Push- and Pull-based Effective Communication in Cyber-Physical Systems

Pietro Talli, Federico Mason, Federico Chiariotti, Andrea Zanella
Department of Information Engineering, University of Padova, Via G. Gradenigo 6/B, 35131, Padua, Italy
Emails: pietro.talli@phd.unipd.it, {federico.mason, federico.chiariotti, andrea.zanella}@unipd.it

***Abstract*—In Cyber-Physical Systems (CPSs), two groups of actors interact toward the maximization of system performance: the sensors, observing and disseminating the system state, and the actuators, performing physical decisions based on the received information. While it is generally assumed that sensors periodically transmit updates, returning the feedback signal only when necessary, and consequently adapting the physical decisions to the communication policy, can significantly improve the efficiency of the system. In particular, the choice between push-based communication, in which updates are initiated autonomously by the sensors, and pull-based communication, in which they are requested by the actuators, is a key design step. In this work, we propose an analytical model for optimizing push- and pull-based communication in CPSs, observing that the policy optimality coincides with Value of Information (VoI) maximization. Our results also highlight that, despite providing a better optimal solution, implementable push-based communication strategies may underperform even in relatively simple scenarios.**

***Index Terms*—Effective communication, Value of Information, Pull-based communication, Cyber-Physical Systems**

I. Introduction

The Industry 4.0 revolution has made Cyber-Physical Systems (CPSs) a fundamental pillar for multiple applications, including area surveillance, data muling, factory automation, teleoperation, and autonomous mobility [1]. A CPS is composed of sensors, which collect information about the system state, and actuators, that can physically modify it and control it. In standard scenarios, it is assumed that sensor transmissions take place periodically, with the goal of minimizing the Age of Information (AoI) at the receiver side. Such an approach ensures that, on average, the actuators have the most recent information about the system conditions.

Recent works have shown that AoI-based optimization may lead to sub-optimal performance, either because updates are unnecessary and may waste communication resources when the environment remains stable, or because an abrupt change in the system state may go unreported for a relatively long time, harming the control performance. In more advanced systems, the communication policy is tailored to the Value of Information (VoI) of the system, which implies that new transmissions are started whenever the environment evolution gets unpredictable at the receiver side. A further improvement in overall performance is obtained by adapting the actuators' decisions to the network operations, thus jointly optimizing communication and control [2]. In the pursuit of this goal, it is possible to adopt the multi-agent reinforcement learning (MARL) paradigm [3], an extension of reinforcement learning (RL) to handle multi-agent scenarios.

In the case of MARL optimization, each actor (i.e., sensor or actuator) is controlled via a distinct RL agent, which adapts its local actions towards the maximization of the overall performance. The fact that multiple agents interact with and simultaneously adapt to the same environment leads to non-stationary training conditions, where estimating the optimal policy is not always guaranteed [4]. Better solutions may be obtained by centralizing the control process in a single unit, which ensures perfect agent coordination at the cost of higher learning complexity and resource consumption. However, centralized solutions require reliable, high-frequency updates, which may be not always feasible, especially in the case Internet of Things (IoT) scenarios, where energy and communication constraints make it challenging to achieve a full environment perception in real-time [5].

In the case of a distributed optimization of CPSs, the placement of the computational intelligence controlling the communication systems is a critical design choice. If RL agents are installed at the sensors, it is possible to observe the current environment (or part of it) and update the actuators' knowledge as needed, in a *push-based* fashion. However, this solution may not be feasible if the sensors have limited computational and communication capabilities, as in the case of IoT networks. The other option is to install the computational intelligence at the actuators, each of which is associated with RL agent that has to both control the physical actions of the actuator itself and request new updates from the sensors. In such a case, communication takes place in a *pull-based* fashion [6], and decisions on when and whether to request an update are made without any information about the environment aside from the last state update and the elapsed time.

The pull-based problem was first presented in [7] and recently addressed in [8], where the authors consider two possible approaches for approximating the optimal policy: the first explicitly estimates the transition probability of the Markov Decision Process (MDP) describing the system evolution, while the second jointly learns communication and control policies as part of a single model. The push-based

This project was funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 (15 March 2022) of the Italian Ministry of University and Research, funded by the European Union NextGenerationEU Project. MUR grant number: PE_0000001, Concession Decree 1549 (11/10/2022) adopted by the Italian Ministry of University and Research, CUP C93C22005250001, project title: RESearch and innovation on future Telecommunications systems and networks, to make Italy more smART (RESTART). F. Chiariotti's activities are funded by NRRP "Young Researchers" grant REDIAL (SoE0000009).

configuration has been investigated in several recent works on Effective Communication (EC) systems [9], in which the actuators' policies must be adapted to the frequency of communication updates. However, the joint optimization of EC and control policies is still relatively unexplored, and, to the authors' knowledge, no studies have directly compared the push- and pull-based communication approaches and analyzed the interdependency between CPS optimization and the VoI.

To overcome these limitations, this work presents an analytical model for optimizing and evaluating push- and pull-based strategies in CPSs. We consider a simple CPS scenario with a single actuator, without local sensing capabilities, that is connected to a base station (BS) via a constrained communication channel. Using the channel involves a significant cost but represents the only way for the actuator to observe the system state. This design involves a trade-off between the minimization of the communication cost and the quality of control, which becomes less accurate as the transmission rate is reduced, in both the push- and pull-based versions. Using this model, we analyze the advantages and drawbacks of each configuration, proving relevant results and showing that the push-based system, while having better performance at the optimum, is a PPAD-hard problem [10].

II. System Model

We consider a system model with a single Actuator that can perceive the environment only through the information provided by a BS. This latter has full access to the system state and, therefore, embodies the sensor network described in the introduction. The goal of the model is to design and evaluate EC strategies between the Actuator and the BS. In the rest of this section, we will first formalize the environment as an MDP, and then characterize the possible solutions in the pull- and push-based communication scenarios.

A. Markov Decision Processes

We consider an MDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathbf{P}$ is the full transition matrix, r is the reward function and $\gamma \in [0,1)$ is the discount factor. Symbol $\mathbf{P}^a$ denotes the transition matrix associated with action $a \in \mathcal{A}$. Each row of $\mathbf{P}^a \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ defines the distribution of the next state over $\mathcal{S}$. For it to be a valid distribution, we impose $P^a_{ss'} \geq 0 \; \forall a \in \mathcal{A}, (s,s') \in \mathcal{S}^2$ and $\sum_{s'} P^a_{ss'} = 1$. The immediate reward for taking action a in state s and transitioning to state s' is denoted by $r^a_{s,s'}$.

In our problem, the Actuator cannot directly observe the state but incurs a cost c_t to obtain it from the BS, which observes it at every step with no associated cost. Hence, we can consider two possible scenarios: in the pull-based communication scenario, the Actuator itself decides whether to request a state update from the BS, which constitutes a passive actor in the system; in the push-based communication scenario, the BS itself can decide whether to transmit the system state to the Actuator. Notably, in the first case, we have a single agent installed at the Actuator, while, in the second case, we have two distinct agents that need to cooperate.

In the simplest case, every sensing action has the same cost, and the communication action is $c_t \in \{0,1\}$. When the state s_t is not available to the Actuator, its *a priori* belief distribution can be computed using the Markovian property of the system state evolution. Knowing the last observed state s_{t-k} and the vector $\mathbf{a} = \{a_{t-k}, a_{t-k+1}, \ldots, a_{t-1}\}$ containing the k actions undertaken since then, as well as the transition matrix $\mathbf{P}$, we can determine the belief distribution $\boldsymbol{\theta}^{\mathbf{a}}_{s_{t-k},k}$:

$$\boldsymbol{\theta}^{\mathbf{a}}_{s_{t-k},k} = \left(\prod_{\ell=t-k}^{t} \mathbf{P}^{a_\ell} \right) \mathbb{1}_{s_{t-k}}, \tag{1}$$

where $\mathbb{1}_s$ is a one-hot column vector whose elements are all 0, except for the one corresponding to s, which is equal to 1.

To jointly optimize the control and communication policies, we need to solve the following problem:

$$\text{maximize } \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \text{ such that } \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c_t\right] \leq C, \tag{2}$$

where C is the cumulative sampling cost that the system can tolerate and r_t is the instantaneous reward at time t. This problem can be seen as a constrained Partially Observable Markov Decision Process (POMDP) where each action, or combination of actions by the BS and Actuator, corresponds to the pair (a_t, c_t). By defining a fixed communication cost $\beta \in \mathbb{R}^+$ and using it as a dual parameter controlling the trade-off between the communication cost and the reward, we can reformulate (2) as an unconstrained POMDP:

$$\text{maximize } \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (r_t - \beta c_t)\right]. \tag{3}$$

In the following, we will denote values and functions related to the Actuator with the subscript A, while the BS will be associated with the subscript B.

The pull-based communication scenario can then be modeled as a problem known as an Action-Contingent Noiselessly Observable MDP (ACNO-MDP) [8], in which the state s_t is only available to the Actuator by means of a (costly) sensing action. Specifically, the sensing action corresponds to a request to the BS, which has perfect state information and can transmit it on demand. If we fix the polling strategy, the resulting system can be treated as a POMDP that can be solved optimally with point-based value iteration [11] or other methods that consider the α-vectors [12].

The push-based problem is more common in the EC literature and is often modeled as a Remote POMDP [13]. In this case, communication is initiated by the BS, which knows both the last transmission to the receiver s_{t-k} and the current state s_t, along with the time since the last update k. Intuitively, the performance of a push-based system should improve with respect to the pull-based one, as the former can exploit information on the specific realization of the system state trajectory, rather than just on its statistics. On the other hand, remote POMDPs are multi-agent problems and may involve critical issues in terms of coordination between the BS and Actuator, which will be explored in the following.

Algorithm 1 Pull-Based Modified Policy Iteration

Require: $\mathbf{P}, r, \beta$
Initialize $\mathbf{V}_A(s,k) \leftarrow 0$, randomize $\boldsymbol{\pi}_A(s,k)$, $\boldsymbol{\Delta}$
while true **do**
  for $(s,k) \in \mathcal{S} \times \{0, ..., T_{\max}\}$ **do**
    $V'_A(s,k) \leftarrow$Update using (4)
  $\mathbf{V}_A \leftarrow \mathbf{V}'_A$ ▷ Value update step
  for $(s,k) \in \mathcal{S} \times \{0, ..., T_{\max}\}$ **do**
    $\pi'_A(s,k) \leftarrow$Update using (6)
    $\Delta'(s) \leftarrow$Update using (7)
  if $\boldsymbol{\pi}'_A = \boldsymbol{\pi}_A \wedge \boldsymbol{\Delta}' = \boldsymbol{\Delta}$ **then**
    return $\boldsymbol{\pi}_A, \boldsymbol{\Delta}$ ▷ Convergence
  else
    $\boldsymbol{\pi}_A, \boldsymbol{\Delta} \leftarrow \boldsymbol{\pi}'_A, \boldsymbol{\Delta}'$ ▷ Policy improvement step

III. Analytical Solution

To solve the MDP, we consider a model-based approach: any RL agent interacting with the environment can compute optimal control and communication policies with complete knowledge of the reward function and the state transition model. This can also be accomplished with statistical learning [8], i.e., by learning the transition probabilities of the underlying MDP before solving the communication-constrained problem. This choice allows for an easy comparison of the policies, without any training issues, and the results can be applied directly to any properly trained agent.

A. Pull-based Communication

To solve the pull-based communication problem optimally, we propose a modified Policy Iteration (PI) algorithm to learn the communication control policies jointly. We recall that PI always converges to the optimal solution in single-agent MDPs, as the one considered in pull-based scenarios.

We formulate a new MDP in which each time step is divided into two sub-steps: one for the control action and one for the communication action. The state of this new MDP is also expanded to the tuple (s_{t-k}, k), which is available to the agent. The action space is then $\mathcal{A}$ in control sub-steps, and $\{0,1\}$ in communication sub-steps. The agent then splits the policy into two parts:

- $\pi_A(s_{t-k}, k)$, the control policy, which maps each state $(s_{t-k}, k) \in \mathcal{S} \times \mathbb{N}$ to an action $a \in \mathcal{A}$;
- $\Delta(s_{t-k})$, the communication policy, which maps each state $s_{t-k} \in \mathcal{S}$ to the time until the next update, $c \in \mathbb{N}^+$.

Since the policy $\boldsymbol{\pi}_A$ depends only on the last observed state, we can determine the vector $\mathbf{a}_s^{\boldsymbol{\pi}_A}$ containing the k actions chosen by the Actuator after observing s. Following the bootstrap principle, the modified PI algorithm then updates the value of a control sub-state as follows:

$$V'_A(s,k) = \sum_{(s',s'')\in\mathcal{S}^2} \theta_{s,k}^{\mathbf{a}_s^{\boldsymbol{\pi}_A}}(s') P_{s',s''}^{\pi_A(s,k)} \Bigg[r_{s',s''}^{\pi_A(s,k)} + \gamma \left(\delta_{s,k}(V_A(s'',0) - \beta) + (1-\delta_{s,k})\, V_A(s,k+1)\right) \Bigg], \tag{4}$$

where $\delta_{s,k}$ is a utility function equal to 1 if $\Delta(s) = k+1$, i.e., if an update is requested after $k+1$ steps, and 0 otherwise. We observe that the division into sub-steps does not affect the reward and the solution to the modified problem is also optimal for the original MDP. Therefore, we do not then need to compute the control sub-state values explicitly.

To perform the policy improvement step, we modified the standard update rule by using a reward that considers the entire evolution of the Markov chain until the state is sampled again. If we consider a state sequence $\mathbf{s}'$, which begins k steps after state s is observed, the belief $\Theta_{s,k}^{\boldsymbol{\pi}_A}(\mathbf{s}')$ follows from (1):

$$\Theta_{s,k}^{\boldsymbol{\pi}_A}(\mathbf{s}') = \theta_{s,k}^{\mathbf{a}_s^{\boldsymbol{\pi}_A}}(s'(1)) \prod_{\ell=1}^{\mathcal{L}(\mathbf{s}')-1} P_{s'(\ell),s'(\ell+1)}^{\pi_A(s,\ell+k-1)}, \tag{5}$$

where $\mathcal{L}(\cdot)$ is a function whose outcome is the dimensionality of the input. Hence, the control policy is improved as follows:

$$\boldsymbol{\pi}'_A = \underset{\boldsymbol{\pi}'_A \in \mathcal{A}^{|\mathcal{S}|}}{\arg\max} \sum_{\mathbf{s}' \in \mathcal{S}^{\Delta(s)-k+1}} \Theta_{s,k}^{\boldsymbol{\pi}'_A}(\mathbf{s}') \Bigg[\sum_{\ell=1}^{\Delta(s)-k} \gamma^{\ell-1} r_{s'(\ell),s'(\ell+1)}^{a_s^{\boldsymbol{\pi}'_A}(\ell+k-1)} + \gamma^{\Delta(s)-k} \left(V_A(s'(\Delta(s)-k+1), 0) - \beta\right) \Bigg]. \tag{6}$$

Instead, the communication policy is updated as:

$$\Delta'(s) = \underset{n \in \mathbb{N}^+}{\arg\max} \sum_{\mathbf{s}' \in \mathcal{S}^n} \Theta_{s,1}^{\boldsymbol{\pi}_A}(\mathbf{s}') \Bigg[\sum_{\ell=1}^{n-1} \gamma^{\ell} r_{s'(\ell),s'(\ell+1)}^{\pi_A(s,\ell)} + \gamma^n (V_A(s'(n),0) - \beta) + r_{s,s'(1)}^{\pi_A(s,0)} \Bigg]. \tag{7}$$

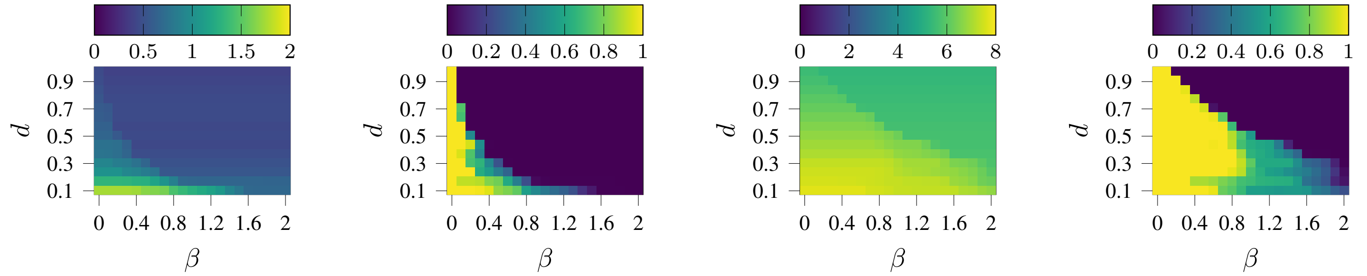
Using the whole sequence of intermediate steps in the update equations is essential to take into account that the states $(s,k)\ \forall\, k > 0$ are non-Markovian, as the belief distribution depends on the action sequence taken since the last communication. The pseudocode for the PI scheme for pull-based communication is given in Alg. 1.

Algorithm 2 Push-Based Alternate Policy Iteration

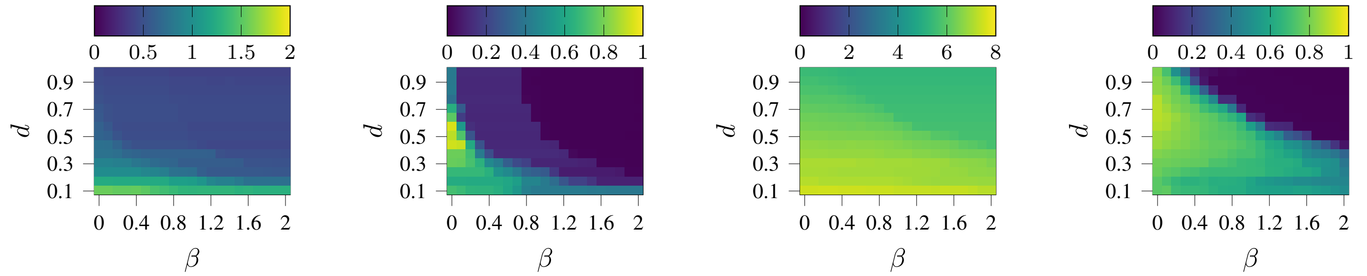
Require: $\mathbf{P}, r, \beta, \boldsymbol{\pi}_B^{(0)}$
Initialize $\boldsymbol{\pi}_B \leftarrow \boldsymbol{\pi}_B^{(0)}$, randomize $\boldsymbol{\pi}_A$
while true **do**
  $\boldsymbol{\pi}'_A \leftarrow$ControlPolicyIteration$(\mathbf{P}, r, \beta, \boldsymbol{\pi}_B)$
  $\boldsymbol{\pi}'_B \leftarrow$CommunicationPolicyIteration$(\mathbf{P}, r, \beta, \boldsymbol{\pi}'_A)$
  if $\boldsymbol{\pi}'_A = \boldsymbol{\pi}_A \wedge \boldsymbol{\pi}'_B = \boldsymbol{\pi}_B$ **then**
    return $\boldsymbol{\pi}_A, \boldsymbol{\pi}_B$ ▷ Convergence
  else
    $\boldsymbol{\pi}_A, \boldsymbol{\pi}_B \leftarrow \boldsymbol{\pi}'_A, \boldsymbol{\pi}'_B$

B. Push-based Communication

In the case of push-based communication, we propose a simple strategy which we call Alternate Policy Iteration (API): starting from a default policy for the BS, we first optimize the Actuator's policy and then optimize the BS, considering the Actuator's actions as fixed. Since one agent's policy is fixed, the other agent sees a Markovian environment, and standard PI can be applied in each step. The procedure is then repeated until both the policies converge, i.e., each agent follows the optimal policy with respect to the other. As we will prove in Sec. V, this strategy always converges in a finite number of steps. The pseudocode for API is given in Alg. 2.



(a) Average reward (pull-based, focused). (b) Update frequency (pull-based, focused). (c) Average reward (pull-based, spread). (d) Update frequency (pull-based, spread).



(e) Average reward (push-based, focused). (f) Update frequency (push-based, focused). (g) Average reward (push-based, spread). (h) Update frequency (push-based, spread).

Fig. 1: Figure of the Reward and Communication Cost for different densities and values of β.

IV. Numerical Evaluation

In this section, we evaluate the two methods proposed on randomly generated MDPs that follow a common structure for a fair and easy comparison. We consider a system with $|\mathcal{S}| = 30$ states and $|\mathcal{A}| = 4$ actions. The reward depends only on the reached state and is defined as

$$r^a_{s,s'} = e^{-\alpha|s'-s_0|}, \ \forall s \in \mathcal{S}, a \in \mathcal{A}, \quad (8)$$

where $s_0 \in \mathcal{S}$ is a target state, and α is a decay parameter determining how much the reward is concentrated around s_0. We consider two possible reward configurations:

- **Focused reward**: $\alpha = 10$, the only state that gives a significant reward is s_0.
- **Spread reward**: $\alpha = 0.01$, the reward is spread among the states near s_0.

In both configurations, the MDP transition matrix is generated as follows: we start from a deterministic transition matrix for every action and then progressively increase the number of non-zero elements by enabling transitions to neighboring states with a certain probability. As more transitions are added, the transition matrix becomes denser, i.e., the number d of non-zero elements divided by the total number of entries of the matrix increases. The density of the deterministic matrix is simply $|\mathcal{S}|^{-1}$. If the initial deterministic matrix $\mathbf{P}^a$ has a transition from s to s', i.e., $P^a_{s,s'} = 1$, the distribution at density d is:

$$P^a_{s,s''} = \frac{4(d|\mathcal{S}| - 2|s' - s''|)}{(d|\mathcal{S}|+1)^2}, \ \forall s'' \text{ s.t. } |s'-s''| < \frac{d|\mathcal{S}|}{2}. \quad (9)$$

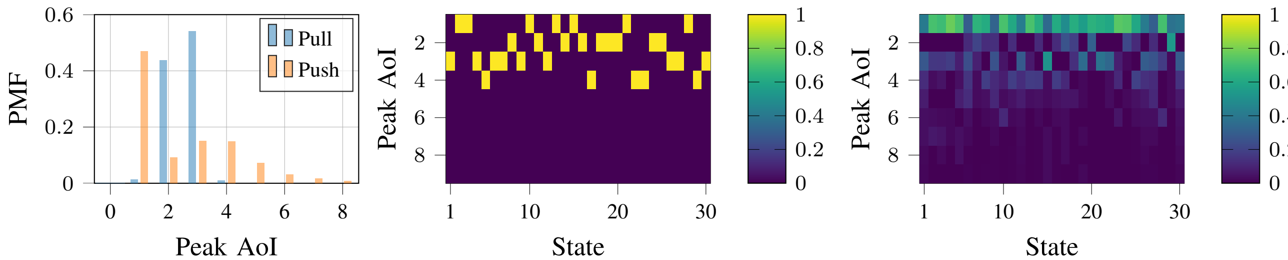
According to this approach, the probability of transitioning to neighbor states decreases linearly with the distance from the original transition state (the distribution has a triangular shape). Higher-density MDPs are then simply more unpredictable versions of the same initial model.

To investigate our model in different scenarios, we repeatedly increase the number of transitions for each state by adding two connections at a time, obtaining a total of 15 MDPs with increasing densities. We also consider different communication costs $\beta \in \{0, 0.1, \ldots, 2\}$. We test each combination of d and β with both pull-based and push-based policies, using the same matrices for the push- and pull-based systems. The code for the numerical evaluations is available to replicate the results[1].

Fig. 1 shows the average obtained reward of the physical process and the frequency of the updates between the BS and the Actuator. First, we can note that the push-based system always obtains a higher overall reward than the pull-based one, as would be expected: the BS, acting with full knowledge of the state, can determine whether an update has a high VoI, i.e., whether it meaningfully affects the Actuator.

Fig. 1b and Fig. 1d also show that the pull-based system only has 2 working points for higher values of d (i.e., more unpredictable transitions): either the agent requests an update at each step, or it never does. This is because the high uncertainty in the belief distribution makes it hard to use it to predict the optimal action even after a single step. This leads the Actuator to always request the new state information if β is low, i.e., if the relative VoI of the update increases, or never to request updates if β is high, moving blindly.

[1]https://www.github.com/pietro-talli/Age-Value-CC



(a) AoI distribution. (b) Pull-based AoI distribution by state. (c) Push-based AoI distribution by state.

Fig. 2: Peak AoI distribution for $d = 0.1$, $\beta = 1$ and sparse reward.

As Fig. 1f and Fig. 1h clearly show, the push-based strategy can reduce the update frequency. In this case, the BS can check if the state of the system evolves in the predicted way, transmitting an update to the Actuator only when a low-probability transition occurs and its knowledge becomes obsolete. This is particularly beneficial if the transition model is more predictable, i.e., for lower values of d, and leads to 3 working regimes: firstly, if d is very high, the BS communicates only if the reward in the next step changes significantly, letting the Actuator move blindly in all other cases. This behavior is particularly noticeable in the focused reward case when the BS only communicates if the state can lead directly to the target.

On the other hand, the BS will also communicate sporadically if the system is very predictable, i.e., for very low values of d, for the reasons we outlined above. In between these two extremes, the BS will communicate more often, as the system is not so random that the belief distribution of the Actuator quickly becomes useless, but also not too predictable. As this type of behavior requires the full knowledge of the state, it cannot be obtained in a pull-based scenario.

To better highlight the different behaviors between pull- and push-based communication, we look at Fig. 2, which analyzes the Peak AoI for a highly predictable MDP with a sparse reward, with $d = 0.1$ and $\beta = 1$. In particular, Fig. 2a shows the Probability Mass Function (PMF) of the Peak AoI for the two approaches: while the average update frequency is similar, the push-based system has a much higher variance than the pull-based one. This is because the first approach triggers transmissions when needed, sometimes after a single step, sometimes after many, depending on the environment's evolution; instead, the pull-based approach only relies on the last observed state, and its decisions are always deterministic. This is confirmed by Fig. 2b, which shows how, in the pull-based scenarios, the Peak AoI distribution for each state is concentrated in a single point. On the other hand, the push-based system may transmit after a long time even when starting from states that have a high expected VoI after a single step, as Fig. 2c shows: while the pull-based system transmits after 1 step when starting from state 2, the push-based system has a non-zero probability of waiting 5 or more steps before transmitting again.

V. Age and Value of Information in Effective Communication

Concepts like AoI and VoI are crucial to analyze the timeliness and relevance of system status updates. In our scenario, we might consider an update strategy that balances the frequency of updates (which determines the average communication cost) with the reward obtained by the Actuator. Let $J(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ be the long-term reward of the Actuator starting from state s and β be the communication cost. We can then give a more general definition of performance in pull-based or scheduled systems as a function of the selected update periods $\mathbf{\Delta} \in (\mathbb{N}^+)^{|\mathcal{S}|}$:

$$R_\beta(\mathbf{\Delta}) = \sum_{s\in\mathcal{S}} \phi(s) \left(J(s|\mathbf{\Delta}) - \mathbb{E}\left[\sum_{n=0}^{\infty} \beta \gamma^{n\Delta(s')} P(s'|s,\mathbf{\Delta}) \right] \right), \tag{10}$$

where $\phi(s)$ is the stationary state distribution induced by the control policy and the update periods $\mathbf{\Delta}$.

We introduce the concept of Pareto dominance [14] to compare the performance of complex schemes with multiple objectives.

Definition 1. *an n-dimensional tuple $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)$* Pareto dominates $\boldsymbol{\eta}'$ *(which we denote as $\boldsymbol{\eta} \succeq \boldsymbol{\eta}'$) if and only if each element of $\boldsymbol{\eta}$ is equal to or better than the corresponding element of $\boldsymbol{\eta}'$, i.e., $\boldsymbol{\eta} \succeq \boldsymbol{\eta}' \iff \eta_j \geq \eta'_j \ \forall j$. Strict Pareto dominance makes the inequality strict for at least one parameter:*

$$\boldsymbol{\eta} \succ \boldsymbol{\eta}' \iff \boldsymbol{\eta} \succeq \boldsymbol{\eta}' \land \exists i : \eta_i > \eta'_i. \tag{11}$$

This concept can then be extended to multi-objective optimization, i.e., to schemes that may have different parameters and multiple performance metrics.

Definition 2. *Let us consider two schemes x and y, parameterized by a vector $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$. Scheme x Pareto dominates y if and only if:*

$$x \succeq y \iff \forall \boldsymbol{\theta}_y \exists \boldsymbol{\theta}_x : \boldsymbol{\eta}(\boldsymbol{\theta}_x) \succeq \boldsymbol{\eta}(\boldsymbol{\theta}_y), \tag{12}$$

where $\boldsymbol{\eta}(\boldsymbol{\theta}_z)$ is the n-dimensional performance vector associated with scheme z and parameters $\boldsymbol{\theta}_z$. The definition of strict dominance between schemes is analogous.

If we only consider the AoI, without including state information in the optimization, we can envision a periodic policy in which the BS sends a new update every $\Delta \in \mathbb{N}^+$ steps. In this case, the optimal update interval Δ^*_{AoI} is then:

$$\Delta^*_{\text{AoI}}(\beta) = \underset{\Delta \in \mathbb{N}^+}{\arg\max} \sum_{s \in \mathcal{S}} \phi(s) \left(J(s \mid \Delta) - \sum_{n=0}^{\infty} \beta \gamma^{n\Delta} \right). \tag{13}$$

In particular, we have that $\Delta(s) = \Delta^*_{\text{AoI}} \; \forall s \in \mathcal{S}$, i.e., the optimal AoI Δ^*_{AoI} is the same for all states.

An adaptive pull-based strategy selecting the best $\Delta_{\text{pull}}(s)$ based on the state $s \in \mathcal{S}$ received in the last update can improve the reward function in (10) will respect the AoI-based approach. In this case, the optimization problem then becomes:

$$\boldsymbol{\Delta}^*_{\text{pull}}(\beta) = \underset{\boldsymbol{\Delta} \in (\mathbb{N}^+)^{|\mathcal{S}|}}{\arg\max} R_\beta(\boldsymbol{\Delta}). \tag{14}$$

Theorem 1. *The pull-based EC strategy Pareto dominates the AoI-based policy, i.e.,* $\boldsymbol{\Delta}^*_{pull}(\beta) \succeq \Delta^*_{AoI}(\beta) \, \forall \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma, \beta \rangle$.

Proof. First, we can trivially show that a scheme working better than another for every value of β is a stronger condition than Pareto dominance. We can then prove that this condition holds by *reductio ad absurdum*: we consider a hypothetical optimal interval Δ^*_{AoI} which performs better than pull-based EC for a given value of β. In this case, $\boldsymbol{\Delta}^*_{\text{pull}}$ cannot be optimal, as the vector in which all elements are equal to Δ^*_{AoI} is one of the possible choices for pull-based EC. □

We observe that the pull-based strategy can always fall back to the same point as the AoI-based one, simply by considering the same interval for every state. In this case, the VoI is implicitly determined by the strategy: the Actuator only requests an update if the expected increase in the long-term reward is larger than the communication cost β, and that increase is precisely the value of the update.

We can further improve performance by adopting a push-based approach, allowing the BS may be able to independently observe the system state s_t and consequently decide whether to update the Actuator's knowledge. This prevents us from defining a fixed state-dependent update period, as the policy $\pi_B(s, k, s_t)$ depends on the current state s_t as well as on the last state update s_{t-k} and the elapsed time k. Particularly, the value function for the BS is:

$$\begin{aligned} V_B(s,k,s') = {} & \pi_B(s,k,s')(V_B(s',0,s') - \beta) + (1 - \pi_B(s,k,s')) \\ & \times \sum_{s'' \in \mathcal{S}} P^{\pi_A(s,k)}_{s's''} \left(r^{\pi_A(s,k)}_{s',s''} + \gamma V_B(s, k+1, s'') \right), \end{aligned} \tag{15}$$

and the optimal policy is the one maximizing the communication value.

The discovery of the optimal policies through API can be seen as the solution to a Markov game [15]: the two agents act as players in a game where the moves are the possible policies and the payoff for each player is the expected reward in the initial state.

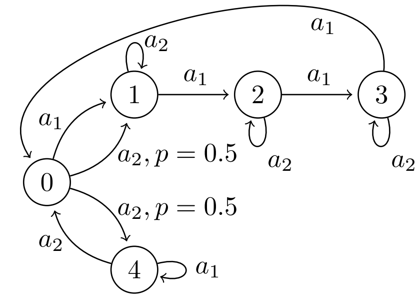


Fig. 3: Example of a Markov model with 5 states and 2 actions.

Theorem 2. *The API approach leads to a Nash Equilibrium (NE) policy in the push-based problem.*

Proof. Firstly, we can trivially prove that the considered Markov game is an exact potential game [16]: as the reward for the two agents is the same, the expected long-term reward is a potential function for the game. We then consider the API strategy: each round of the iterated algorithm leads to the optimal policy when the strategy of the other agent is given, due to the optimality of standard PI. The API algorithm is then an Iterated Best Response (IBR) scheme for the game, which leads to a NE in a finite number of steps in all finite potential games [16]. □

However, reaching an NE is *not* a guarantee of Pareto optimality: games may have multiple NEs, and finding the optimal one is PPAD-hard [10]. The push-based approach may be actively harmful, even with respect to an AoI policy.

Theorem 3. *The optimal EC solution to the push-based problem,* $\boldsymbol{\pi}^*_{B,push}$*, Pareto dominates the pull-based EC strategy:*

$$\boldsymbol{\pi}^*_{B,push}(\beta) \succeq \boldsymbol{\Delta}^*_{pull}(\beta) \succeq \Delta^*_{AoI}(\beta) \; \forall \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma, \beta \rangle. \tag{16}$$

However, the solution obtained by the API strategy, $\boldsymbol{\pi}^{API}_{B,push}$*, does not Pareto dominate the AoI-based strategy:*

$$\exists \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma, \beta \rangle : \boldsymbol{\pi}^{API}_{B,push}(\beta) \not\succeq \Delta^*_{AoI}(\beta). \tag{17}$$

Proof. The proof of the first statement is simple, and follows the proof of Theorem 1: as the BS can act with full knowledge of the state, the pull-based solution is a possible solution to the push-based problem, and in some cases, better solutions exist. The knowledge of the realization of the state can improve the reward, either by cutting unnecessary transmissions or by improving the actuator's performance.

As finding the optimal solution to a Markov game is PPAD-hard [10], no polynomial-time algorithm can reliably find $\boldsymbol{\pi}^*_{B,\text{push}}$. We can then give a counterexample to prove the second part of the theorem: we consider a simple MDP with 5 states and 2 actions, whose evolution is depicted in Fig. 3. The two transition matrices corresponding to a_1 and a_2 are:

$$\mathbf{P}^{a_1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{P}^{a_2} = \begin{bmatrix} 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{18}$$

The reward is then always 0, except for when the environment transits to state 0, i.e., we have $r^a_{s,0} = 1$ and $r^a_{s,s'} = 0 \; \forall s' \neq 0$.

We can easily see that taking action a_1 in state 0 leads to a loop with 4 states, while taking action a_2 may lead to a shorter path back to the reward-giving state. We consider a policy Δ_{AoI} that transmits in each odd step, ensuring that the actuator always knows if it lands in state 1 or state 4 after taking action a_2. Its expected long-term reward is:

$$\begin{aligned} R(\Delta_{\text{AoI}}) =& 1 - \gamma\beta + \frac{\gamma^2 R(\Delta_{\text{AoI}})}{2} - \frac{\gamma^3\beta}{2} + \frac{\gamma^4 R(\Delta_{\text{AoI}})}{2} \\ =& \frac{2-(2\gamma+\gamma^3)\beta}{2(1-\gamma^2-\gamma^4)}. \end{aligned} \tag{19}$$

Considering a push-based approach and applying the API strategy, the final results depend on the initial policy of the BS. If the process starts from a policy that communicates often, e.g., one that always communicates the state, the algorithm will converge to the optimum joint policy, which only communicates if it deviates from the short cycle (i.e., if the system ends up in state 1). In this case, the expected reward is

$$\begin{aligned} R(\boldsymbol{\pi}^*_{B,\text{push}}) =& 1 - \gamma\beta + \frac{\gamma^2 R(\boldsymbol{\pi}^*_{B,\text{push}})}{2} + \frac{\gamma^4 R(\boldsymbol{\pi}^*_{B,\text{push}})}{2} \\ =& \frac{2-\gamma\beta}{2(1-\gamma^2-\gamma^4)}, \end{aligned} \tag{20}$$

which is better than the AoI policy for any value of $\beta \in \mathbb{R}^+$ and $\gamma \in (0, 1)$.

Instead, if the API strategy starts from a policy that *never* communicates, the actuator will take the conservative choice, and always take action a_1. This is another NE of the system, as the BS should never communicate if the actuator's policy is independent of the state. The reward for this solution is:

$$R(\boldsymbol{\pi}^{\text{API}}_{B,\text{push}}) = 1 + \gamma^4 R(\boldsymbol{\pi}^{\text{API}}_{B,\text{push}}) = (1-\gamma^4)^{-1}. \tag{21}$$

In this case, the API solution is not Pareto dominant, as it performs worse than a simple AoI-based strategy under the following conditions:

$$\beta < \frac{2}{(2+\gamma^2)(1-\gamma^4)}. \tag{22}$$

□

The optimal policy might not be easy to obtain in more complex systems, and if we consider previously unknown cases in which PI must be replaced by RL, running multiple training procedures to achieve the optimal NE might be expensive or impossible. Despite this evidence, most of the literature on VoI has so far focused on *push-based* solutions [17], which fall prey to this coordination problem.

VI. Conclusion

This work analyzes EC strategies in CPSs, considering the performance of the control system as the VoI and analyzing the interactions between the communication and control policies. We propose an analytical framework to adapt classical optimization tools such as PI to this context, providing numerical results that show a strong dependency between the VoI and the structure of the underlying MDP. Less predictable MDPs and sparser rewards tend to lead to a higher value of updates, although there are interesting patterns in the interaction between communication and control.

Finally, our analysis revealed that the common push-based view of communication and control problems may not always be optimal, as the game theoretical properties of multiagent scenarios do not guarantee that a solution may be better even than a simple AoI-based optimization. Future extensions of this work will concentrate on this conundrum, analyzing more complex examples of remote POMDPs and devising heuristic strategies that provide good performance in practical scenarios.

References

[1] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, "Networked control systems: A survey of trends and techniques," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 1, pp. 1–17, 2019.

[2] F. Mason, F. Chiariotti, A. Zanella, and P. Popovski, "Multi-agent reinforcement learning for pragmatic communication and control," *Preprint arXiv:2302.14399*, 2023.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE T. Sys., Man, & Cyber., Part C (Appl. & Rev.)*, vol. 38, no. 2, pp. 156–172, 2008.

[5] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges," *IEEE Comm. Surv. & Tut.*, vol. 22, no. 3, pp. 1722–1760, 2020.

[6] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *Int. Symp. Inf. Th. (ISIT)*. IEEE, 2017, pp. 326–330.

[7] G. E. Monahan, "State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms," *Mgmt. Sci.*, vol. 28, no. 1, pp. 1–16, 1982.

[8] H. A. Nam, S. Fleming, and E. Brunskill, "Reinforcement learning with state observation costs in action-contingent noiselessly observable Markov decision processes," *Adv. Neur. Inf. Proc. Sys. (NeurIPS)*, vol. 34, pp. 15 650–15 666, 2021.

[9] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," in *Int. Conf. Learning Repr. (ICLR)*, 2019.

[10] X. Deng, N. Li, D. Mguni, J. Wang, and Y. Yang, "On the complexity of computing Markov perfect equilibrium in general-sum stochastic games," *Nat. Sci. Rev.*, vol. 10, no. 1, p. nwac256, 2023.

[11] J. Pineau, G. Gordon, S. Thrun *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *18th Int. Joint Conf. AI (IJCAI)*, vol. 3, 2003, pp. 1025–1032.

[12] T. Smith and R. Simmons, "Point-based POMDP algorithms: improved analysis and implementation," in *21st Conf. Uncertainty AI*, 2005, pp. 542–549.

[13] T.-Y. Tung, S. Kobus, J. P. Roig, and D. Gündüz, "Effective communications: A joint learning and communication framework for multi-agent reinforcement learning over noisy channels," *IEEE J. Sel. Areas Comm.*, vol. 39, no. 8, pp. 2590–2603, 2021.

[14] V. Pareto, *Manuale di economia politica con una introduzione alla scienza sociale*. Società Editrice Libraia, 1919.

[15] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal Nash equilibrium in team Markov games," *Adv. Neur. Inf. Proc. Sys. (NIPS)*, vol. 15, 2002.

[16] D. Monderer and L. S. Shapley, "Potential games," *Games & Econ. Behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[17] D. Gündüz, F. Chiariotti, K. Huang, A. E. Kalør, S. Kobus, and P. Popovski, "Timely and massive communication in 6G: Pragmatics, learning, and inference," *IEEE BITS*, 2023.