \forall c_i \tag{17}$$

with probability a_{m+2}. $m+2$ being the maximum number of allowed re-transmissions in IEEE 802.11 wireless links.

Proof. In case a command is not lost in the IEEE 802.11 wireless link (less than $m+2$ re-transmissions), if we take the law of total probability and the analytical model in [7], the average wireless delay is

$$\mathbb{E}[\Delta_W(c_i)] = \sum_{j=0}^{m+1} a_j \cdot \mathbb{E}_j[\Delta_W(c_i)], \quad \forall c_i \tag{18}$$

Note that this happens with probability $1 - a_{m+2}$. Therefore, if we foresee that a command is not lost, we have to rescale (18) by $\frac{1}{1-a_{m+2}}$, i.e., the probability that a command is not lost.

Since we know that $\Delta(c_i) = \Delta_T(c_i) + \Delta_W(c_i)$, if we take the expectation at both sides of the equatlity, use Assumption 1, and the rescaled version of (18); we obtain (16).

According to [7], (17) holds because a packet is lost $\Delta_W(c_i) = \infty$ in a IEEE 802.11 wireless link with probability a_{m+2}. □

Corollary 1. *A control command c_i traversing a transport network, and an IEEE 802.11 wireless link under interference, experiences an unbounded delay, that is*

$$\mathbb{P}\Big(\Delta(c_i) > K,\ \forall K \in \mathbb{R}\Big) > 0 \tag{19}$$

Proof. In particular, Lemma 1 says that $\mathbb{P}\Big(\Delta(c_i) > K,\ \forall K \in \mathbb{R}\Big) = a_{m+2}$. □

Lemma 2. *In IEEE 802.11 wireless links, the causality assumption*

$$|\Delta(c_{i+1}) - \Delta(c_i)| \le |g(c_{i+1}) - g(c_i)|, \quad \forall c_{i+1}, c_i \tag{20}$$

only holds on average with probability $\sum_{j=0}^{m+1} a_j^2$.

Proof. We prove the lemma by cases, namely considering the different combinations of required re-transmissions of commands c_i and c_{i+1}.

If either command c_i or c_{i+1} is lost ($m+2$ re-transmissions), then we have with probability a_{m+2} that equation 20 does not hold, since either $\Delta_W(c_i) = \infty$ or $\Delta_W(c_{i+1}) = \infty$;

If c_{i+1} has j_2 RTX, and c_i has j_1 RTX (with $j_1 < j_2$), then

$$\Delta_W(c_{i+1}) - \Delta_W(c_i) = |\Delta_W(c_{i+1} - \Delta_W(c_i)| = \le g(c_{i+1}) - g(c_i) \tag{21}$$

We can take the expectation on the left side hand and obtain:

$$\mathbb{E}[\Delta_W(c_{i+1})] - \mathbb{E}[\Delta_W(c_i)] = T_s + j_2 T_{col} + \tilde{\sigma}\sum_{k=0}^{j_2}\frac{W_k - 1}{2} - T_s - j_1 T_{col} - \tilde{\sigma}\sum_{k=0}^{j_1}\frac{W_k - 1}{2} = (j_2 - j_1)T_{col} + \tilde{\sigma}\sum_{k=j_1+1}^{j_2}\frac{W_k - 1}{2} \tag{22}$$

[4]For the other axis residuals have the same pattern

with T_s the transmission time, T_{col} the collision time, $\tilde{\sigma}$ the average slot time, and W_k the k^{th} back-off window. Based on (22), the causality assumption does not hold, since

$$\exists\ T_s, T_{col}, \tilde{\sigma}, W_k : \quad (j_2 - j_1)T_{col} + \tilde{\sigma} \sum_{k=j_1+1}^{j_2} \frac{W_k - 1}{2} > g(c_{i+1}) - g(c_i) \quad (23)$$

The same reasoning applies in the case $j_1 > j_2$.

If command c_i required same re-transmissions as command c_{i+1} (i.e. $j_1 = j_2$), then (22) $\mathbb{E}[\Delta_W(c_{i+1})] = 0$, and the causality assumption (20) holds. This event occurs with probability $\sum_{j=0}^{m+1} a_j^2$. □

Corollary 2. *In IEEE 802.11 wireless links, the causality assumption*

$$|\Delta(c_{i+1}) - \Delta(c_i)| \leq |g(c_{i+1}) - g(c_i)|, \quad \forall c_{i+1}, c_i \quad (24)$$

does not hold.

Proof. The causality assumption does not hold with probability 1 (see Lemma), therefore, the causality assumption does not hold in IEEE 802.11 wireless links. □

REFERENCES

[1] D. Aschenbrenner, M. Fritscher, F. Sittner, M. Krauß, and K. Schilling, "Teleoperation of an industrial robot in an active production line," *IFAC-PapersOnLine*, vol. 48, no. 10, pp. 159–164, 2015, 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315009921

[2] P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 978–1013, 2018.

[3] S.-Y. Lien, S.-L. Shieh, Y. Huang, B. Su, Y.-L. Hsu, and H.-Y. Wei, "5G New Radio: Waveform, Frame Structure, Multiple Access, and Initial Access," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 64–71, 2017.

[4] "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Redline," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) - Redline*, pp. 1–5229, 2012.

[5] S. Vitturi, F. Tramarin, and L. Seno, "Industrial wireless networks: The significance of timeliness in communication systems," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 40–51, 2013.

[6] P. M. Kebria, H. Abdi, M. M. Dalvand, A. Khosravi, and S. Nahavandi, "Control methods for internet-based teleoperation systems: A review," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 1, pp. 32–46, 2019.

[7] P. Bosch, S. Latré, and C. Blondia, "An analytical model for IEEE 802.11 with non-IEEE 802.11 interfering source," *Computer Networks*, vol. 172, p. 107154, 2020.

[8] C. Guimarães, M. Groshev, L. Cominardi, A. Zabala, L. M. Contreras, S. T. Talat, C. Zhang, S. Hazra, A. Mourad, and A. de la Oliva, "DEEP: A Vertical-Oriented Intelligent and Automated Platform for the Edge and Fog," *IEEE Communications Magazine*, vol. 59, no. 6, pp. 66–72, 2021.

[9] C. Guimarães, X. Li, C. Papagianni, J. Mangues-Bafalluy, L. M. Contreras, A. Garcia-Saavedra, J. Brenes, D. S. Cristobal, J. Alonso, A. Zabala, J.-P. Kainulainen, A. Mourad, M. Lorenzo, and C. J. Bernardos, "Public and non-public network integration for 5growth industry 4.0 use cases," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 108–114, 2021.

[10] 5GPP and 5GIA, "The european 5G annual Journal/2021," https://bscw.5g-ppp.eu/pub/bscw.cgi/d424095/5G%20European%20Annual%20Journal%202021.pdf, 2021, online; accessed 28 October 2021.

[11] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network Intelligence in 6G: Challenges and Opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 7–12. [Online]. Available: https://doi.org/10.1145/3477091.3482761

[12] M. A. Uusitalo, M. Ericson, B. Richerzhagen, E. U. Soykan, P. Rugeland, G. Fettweis, D. Sabella, G. Wikström, M. Boldi, M.-H. Hamon, H. D. Schotten, V. Ziegler, E. C. Strinati, M. Latva-aho, P. Serrano, Y. Zou, G. Carrozzo, J. Martrat, G. Stea, P. Demestichas, A. Pärssinen, and T. Svensson, "Hexa-x the european 6g flagship project," in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2021, pp. 580–585.

[13] J. E. Solanes, A. Muñoz, L. Gracia, A. Martí, V. Girbés-Juan, and J. Tornero, "Teleoperation of industrial robot manipulators based on augmented reality," *The International Journal of Advanced Manufacturing Technology*, vol. 111, no. 3, pp. 1077–1097, 2020.

[14] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019.

[15] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5628–5635.

[16] J. E. Solanes, A. Muñoz, L. Gracia, A. Martí, V. Girbés-Juan, and J. Tornero, "Teleoperation of industrial robot manipulators based on augmented reality," *The International Journal of Advanced Manufacturing Technology*, vol. 111, no. 3, pp. 1077–1097, 2020.

[17] X. Li, A. Garcia-Saavedra, X. Costa-Perez, C. J. Bernardos, C. Guimarães, K. Antevski, J. Mangues-Bafalluy, J. Baranda, E. Zeydan, D. Corujo, P. Iovanna, G. Landi, J. Alonso, P. Paixão, H. Martins, M. Lorenzo, J. Ordonez-Lucena, and D. R. López, "5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[18] B. H. Jafari and M. W. Spong, "Passivity-based switching control in teleoperation systems with time-varying communication delay," in *2017 American Control Conference (ACC)*, 2017, pp. 5469–5475.

[19] D. Sun, F. Naghdy, and H. Du, "Neural network-based passivity control of teleoperation system under time-varying delays," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1666–1680, 2017.

[20] C. Ott, J. Artigas, and C. Preusche, "Subspace-oriented energy distribution for the time domain passivity approach," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 665–671.

[21] M. Franken, B. Willaert, S. Misra, and S. Stramigioli, "Bilateral telemanipulation: Improving the complementarity of the frequency- and time-domain passivity approaches," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2104–2110.

[22] H. K. Khalil, *Nonlinear systems*. Prentice-Hall, 2002.

[23] E. Mujčić, A. Mujčić, and S. Pajazetović, "Internet-based teleoperation using wave variables and correction of position error," in *2016 International Conference on Smart Systems and Technologies (SST)*, 2016, pp. 219–224.

[24] D. Sun, F. Naghdy, and H. Du, "Transparent four-channel bilateral control architecture using modified wave variable controllers under time delays," *Robotica*, vol. -1, pp. 1–17, 07 2014.

[25] ——, "Wave-variable-based passivity control of four-channel nonlinear bilateral teleoperation system under time delays," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 238–253, 2016.

[26] G. Niemeyer and J.-J. Slotine, "Stable adaptive teleoperation," *IEEE Journal of Oceanic Engineering*, vol. 16, no. 1, pp. 152–162, 1991.

[27] D.-H. Zhai and Y. Xia, "Adaptive control for teleoperation system with varying time delays and input saturation constraints," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 6921–6929, 2016.

[28] Y.-C. Liu and M.-H. Khong, "Adaptive control for nonlinear teleoperators with uncertain kinematics and dynamics," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2550–2562, 2015.

[29] P. M. Kebria, A. Khosravi, S. Nahavandi, P. Shi, and R. Alizadehsani, "Robust adaptive control scheme for teleoperation systems with delay and uncertainties," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3243–3253, 2020.

[30] Z. Chen, F. Huang, W. Sun, J. Gu, and B. Yao, "RBF-Neural-Network-Based Adaptive Robust Control for Nonlinear Bilateral Teleoperation Manipulators With Uncertainty and Time Delay," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 2, pp. 906–918, 2020.

[31] A. Forouzantabar, H. Talebi, and A. Sedigh, "Adaptive neural network control of bilateral teleoperation with constant time delay," *Nonlinear Dynamics*, vol. 67, no. 2, pp. 1123–1134, 2012.
[32] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.
[33] P. M. de Sant Ana, N. Marchenko, P. Popovski, and B. Soret, "Wireless control of autonomous guided vehicle using reinforcement learning," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.
[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[35] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. Citeseer, 1994, vol. 37.
[36] C. Lozoya, P. Martí, M. Velasco, J. Fuertes, and E. Martín, "Simulation study of a remote wireless path tracking control with delay estimation for an autonomous guided vehicle," *The International Journal of Advanced Manufacturing Technology*, vol. 52, pp. 751–761, 02 2011.
[37] M. Zorzi, R. Rao, and L. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 46, no. 2, pp. 445–455, 1997.
[38] J. R. Jackson, "Networks of waiting lines," *Operations research*, vol. 5, no. 4, pp. 518–521, 1957.
[39] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.
[40] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, 2014, pp. 3104–3112.
[41] P. P. Pham, "Comprehensive analysis of the IEEE 802.11," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 691–703, 2005.
[42] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.
[43] G. I. Palmer, V. A. Knight, P. R. Harper, and A. L. Hawa, "Ciw: An open-source discrete event simulation library," *Journal of Simulation*, vol. 13, no. 1, pp. 68–82, 2019.
[44] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[45] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.
[46] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.
[47] W. H. Greene, *Econometric analysis*. Pearson Education India, 2003.
[48] N. Sugiura, "Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's," *Communications in Statistics-theory and Methods*, vol. 7, no. 1, pp. 13–26, 1978.
[49] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*. Springer Science & Business Media, 2009.

Milan Groshev received the B.S. degree in telecommunication engineering from the Saints Cyril and Methodius University of Skopje, Macedonia in 2008 and the M.S. degree in telecommunication engineering from the Politecnico di Torino, Turin, Italy in 2016. He is currently pursuing the Ph.D. degree in telematics engineering at University Carlos III Madrid (UC3M), Spain.

His doctoral research investigates the integration of Edge and Fog in virtual environments with objective to build lightweight, low cost and smarter robots.

Jorge Martín Pérez obtained a B.Sc in mathematics, and a B.Sc in computer science, both at Universidad Autónoma de Madrid (UAM) in 2016. He obtained his M.Sc. and Ph.D in Telematics from Universidad Carlos III de Madrid (UC3M) in 2017 and 2021, respectively. His research focuses in optimal resource allocation in networks, and since 2016 he participates in EU funded research projects in UC3M Telematics department.

Carlos Guimarães is currently a Senior Technologist at ZettaScale Technology SARL (France) where he is developing data-centric networking solutions. Prior to that, he had worked as a Postdoctoral Researcher at Universidad Carlos III de Madrid (Spain), having received the M.Sc. degree in computer and telematics engineering from the Universidade de Aveiro (Portugal) in 2011, and the Ph.D. degree in computer science, in 2019, under the scope of MAP-i Doctoral Program (Portugal).

Antonio de la Oliva received his telecommunications engineering degree in 2004 and his Ph.D. in 2008 from the Universidad Carlos III Madrid (UC3M), Spain, where he has been an associate professor since then.

He is an active contributor to IEEE 802 where he has served as Vice-Chair of IEEE 802.21b and Technical Editor of IEEE 802.21d. He has also served as a Guest Editor of IEEE Communications Magazine. He has published more than 30 papers on different networking areas.

Carlos J. Bernardos received a Telecommunication Engineering degree in 2003, and a PhD in Telematics in 2006, both from the University Carlos III of Madrid, where he worked as a research and teaching assistant from 2003 to 2008 and, since then, has worked as an Associate Professor. His research interests include IP mobility management, network virtualization, cloud computing, vehicular communications and experimental evaluation of mobile wireless networks. He has published over 70 scientific papers in international journals and conferences. He has participated in several EU funded projects, being the project coordinator of 5G-TRANSFORMER and 5Growth.