Learning Semantic Congestion Control for Cyber Physical Systems

Polina Kutsevol, Yash Deshpande, Wolfgang Kellerer
Chair of Communication Networks, Technical University of Munich, Germany
Email: {polina.kutsevol, yash.deshpande, wolfgang.kellerer}@tum.de

***Abstract*—Goal-oriented (GO) semantic communication facilitates scaling modern networks with growing real-time traffic generated within networked Cyber Physical Systems (CPSs). Network resource management in GO communication prioritizes data effectiveness for the application goal. This implies reducing network resources allocated to low-priority information. Existing GO approaches often lack generalization, because they tailor particular network schemes to particular applications. In the current work, we propose a practical GO scheme operating in the transport layer (TL) middleware, i.e., not requiring specific hardware or network structure. Using Reinforcement Learning (RL), the proposed GO RL TL captures the potential contribution of the currently sampled observed state to the real-time CPS process evolution at the remote monitor. Together with the network congestion level, the state's effect on the application goal determines whether the distributed sensors deploying GO RL TL agents accept corresponding packets into the network or discard them. The offline environment for training uses the real data traces of traffic patterns and application dynamics. The model generalizes to arbitrary network and application setups present in traces by learning the corresponding inter-dependencies from data. The extensive hardware tests witness the adaptability of the proposed GO RL TL, as well as its superiority in application performance compared to competitors. GO RL TL improves remote estimation mean-squared error by 20% to 100% in static network conditions, and by $\sim 30\%$ in the dynamic setup.**

I. Introduction

The modern advancements in networking technologies enable modular and more flexible setups of real-time Cyber Physical Systems (CPSs). Industrial IoT, Smart Homes and Cities, medical and robotic devices include more and more communication-supported operations, often realized via wireless networks [1]. However, the communication capacities are often not enough to promptly deliver massive amounts of real-time sensory data. CPSs' application performance is, in turn, highly sensitive to network adverse effects. For instance, Fig. 1 demonstrates the dynamics of a real-time control process actuated towards zero. The periodically sampled state measurements are transmitted to the actuator over a wireless network. A fast sampling rate, as in the left plot of Fig. 1, leads to increased network congestion and higher round-trip times (RTTs). Only observing the delayed dynamics, the actuator becomes less effective, and the process significantly deviates from the setpoint. That is, the application performance is disturbed. A straightforward way to reduce congestion and, by that, lower delays is to decrease the sampling rate, as shown in the middle plot in Fig. 1. However, it can also be detrimental because the process gets destabilized between the actuation points. The natural solution to facilitate stability when the transmission opportunities are limited is to trigger measurements not periodically but when the deviation gets higher, as given in the right plot in Fig. 1, i.e., directly include the application content into decision-making.

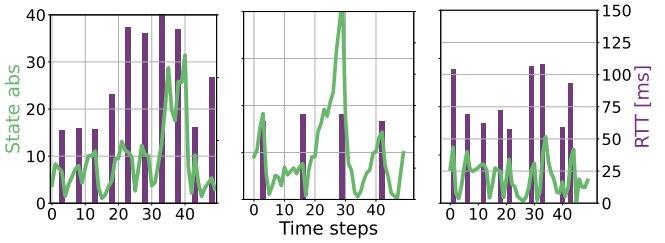


Fig. 1: Example dynamics of the process actuated towards zero setpoint, including fast, slow and state-based sampling. The actuator observes delayed state samples after the transmission over the network. The bars' x-coordinate corresponds to the transmission instance, and the height of the bar - experienced round-trip time (RTT).

This is an example of applying the novel concept of semantic goal-oriented (GO) communication [2]. It is often applied in the CPS field due to the strong dependence of the CPS performance on the network conditions. Semantic communication emphasized the significance of the transmitted data for the communication purpose. Information crucial for the application gets priority when managing scarce network resources. Improved process stability in the right plot in Fig. 1 witnesses the benefits of the GO communications approach, involving sampling the state measurements based on their effect on further actuation.

A joint network application co-design often implies tailoring specific communication mechanisms to specific applications and considering nuanced application-network interplay. For example, control theory approaches use precise network modeling to ensure stability under network constraints [3] and develop joint transmission-control strategies [4]. Network-oriented schemes incorporate general mechanisms for real-time applications [5], not tailored for particular CPSs. Alternatively, there are works proposing particular protocol modifications like scheduling over WiFi [6] or 5G [7]. Thus, existing semantic communication approaches often have limited gener-

The authors acknowledge the financial support by the Federal Ministry of Research, Technology and Space of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK002, and by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) project KI.FABRIK (grant no. DIK0249).

978-3-903176-75-1 ©2025 IFIP

alization capabilities. In networking, end-to-end transmission control is realized at the Transport Layer (TL). Similarly, we aim at hardware- and infrastructure-independent GO TL design, enhancing end-to-end application performance for a generic underlying network. We envision that software-based GO TL can be implemented in the middleware between the CPS application and the native networking TL. Thus, no hardware modification or updates are required, enabling easy deployment in practical setups with different wireless equipment. Installing a custom middleware still relies on software updates, which are, nevertheless, more feasible, owing to the programmability of modern OS-based Systems on Chip (SoC).[1]

Generally, GO approaches are evolving around the effectiveness concept, which refers to the contribution of a certain piece of data to the application goal [8]. For example, for CPSs that involve remote monitoring, the sensor can consider the expected improvement in the estimation accuracy when evaluating the effectiveness of recent measurements. Certain network structures, e.g., fixed delay setups with Bernoulli dropouts, allow for the analytical modeling of the effectiveness [3]. However, for general TL with unbounded and correlated delays and packet losses, the propagation of data through the network and its exact influence on the receiver in the future are difficult to track. We propose utilizing a data-driven approach to empirically capture the effect of transmitting the information on the future application dynamics from experience. In particular, the decision maker of our proposed GO TL is a Reinforcement Learning (RL) agent. It is trained to identify how the sensor's decision to transmit or discard detected data affects future system dynamics and which decision is more beneficial to the application, given available network resources.

RL is often utilized when dealing with decision makers interacting with dynamic environments, especially if the environment's state space is large. For example, multiple recent studies [9]–[11] integrate RL into congestion control mechanisms at TL, where the decision about the transmission rate is made to fit the available bandwidth. Even when assisted by RL, the existing TL approaches optimize the sending rate for conventional networking metrics, such as throughput and delay, which do not always translate to superior application performance [12]. The CPS performance is directly considered in the control theory field, often referring to RL to solve evolutionary tasks and optimize the control strategy. For Networked Control Systems (NCSs), communication cost is modeled as an additional term in the optimization objective. RL is utilized to jointly design transmission triggering and control policies [4]. However, such works normally do not include a realistic network but rather assign each transmission a constant cost. The architecture we propose in this work considers both the congestion status and explicit application performance. Therefore, the GO RL-based TL (**GO RL TL**) can be applied in realistic setups and minimize network adverse effects on the control efficiency in CPS.

An important practical issue of RL-based approaches applied for real-time control is the fact that the RL agent often cannot train online using the physical plant directly, because suboptimal decisions can irreversibly destabilize the system. Constructing the model of the physical environment, including its communication counterpart, requires considerable effort and should be individualized for each networking setup. In this work, we propose a framework that uses collected network traces to build an offline environment for the RL agent by recording possible state transitions in the historical data, depending on the transmission decisions[2]. This method generalizes to arbitrary communication scenarios and enables capturing complicated network-application interdependencies in an empirical way.

The contributions of this work are summarized as follows:

- We propose a semantic RL-based transport layer design **GO RL TL** that targets the real-time CPS performance improvement, without restricting to a particular application or network structure. An RL agent integrated within the CPS sensor decides whether to transmit or discard sensed real-time process measurements based on their effect on the control process evolution and the network congestion level.
- The proposed framework solves practical issues of deploying RL models by designing an offline environment for training from the collected data traces. As a result, the trained model can adapt to varying network conditions present in the traces. The collected data traces, as well as the source code for the offline environment, training, and deploying the model are available in the open-source repository [13].
- We perform an extensive experimental testing of the **GO RL TL** using wireless IIoT Zolertia sensors [14], implementing an IEEE 802.15.4 network stack [15]. We test various control policies and scenarios with dynamically changing congestion level. The proposed method is able to outperform all the State of the Art (SotA) approaches and demonstrates superior adaptation capabilities.
- To effectively manage SoCs constituting the majority of CPS deployment, we present the guidelines on how the Device Management entity can select a preferred TL configuration. The choice depends on the SoC's hardware capabilities and required application performance, and explores the RL model's memory footprint-effectiveness tradeoff. Considering multiple available SoC models, we validate the feasibility of **GO RL TL**. Interestingly, for particularly low storage capabilities, the **GO RL TL** is not recommended.

[1]For instance, from the SoCs discussed later and listed in Table I, the majority are OS-based and support user-defined applications, simplifying the software integration of the middleware TL.

[2]Note that online training necessitates suboptimal exploration decisions in the deployment phase. Initial data collection for the offline environment is more flexible and can be done prior to deployment, as we discuss in Section V-D. Furthermore, the offline environment enables tuning the model on the run without affecting the real CPS.

II. Related Work

Utilizing RL for congestion control is becoming more evident in recent research works and implementations [9]–[11]. In contrast to conventional congestion control relying on a set of pre-defined rules, RL-based schemes can be more effective when adapting to non-stationary scenarios. Moreover, exploiting a flexible reward design, the sending rate for various streams can be adapted according to their specific requirements. These requirements, though, are still formulated using conventional network metrics such as latency, reliability, and throughput. Networking RL-based congestion control schemes do not explicitly consider application performance, as semantic approaches require.

When referring to control theory, the RL here is widely utilized to define policies optimizing specific control objectives. To limit the transmission frequency in NCSs, the communication cost term can be integrated into the optimization objective [4], [7], [16]. Such approaches, referred to as event-triggering, normally cannot be extended to control the network congestion level and do not include the influence of network adverse effects on the real-time applications, which cannot be ignored in general networking setups.

For the scenarios with predictable transmission outcomes, i.e., for scheduling transmissions within time slots, the works [17]–[19] propose centralized RL approaches maximizing application performance. In contrast, the works [20]–[22] tackle the distributed transmission control design using multi-agent RL (MARL). [21] integrates constant communication cost into the objective, while agents are pre-trained with multiple cost values to generalize for various congestion levels. Both training and testing are performed in the simulator, but control loops do not influence each other through the network during training. In the tests with multiple loops, the communication cost converges to the value where the loops also do not affect each other. Therefore, it is not clear how the approach proposed in [21] will perform in real scenarios where this mutual influence persists. The work [22] mentions the ability to generalize for different networking scenarios. However, it performs training and testing in the same simulator, which puts the efficiency of the proposed distributed transmission schemes in the real world in question. Incorporating non-negligible network delays and losses is the main difference of this work from [21], [22] because of the time causality that changes the environment construction.

Similar feasibility issues are common for all the RL-based transmission techniques for control applications. RL methods require either an online setting or a precise environment simulation [23]. An online setting is used for RL-based congestion control, but it is not feasible for real-time mission-critical CPSs. The abstract simulators cannot assure the translation of the superior RL performance into real life. Detailed simulators have to be tailored to a specific scenario and do not generalize. In this work, we solve this issue by using historical data to construct an offline environment model. Such a design not only enhances the practical feasibility of the proposed **GO RL TL**, but also introduces versatility in a wide range of applications and networking setups.

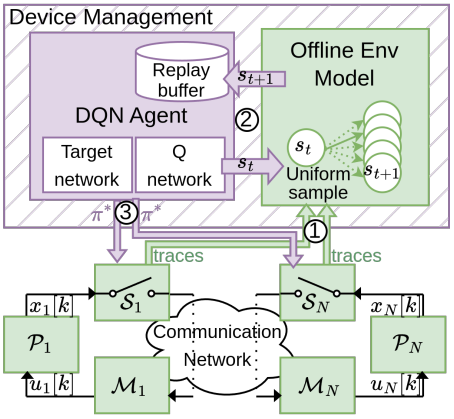


Fig. 2: Framework overview, including 1) pre-training (traces collection); 2) offline training; and 3) model distribution and deployment.

III. System Model

We consider a set of N dynamic processes, where the remote sensors $\mathcal{S}_i$ observe the real-time state of the physical plants $\mathcal{P}_i$, as illustrated in the bottom of Fig. 2. The sensors transmit observations to the monitors $\mathcal{M}_i$ over an arbitrary network, where the received data is used for estimation of the current process state and possibly for further plant actuation to the desired set point. This scenario is the generalization of the setups we considered in our previous work [12], [24]. The typical CPS use-cases include cruise [21], process, temperature control, robotics [22], and trajectory tracking [6]. 3GPP specifies delay requirements of $10 - 40$ms for such real-time systems. The discrete plant state dynamics have the following form:

$$\boldsymbol{x}_i[k+1] = f(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k], \boldsymbol{\omega}_i[k]), \tag{1}$$

with $\boldsymbol{x}_i[k] \in \mathbb{R}^n$ denoting the state of the plant $\mathcal{P}_i$ at time step k, $\boldsymbol{u}_i[k] \in \mathbb{R}^m$ being a control input, and $\boldsymbol{\omega}_i[k] \in \mathbb{R}^n$ is a Gaussian process noise. Example process dynamics is illustrated in Fig. 1.

Due to network adverse effects, such as communication delays and losses, the monitor is not guaranteed to receive all actual state measurements. Therefore, it builds an estimation of the current state based on past measurements. If at the time step k, the controller receives a measurement $\boldsymbol{x}_i[\nu(k)]$ generated at time step $\nu(k)$, it updates an Minimum Mean Squared Error (MMSE) estimation $\hat{\boldsymbol{x}}_i[t]$ of the plant state for time steps $t \in [\nu(k), k]$ as follows:

$$\hat{\boldsymbol{x}}_i[t] = \begin{cases} \boldsymbol{x}_i[t], \text{ if } t = \nu(k), \\ \mathbb{E}[f(\hat{\boldsymbol{x}}_i[t-1], \boldsymbol{u}_i[t-1], \boldsymbol{\omega}_i[t-1]) | \\ \qquad\qquad |\hat{\boldsymbol{x}}_i[t-1], \boldsymbol{u}_i[t-1]], \text{ otherwise.} \end{cases} \tag{2}$$

In other words, the controller uses a fresh state as an estimation if it is available. Otherwise, it propagates the last available state according to the process model (1) to get an MMSE estimation for the current time step.

If the monitor's ultimate objective is to drive the plant to a reference setpoint, the controller uses current estimation $\hat{\boldsymbol{x}}_i[k]$ to define an actuation input $\boldsymbol{u}_i[k]$. We do not enforce a particular control algorithm to determine $\boldsymbol{u}_i[k]$. The control strategies used for our evaluations are given in Section V.

In the context of real-time monitoring and control, the receiver benefits from more timely sensor data because this data refines the estimation and facilitates efficient actuation. On the other hand, each transmission consumes network resources, potentially increasing the congestion level and disturbing the timely delivery of other measurements that can be of higher significance. An overarching optimization problem lies in defining a **distributed goal-oriented transport layer admission scheme π applied by the sensor to filter the most significant data for the application, given the available network resources**. At each time step, the sensor $\mathcal{S}_i$ calculates $\delta_i[k] = \pi_i(s_i[k]) \in \{0, 1\}$ that dictates the decision regarding admitting $\boldsymbol{x}_i[k]$ to the underlying network. Here, $s_i[k]$ is the overall process state perceived by the sensor. Admitted packets are passed down the network stack to the networking TL responsible for establishing and keeping the connection to the receiver. The GO TL cannot further affect the status of the transmitted packet, meaning that the proposed mechanism does not induce any hardware modifications to the setup. Upon receiving a packet containing the state measurement, the controller issues the transmission of the reception acknowledgment (ACK). Similar to data packets, ACK packets can be delayed or lost in the network. Packets rejected by the goal-oriented TL are discarded without further consideration.

We do not put any limitations on the network structure. The only requirement is the ability of the sensor to interface to the networking TL and acquire the current network status of previously transmitted packets, i.e., whether they are already ACKed, considered lost due to ACK timeout, or their status is yet unknown[3].

IV. Reinforcement Learning Model

The sensor's decision to accept or reject a sample affects the overall system state, including not only future receiver's actions in this particular loop, but also the network congestion state and, consequently, other network users. This dynamic system can be described by the Markov Decision Process (MDP) model defined by the tuple $(\mathcal{S}, \mathcal{A}, P_a(s_t, s_{t+1}), R_a(s_t, s_{t+1}))$. Here, $\mathcal{S}$ defines a state space, $\mathcal{A}$ is a set of admissible actions, $P_a(s_t, s_{t+1})$ is the transition probability between states depending on the taken action, and $R_a(s_t, s_{t+1})$ is the one-stage reward. The ultimate goal of the decision maker is to maximize the cumulative discounted reward. The achieved maximum is defined as:

$$Q^*(s_0) = \max_{a_t = \pi^*(s_t)} \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1}} P_{a_t}(s_t, s_{t+1}) R_{a_t}(s_t, s_{t+1}), \tag{3}$$

where γ is a discount factor. The policy of interest $\pi^*(s_t)$ achieving $Q^*(s_0)$ can be found by solving the functional Bellman equation:

$$Q^*(s) = \max_a \sum_{s'} (P_a(s_t, s_{t+1}) R_a(s_t, s_{t+1}) + \gamma Q^*(s_{t+1})). \tag{4}$$

(4) is solved iteratively via updating $Q(s)$ with currently optimal actions until the left and right sides of (4) converge.

Next, we define the components of MDP for the considered CPS setup. Sensors' decisions at TL are done locally in a distributed manner, with no interaction between sensors, apart from mutual influence through the communication network. Therefore, the action space $\mathcal{A}$ is defined by $\delta[k] \in \{0, 1\}$ for accepting or discarding the recent state measurement.[4] To design a semantic TL facilitating the application goal and congestion control, the reward includes two terms:

$$R_a(s_t, s_{t+1}) = R_{a,app}(s_t, s_{t+1}) + R_{a,comm}(s_t, s_{t+1}), \tag{5}$$

where $R_{a,app}(s_t, s_{t+1})$ reflects a contribution of the decision a into the monitoring or control objective and $R_{a,comm}(s_t, s_{t+1})$ is a communication cost for increasing the congestion level.

Individual sensors cannot observe the overall system state, because they do not have access to all the components of the underlying network. The network state is defined through the current and past decisions of all the network users and communication mechanisms along the network path. Therefore, the considered MDP process is partially observed (POMDP), and sensor observations are not necessarily Markovian[5]. Such systems can be tackled by including a context and defining an MDP state by the window of past observations and decisions [25]. As a result, a POMDP is approximated with a classical MDP, solved with conventional optimization methods. We define a state of the approximate MDP at time step k as:

$$s_k = (p[k-W], p[k-W+1], \ldots, p[k-1], T[k], \boldsymbol{x}[k]), \tag{6}$$

including the tags $p[t]$ for past TL decisions in the window of W time steps and corresponding transmission statuses:

$$p[t] = \begin{cases} 0, & \text{if } \delta[t] = 0, \\ 1, & \text{if } \delta[t] = 1 \text{ and } \boldsymbol{x}[t] \text{ is ACKed}, \\ 2, & \text{if } \delta[t] = 1 \text{ and } \boldsymbol{x}[t] \text{ is } \textbf{not} \text{ ACKed}. \end{cases} \tag{7}$$

$T[k]$ reflects the current network state in terms of achieved throughput over the window $W_T >> W$, i.e., W_T is sufficiently large to capture the bandwidth available to a certain sender-receiver pair:

$$T[t] = \frac{\sum_{i=t-W_T}^{t-1} p[i] | (p[i] = 1)}{\sum_{i=t-W_T}^{t-1} d[i] | (p[i] = 1)}, \tag{8}$$

where $d[i]$ is the delay of the packet containing $x[i]$ extracted from the corresponding ACK. $T[t]$ is the total packet data rate

[3]Note that triggering ACK at the receiver can also be done by the application process, i.e., we do not assume any reliability mechanism for the underlying networking TL.

[4]From now on, we omit the index identifying a control loop for the sake of notation simplicity.

[5]These are the reasons for stability issues if the problem is tackled by MARL with an objective to achieve the global control performance, and the decision space being the combination of all the local decisions. Another complication is a variable number of agents. Therefore, similarly to [21], we solve the problem independently for a single agent, while the decisions of others are part of the stochastic environment evolution.

over W_T in pkt/ms. We stress that the state only includes the variables directly observed by the sensor.

In (5), the application counterpart $R_{a,app}(s_t, s_{t+1})$ represents the effectiveness of s_{t+1} w.r.t. the application goal. For instance, if the monitoring task is considered, the sensor uses s_{t+1} to infer the expected controller estimation error. In this work, we focus on real-time control, and the application performance is measured by the deviation of the plant state $\boldsymbol{x}[t+1]$ from the reference.

To prevent network over-utilization when admitting the packet, we define $R_{a,comm}(s_t, s_{t+1})$ to be proportional to the instantaneous sending rate. With that, we reduce bursts of admitted packets, detrimental to real-time applications due to network buffering. If the sensor observes a lower bandwidth, it should be more conservative when admitting further packets. Therefore, $R_{a,comm}(s_t, s_{t+1})$ is inversely proportional to $T[t+1]$:

$$R_{a,comm}(s_t, s_{t+1}) = -a \times \frac{\sum_{i=t-W}^{t_1} p[i] | (p[i] > 0)}{T[t+1]}. \quad (9)$$

As follows from (9), discarding the packet, i.e., $a = \delta[t] = 0$ does not incur communication cost.

The last component of MDP is the transition model $P_a(s_t, s_{t+1})$. For realistic network setups, building the detailed environment model is not feasible. RL can handle model-free scenarios by learning the transitions via direct interactions with an environment. When designing a TL as an enhancement for existing CPS setups, suboptimal interactions can disturb the stability of the involved control loops. Moreover, online RL would require the sensors to perform training locally, demanding considerable computing resources. Real-life deployments often feature low-cost SoCs that do not provide such capabilities. We address these issues by designing a data-driven offline environment from past traces collected from the considered deployment. The training of π^* can be done using remote compute resources, i.e., in the Device Management entity in the edge or cloud server.

In detail, we reshape the collected dynamics of $\{p[t]\}$, quantized $\{\boldsymbol{x}[t]\}$, and $\{T[t]\}$ to form the states s_t and record all the consecutive (s_t, s_{t+1}) pairs. For each unique s_t, we record the list $L(s_t)$ of observed s_{t+1}. To limit the memory footprint of the model, the lists containing more than 100 entries are truncated, so their probability distribution is maintained. The resulting dictionary $\{s_t, L(s_t)\}$ represents an empirical environment model. Real transitions are emulated by sampling a random entry from $L(s_t)$, without the need for direct interaction with the environment.

The state dimensionality increases exponentially with W, constraining the usage of dynamic programming to find exact π^*. It would require iterating over all the states at each optimization round. Moreover, policy and value iteration are sensitive to potential model errors w.r.t. the real environment. These facts motivate an offline RL approach that learns the Q-function iteratively from a subset of sampled state transitions.

The overall framework is depicted in Fig. 2. The sensors collect observation traces, including process dynamics and ACK statistics, before deploying an RL agent, and transfer them to the Device Management entity. This data is used to build an offline environment for training. We stress that the environment reflects the network configurations available in data traces. For instance, if the data were collected in scenarios featuring both scarce and unconstrained network resources, the offline model would be able to emulate typical system evolution in any of the present settings.

The offline Q-network maps each possible system state s_t to the Q-function value $Q^*(s_t)$. Due to high state dimensionality, we use a neural network for this mapping. The resulting deep Q-network (DQN) [26] includes a replay buffer that stores sampled transitions and serves as a source of uncorrelated inputs for training the target and Q neural networks. The target network is updated less frequently than the Q-network to facilitate learning stability.

The distributed trained policy π^* that maps the state to a binary action - an admission decision - is populated back to all the sensors. Now, the sensors can make an instantaneous TL decision based on the real-time observed system state. We envision that the sensor can continue collecting data traces, so the offline model can be augmented if the network conditions drastically change, and the agent can be post-trained accordingly. However, implementing such a setup is left for future work.

V. Use-case Scenarios

In our evaluations, we focus on the use-case of controlling a linear plant, where the monitors $\mathcal{M}_i$ aim to drive the plant to the zero set-point by applying a control input $\boldsymbol{u}_i[k]$, as illustrated at the bottom of Fig. 2. Linear models are widely utilized in control theory research and in practice due to their simplicity and efficiency [3], [7], [17]. Keeping the plant in the desired position is an ultimate real-time application objective assisted by **GO RL TL**. To demonstrate the versatility of our approach against various application parameters, we explore two types of control laws: a Linear Quadratic Gaussian (LQG) controller [27] and a Proportional-Integral-Derivative (PID) controller [28].

We use Linear Time-Invariant plant state model that reads as:

$$\boldsymbol{x}_i[k+1] = \boldsymbol{A}_i \boldsymbol{x}_i[k] + \boldsymbol{B}_i \boldsymbol{u}_i[k] + \boldsymbol{\omega}_i[k], \quad (10)$$

where $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are state and input matrices defined by the application. Under this model, the MMSE estimation takes the following form:

$$\hat{\boldsymbol{x}}_i[k] = \boldsymbol{A}_i^{\Delta_i[k]} \boldsymbol{x}_i[\nu_i(k)] + \sum_{q=1}^{\Delta_i[k]} \boldsymbol{A}_i^{q-1} \boldsymbol{B}_i \boldsymbol{u}_i[k-q], \quad (11)$$

where $\Delta_i[k] = k - \nu_i(k)$ is an instantaneous Age of Information (AoI) at the receiver. [6]

A. LQG controller

The LQG controller is proven to be optimal to minimize an LQG cost under scenarios satisfying the separation principle,

[6] AoI captures timeliness and is one of the semi-semantic metrics widely used for real-time applications. In this work, we compare the effectiveness of the AoI-based schemes to the methods considering the information content beyond age.

i.e., where the observer and the controller function can be designed individually. The LQG cost is defined as:

$$\mathcal{J}_i \triangleq \limsup_{\mathcal{T}\to\infty} \left(\frac{1}{\mathcal{T}} \sum_{k=0}^{\mathcal{T}-1} (\boldsymbol{x}_{i,k})^T \boldsymbol{Q}_i \boldsymbol{x}_{i,k} + (\boldsymbol{u}_{i,k})^T \boldsymbol{R}_i \boldsymbol{u}_{i,k} \right), \tag{12}$$

where $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ are weighting parameters. The control law minimizing (12) reads as:

$$\boldsymbol{u}_{i,k} = -\boldsymbol{K}_i \hat{\boldsymbol{x}}_{i,k}, \tag{13}$$

in which $\boldsymbol{K}_i$ can be calculated numerically [27]. The LQG controller is not optimal in many realistic scenarios, e.g., when there is no perfect monitor-sensor ACK link[7]. A suboptimal LQG controller is still popular due to its straightforward design and usage. The **Go RL TL** is designed to improve application performance in real setups. Thus, the demonstration of the enhancement of the ultimate LQG control performance has a significant practical value. A similar conclusion applies to a PID controller, one of the most popular controllers for industrial applications.

B. PID controller

PID controllers are popular within a wide range of scenarios due to their simplicity of operation and tuning. The actuation input of the PID controller includes three terms:

$$\boldsymbol{u}_i[k] = -\boldsymbol{K}_{p,i}\hat{\boldsymbol{x}}_i[k] - \boldsymbol{K}_{i,i} \sum_{t=0}^{k} \hat{\boldsymbol{x}}_i[t] - \boldsymbol{K}_{d,i}(\hat{\boldsymbol{x}}_i[k] - \hat{\boldsymbol{x}}_i[k-1]). \tag{14}$$

The first proportional term is tuned by the $\boldsymbol{K}_{p,i}$ parameter. It eliminates existing state error w.r.t. the zero set-point. The second integral term defined through $\boldsymbol{K}_{i,i}$ corrects the remaining accumulated steady-state errors. Finally, the differential term set by $\boldsymbol{K}_{d,i}$ dampens the applied force. The utilized PID parameters are available at [13].

C. Communication scenario

GO RL TL we propose in this work is integrated through middleware and generalizes for different communication networks. For the evaluation, we use a real setup featuring low-power, low-cost Zolertia ReMote wireless sensors [14] tailored for industrial applications. The sensors implement an IEEE 802.15.4 communication stack [15] at the Medium Access Control (MAC) and lower OSI layers, supporting up to 250 kbps data rate. A contention-based access scheme implies a strong mutual influence of individual control loops on each other through the communication network. The sensors attempt to access the wireless channel randomly, leading to collisions, which, in turn, magnify the experienced delays and packet drop rates [8].

The application dynamics for each loop are emulated on the PC as independent processes. One process is for the plant and the sensor sampling its state, and one for the controller. Every 10ms, sensors form a separate packet with a current state. As we have shown in [12], for such fast dynamics, the sensors have to discard some of the observed data. Otherwise, the generated traffic overwhelms the network, and the loops are destabilized. **GO RL TL** prioritizes more significant data that is pushed to the ReMote associated with the sensor for further wireless transmission to the ReMote corresponding to the controller. All the data received by the second ReMote is transferred to the controller process on the PC, which performs real-time plant actuation. The controller Zolertia replies with an ACK after each successful reception.

D. Training Framework

The data for the offline environment model is collected by running 1 to 5 LQG or PID control loops, with sensors randomly accepting state measurements. We vary average traffic generation rates to capture all the feasible congestion levels. It is essential that collected traces capture various network conditions and the effect of different traffic patterns on the congestion level. An important direction for future work is defining the boundaries of RL generalization, i.e., exploiting trained **GO RL TL** with more real loops than have been present in the traces. We envision that instead of running the real control process, one can collect the network-related statistics, i.e., the transmission timestamps and the delay samples. Using these inputs, the corresponding application dynamics can be reconstructed with the help of the available control emulators. In that way, one can avoid operating a mission-critical system with a suboptimal transmission scheme. If the **GO RL TL** scheme is integrated into the existing setup, previously collected network traces can be reused.

We train one agent for the LQG controller and one for the PID controller, which can be utilized for any network configuration. Target and Q-networks within the DQN Agent are deep neural networks with 3 fully connected layers, 2 activation and 2 normalization layers in between. The model parameters, as well as other relevant numerical values, are given at [13], together with the collected data and model source code.

VI. Experimental Results

To analyze the performance of the proposed RL TL, we compare it to the following benchmarks:

- Zero-Wait Event-Triggering (**ZW ET**) accepts the updates to the network when the deviation between the measured state and the augmented controller estimation exceeds the threshold [9]. To avoid potential buffering, further packets are not accepted when there are active Outstanding Packets (OPs) awaiting ACKs. We have proposed **ZW ET** as a proof of concept for GO TL in our previous work [12].
- Age Control Protocol (**ACP**) is a SotA TL scheme that adapts the sending rate to minimize AoI at the receiver [5]. Using ACKs, **ACP** tracks the changes in the amount

[7]We refer the reader to our previous work [12], [24] for more detailed discussion on applying GO TL with LQG controllers.

[8]Testing different network scenarios, i.e., control over the Internet, is an important direction for future work.

[9]The sensor replicates the controller process to augment its current estimation. From the sensor perspective, the controller is aware of ACKed packets only. Uncertainty regarding the exact controller state creates a mismatch between the augmented by sensor and real estimation.

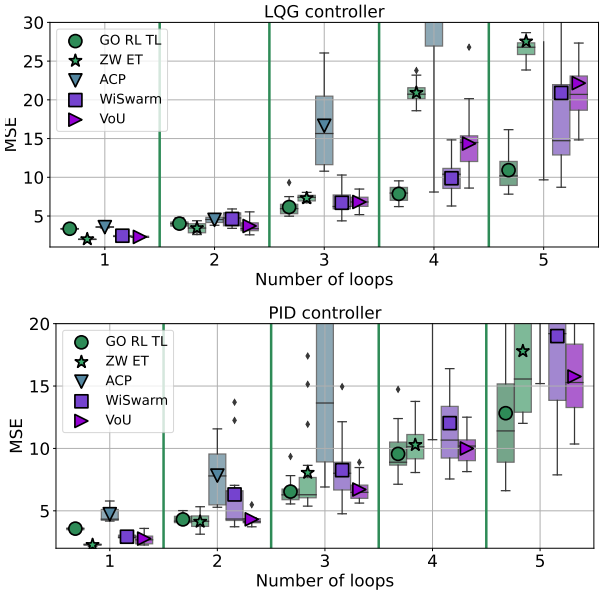


Fig. 3: Experimental MSE of benchmark TLs compared to the proposed **GO RL TL** when the number of control loops is fixed throughout a run.

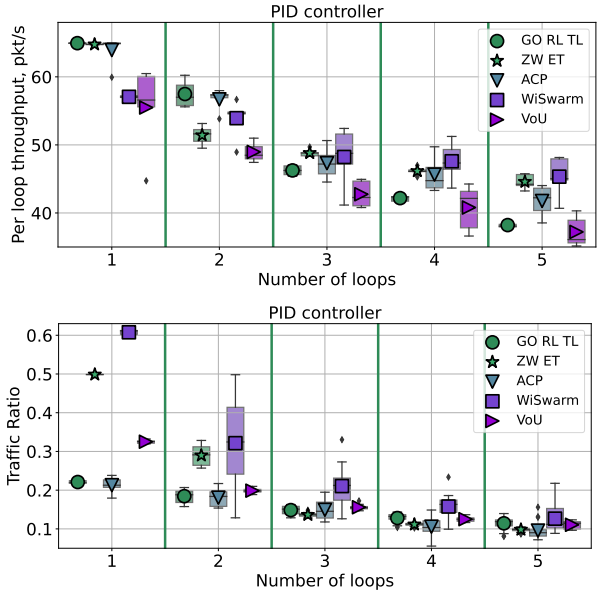


Fig. 4: Experimental throughput per loop and the ratio of transmitted traffic w.r.t. all the sensed data of benchmark TLs compared to the proposed **GO RL TL**.

of backlogged packets and adjusts it to reduce AoI. This scheme only considers data timeliness and does not include its effectiveness.

- **WiSwarm** is a SotA approach [6] designed for the TL scheduling over WiFi. The centralized entity, aware of the exact state of all the sensors, grants the transmission opportunity to one loop at each time step, prioritizing maximum Whittle index, a function of AoI and exact estimation error. This scheme cannot be integrated into distributed setups and serves as an interference-free benchmark.
- Value of Information (**VoU**) - based TL is the scheme we proposed in [24]. There, the sensor evaluates the efficiency of sampled updates by augmenting the estimation error and compares it to a dynamic congestion-aware transmission cost scaled by the threshold. **ZW ET** disregards the OPs' influence on the augmentation. In contrast, **VoU** uses an abstract network model to estimate the network status of OPs and their influence on estimation.

Note that in [12], we have shown that the conventional schemes, such as TCP, UDP, and classical event-triggering from control theory, do not sustain system stability under the same network conditions as in the current work.

As an application performance metric, we consider the mean squared error (MSE) of state deviation from the zero set point. For each scheme, we perform 5 simulation runs, each by 5000 time steps, i.e., 50s. As a result, for each control loop, for each run, we get:

$$MSE = \frac{1}{4001} \sum_{k=1000}^{5000} \boldsymbol{x}_i[k]^2, \tag{15}$$

where the first 1000 time steps are excluded to eliminate transient phase influence.

A. *Static number of control loops*

In the first set of experiments, we fix the number of active control loops. The results are shown in Fig. 3 and 4. For the threshold-based schemes (**ZW ET**, **VoU**), we manually pick the best-performing threshold for a given number of loops[10]. **GO RL TL**, **ACP**, and **WiSwarm** adapt to the instantaneous network state and do not require adjusting the parameters.

Similar trends can be observed for both controller types: LQG and PID. First of all, **ACP**, the only method not considering the data effectiveness, leads to the worst application performance, especially with a high congestion level. Nevertheless, as shown in Fig. 4, **ACP** maintains a conservative sending rate, preventing congestion and consistently showing high throughput. Therefore, optimizing for conventional networking metrics such as throughput does not ensure the application performance. All other techniques keep MSE limited for up to 5 control loops. More advanced triggering method **VoU** consistently outperforms **ZW ET**, suggesting that a more elaborate analysis of effectiveness-transmission cost tradeoff is a promising approach in GO TL. On average, **VoU** allows for more traffic, sacrificing the throughput, and therefore higher congestion level and higher delays. Nevertheless, by getting more important status updates through the network, **VoU** improves MSE. Centralized **WiSwarm** naturally adapts to a varying number of users requesting resources. Combined with the application weights explicitly including the estimation error, **WiSwarm** provides a competitive performance. The transmission coordination enabled by scheduling puts both the traffic amount and the throughput to a maximum among other methods, witnessing effective congestion avoidance. Unfortu-

[10]For higher congestion levels, i.e., more loops, a greater threshold value assures that the sensor maintains a conservative sending rate, and only the most significant data is transmitted.

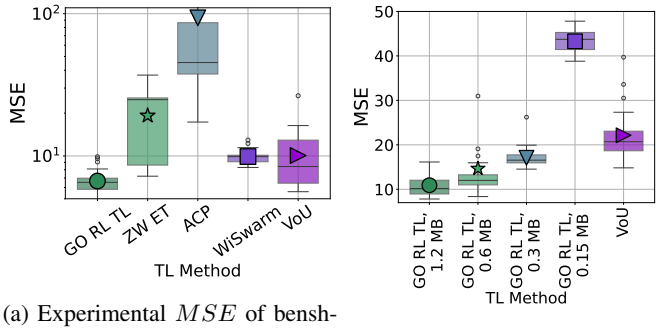
nately, **WiSwarm** cannot be used in arbitrary scenarios where a centralized scheduler and instant access to estimation errors cannot be provided.

Most importantly, fully distributed **GO RL TL** outperforms all other techniques for $N \geq 3$. If the network resources are over-provisioned, e.g., with $N = 1$, the best performance is achieved when all the state measurements are admitted to the network. **GO RL TL** does not cover this corner case and, therefore, performs slightly worse. The reason is that the model is unified for varying congestion levels. Even with mild network constraints, transmitting bursts of packets can result in the complete loop destabilization. In contrast, lowering the sending rate w.r.t. the optimal does not have such a significant effect. The RL model behaves more conservatively to avoid drastic performance degradation. To sum up, the ability of RL to generalize comes with suboptimal performance for corner cases. Nevertheless, for more scarce network resources, the **GO RL TL** scheme shows at least 20% better MSE than other benchmarks for the PID controller and almost 100% better for the LQG controller. Similar to **VoU**, the **GO RL TL** optimizes the transmission decisions to minimize future MSE. The network-related performance of **GO RL TL** and **VoU** is also similar, i.e., higher congestion level as the price for more updates being delivered and better resulting MSE. The difference is that the data-driven offline environment used for RL captures the update effectiveness even if the corresponding packet faces delays and losses that are hard to model analytically[11]. Moreover, **GO RL TL** does not require threshold adaptation to a particular congestion level, as opposed to **VoU**. To sum up on the evolution of our own work, the superior ability of **VoU** w.r.t. **ZW ET** and of **GO RL TL** w.r.t. **VoU** to generalize for different network conditions translates into better application performance of the corresponding TL methods.

B. Dynamic number of heterogeneous control loops

In the next experiment, we test the adaptability of the different TL schemes to dynamic network conditions. Moreover, instead of homogeneous controllers of the same type, the new setting features heterogeneous applications. In more detail, the maximum number of simultaneously active control loops is fixed to 5, 2 of which implement a PID controller and 3 - an LQG controller. An application process within each loop wakes up and suspends according to random patterns following an exponential distribution. As a result, the number of active loops and the congestion level change randomly over time. homogeneous For **ZW ET** and **VoU**, we pick the thresholds conservatively, such that the applications do not destabilize for the maximum expected congestion. For **GO RL TL**, the sensors within PID and LQG loops deploy the models trained with PID or LQG data, respectively. Note that for the offline environment built for the PID controller, no data from the mixed setup has been used, and vice versa. Therefore, this

[11]Note that **WiSwarm** avoids the necessity for such modeling due to scheduled transmissions with constant one-slot delays.



(a) Experimental MSE of benshmark TLs compared to the proposed **GO RL TL** for a dynamic setup.

(b) The MSE performance reduction if **GO RL TL** model is truncated.

experiment verifies whether the RL model can generalize to application configurations not present in the traces.

The results presented in Fig. 5a show that **GO RL TL** outperforms **WiSwarm** and **VoU** by 30%, whereas **ZW ET** and **ACP** perform considerably worse. **VoU** results in a significant spread in recorded MSEs, meaning that some control loops experience much higher state deviations. The **WiSwarm** adaptation capability is natural due to the presence of the centralized scheduler. The proposed fully distributed **GO RL TL** method exhibits consistently lower MSE while not requiring any modifications to the RL models. We conclude that building an environment that reflects the effect of different network conditions on the particular control application is sufficient to achieve high performance for the corresponding control loops in divergent scenarios, where other traffic can be initiated by arbitrary users.

C. Model Management in Practice

The superior **GO RL TL** performance comes with certain training and deployment overheads. Training requires network traces for building an offline environment, as well as server compute resources. The training time, which is $<$ 1h, does not bring significant overhead, since it is done offline. However, even if it is feasible to train the model, the local resources of the SoCs deploying the RL model may not be enough to operate it. A neural network within the RL model has $<$ 1ms inference time, which is not a significant addition to the overall latency and fits within a 10ms sampling period. For comparison, the inference is slower than the time required for the decision of **ZW ET**, **ACP**, or **WiSwarm**, but considerably faster than **VoU**, which takes up to 2ms [24].

Importantly, the model has to be stored in the sensor's RAM, which is often limited for low-cost devices. For instance, the full model size for our parameters is $\sim$1.2MB. If local RAM on SoCs is not enough, or if the deployment has mild MSE demands, the model's weights can be truncated without the need for retraining, and its volume can be compressed. Fig. 5b represents MSE performance when the full RL model is deployed, and when its size is truncated. Up to 4-x size reduction, the **GO RL TL** scheme still outperforms **VoU**. However, if the model is truncated too drastically, deploying **GO RL TL** is not beneficial, and the Device Management

SoC Model	Protocol Support	RAM	Extendable
RW612	WiFi, 802.15.4	1.2MB	8MB
CC355xE	WiFi	1MB	8MB
nRF54H20	802.15.4	1MB	No
ESP32-S2 / H2	WiFi, 802.15.4	320kB	8MB
SiWx917Y	WiFi	672kB	2MB
RTL8762G	802.15.4	384kB	No
nRF52840	802.15.4	256kB	No

TABLE I: Wireless SoCs supporting the WiFi or IEEE 802.15.4 protocols with different on-chip RAM capacity.

entity can suggest the **VoU** TL middleware instead, which has a much smaller memory footprint. Indeed, **VoU** only requires storing some ACK statistics, which do not have to be located in RAM, the most constrained memory. Additionally, simple models can be preferred over **GO RL TL** if it is expected that network resources remain overprovisioned. In that case, the straightforward **ZW ET** can show superior performance at no operational costs.

Table I shows some exemplary wireless SoCs with on-chip and extendable RAM capacity. It shows that a wide range of capacities are available from different vendors for both WiFi and 802.15.4 protocols. Many SoCs can natively accommodate the 2-x size reduced model of **GO RL TL**. Extendable RAM makes even a full-sized **GO RL TL** suitable for a wider range of SoCs.

VII. Conclusion

In the current work, we have presented a highly practical solution that enhances the networked CPS performance by 20 to 100%. Namely, we propose a semantic RL-based transport layer middleware that enables distributed filtering of real-time sensory data, facilitating both the CPS efficiency and end-to-end congestion control.The data-driven approach brings adaptability and versatility that are often missing in existing semantic communication mechanisms. However, particular deployments have to carefully consider the tradeoffs related to the need for data traces, as well as compute and storage demands for training and inference. An important direction for further work is online model adaptation to unforeseen deployment changes. We envision that the Device Management can continue training in the background using new data, and local models will be updated upon divergence. A pre-trained model that avoids network over-utilization can serve as an initialization and as a rollback in case of unseen network conditions. It is useful during training and model adaptation, which time is not negligible for an online setting.

References

[1] M. Bernas, A. Mykytyshyn, V. Kartashov, V. Levytskyi, and D. Martjanov, "The role of cyber-physical systems and internet of things in development smart cities for industry 4.0." in *CITI*, 2023, pp. 91–102.

[2] E. Uysal, O. Kaya, A. Ephremides, J. Gross, M. Codreanu, P. Popovski, M. Assaad, G. Liva, A. Munari, B. Soret *et al.*, "Semantic communications in networked systems: A data significance perspective," *IEEE Network*, vol. 36, no. 4, pp. 233–240, 2022.

[3] T. Chang, X. Cao, and W. X. Zheng, "A lightweight sensor scheduler based on aoi function for remote state estimation over lossy wireless channels," *IEEE Transactions on Automatic Control*, vol. 69, no. 3, pp. 1697–1704, 2023.

[4] N. Funk, D. Baumann, V. Berenz, and S. Trimpe, "Learning event-triggered control from data through joint optimization," *IFAC Journal of Systems and Control*, vol. 16, p. 100144, 2021.

[5] T. Shreedhar, S. K. Kaul, and R. D. Yates, "An age control transport protocol for delivering fresh updates in the internet-of-things," in *2019 IEEE WoWMoM*. IEEE, 2019, pp. 1–7.

[6] V. Tripathi, I. Kadota, E. Tal, M. S. Rahman, A. Warren, S. Karaman, and E. Modiano, "Wiswarm: Age-of-information-based wireless networking for collaborative teams of uavs," in *IEEE INFOCOM 2023-*. IEEE, 2023, pp. 1–10.

[7] A. Termehchi and M. Rasti, "A learning approach for joint design of event-triggered control and power-efficient resource allocation," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6322–6334, 2022.

[8] F. Alawad and F. A. Kraemer, "Value of information in wireless sensor network applications and the iot: A review," *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9228–9245, 2022.

[9] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3050–3059.

[10] X. Nie, Y. Zhao, Z. Li, G. Chen, K. Sui, J. Zhang, Z. Ye, and D. Pei, "Dynamic tcp initial windows and congestion control schemes through reinforcement learning," *IEEE JSAC*, vol. 37, no. 6, pp. 1231–1247, 2019.

[11] Y. Chen, H. Shi, Q. Weng, and Z. Shi, "Congestion control design of multicast quic based on reinforcement learning," in *2023 International Conference on Ubiquitous Communication*. IEEE, 2023, pp. 232–236.

[12] P. Kutsevol, O. Ayan, N. Pappas, and W. Kellerer, "Experimental study of transport layer protocols for wireless networked control systems," in *2023 20th Annual IEEE SECON*. IEEE, 2023, pp. 438–446.

[13] "Goal-oriented rl transport layer," 2025, the GitHub repository is available at: https://github.com/pkutsevol/GoRlTl.

[14] "Zolertia Remote: Lightweight Internet of Things hardware development platform," https://zolertia.io/product/re-mote/.

[15] "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, 2016.

[16] D. Baumann, J.-J. Zhu, G. Martius, and S. Trimpe, "Deep reinforcement learning for event-triggered control," in *2018 IEEE CDC*. IEEE, 2018, pp. 943–950.

[17] J. Holm, F. Chiariotti, A. E. Kalør, B. Soret, T. B. Pedersen, and P. Popovski, "Goal-oriented scheduling in sensor networks with application timing awareness," *IEEE Transactions on Communications*, vol. 71, no. 8, pp. 4513–4527, 2023.

[18] B. Demirel, A. Ramaswamy, D. E. Quevedo, and H. Karl, "Deepcas: A deep reinforcement learning algorithm for control-aware scheduling," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 737–742, 2018.

[19] A. S. Leong, A. Ramaswamy, D. E. Quevedo, H. Karl, and L. Shi, "Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems," *Automatica*, vol. 113, p. 108759, 2020.

[20] Y. Deshpande, O. Ayan, and W. Kellerer, "Improving aoi via learning-based distributed mac in wireless networks," in *IEEE INFOCOM 2022*, 2022, pp. 1–8.

[21] Z. Jiang, Z. Cao, S. Fu, F. Peng, S. Cao, S. Zhang, and S. Xu, "Revealing much while saying less: Predictive wireless for status update," in *IEEE INFOCOM 2020*. IEEE, 2020, pp. 1419–1428.

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

[23] A. Murad, F. A. Kraemer, K. Bach, and G. Taylor, "Information-driven adaptive sensing based on deep reinforcement learning," in *Proceedings of the 10th International Conference on the Internet of Things*, 2020, pp. 1–8.

[24] P. Kutsevol, O. Ayan, N. Pappas, and W. Kellerer, "Goal-oriented middleware filtering at transport layer based on value of updates," *arXiv preprint arXiv:2502.17350*, 2025.

[25] A. Kara and S. Yuksel, "Near optimality of finite memory feedback policies in partially observed markov decision processes," *Journal of Machine Learning Research*, vol. 23, no. 11, pp. 1–46, 2022.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] H. Kwakernaak and R. Sivan, *Linear optimal control systems*. Wiley-interscience New York, 1972, vol. 1.

[28] A. Visioli, *Practical PID control*. Springer Science & Business Media, 2006.