Goal-Oriented Scheduling in Sensor Networks with Application Timing Awareness

Josefine Holm, *Student Member, IEEE,* Federico Chiariotti, *Member, IEEE,* Anders E. Kalør, *Member, IEEE,* Beatriz Soret, *Senior Member, IEEE*, Torben Bach Pedersen, *Distinguished Contributor, IEEE*, and Petar Popovski, *Fellow, IEEE*

Abstract—**Taking inspiration from linguistics, the communications theoretical community has recently shown a significant recent interest in *pragmatic*, or goal-oriented, communication. In this paper, we tackle the problem of pragmatic communication with multiple clients with different, and potentially conflicting, objectives. We capture the goal-oriented aspect through the metric of Value of Information (VoI), which considers the estimation of the remote process as well as the timing constraints. However, the most common definition of VoI is simply the Mean Square Error (MSE) of the whole system state, regardless of the relevance for a specific client. Our work aims to overcome this limitation by including different summary statistics, i.e., value functions of the state, for separate clients, and a diversified query process on the client side, expressed through the fact that different applications may request different functions of the process state at different times. A query-aware Deep Reinforcement Learning (DRL) solution based on statically defined VoI can outperform naive approaches by 15-20%.**

I. Introduction

Semantic and goal-oriented communications [1] aim to go beyond the traditional domain of communication theory towards optimizing communication systems with respect to a specific task or goal. In [2], Shannon and Weaver talk about the *semantics* and *effectiveness* levels of the communication problem: semantic communication corresponds to the transmission of the most meaningful information for the given context, while effective or goal-oriented communication aims at satisfying the specific requests of the receiver. Following the nomenclature in linguistics, we denote the latter as *pragmatic* communication. In this sense, pragmatic communication is about transmitting the most relevant information for the receiver in order to attain a certain goal, taking into account both timing constraints and the shared context, which acts as an implicit information channel between the transmitter and receiver.

J. Holm (jho@es.aau.dk), F. Chiariotti (corresponding author, chiariot@dei.unipd.it), A. E. Kalør (aek@es.aau.dk), B. Soret (bsa@es.aau.dk), and P. Popovski (petarp@es.aau.dk) are with the Department of Electronic Systems, Aalborg University, 9220 Aalborg Øst, Denmark. T. B. Pedersen (tbp@cs.aau.dk) is with the Department of Computer Science, Aalborg University, 9220 Aalborg Øst, Denmark. F. Chiariotti is also with the Department of Information Engineering, University of Padova, 35131 Padua, Italy. A. E. Kalør is also with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong. B. Soret is also with the Telecommunications Research Institute (TELMA), Universidad de Málaga, 29071 Málaga, Spain (bsoret@ic.uma.es). This work was supported by the Independent Research Fund Denmark, Grants No. 8022-00284B "SEMIOTIC" and No. 1056-00006B, EU Horizon 2020 Grant no. 957345 "MORE," the Villum Investigator Grant "WATER" from the Velux Foundation, Denmark, and the SoE Young Researchers grant "REDIAL," as part of the European Union's Italian National Recovery and Resilience Plan (NRRP).

This new perspective is crucial in the context of the Industry 4.0 and Industrial Internet of Things (IIoT) paradigms, which aim at automating manufacturing and industrial processes in a flexible and easily reconfigurable fashion. A representative scenario is the remote estimation of the state of a system by a distributed network of low-power sensors, a classical Internet of Things (IoT) problem [3]. The recently proposed Value of Information (VoI) metric [4] represents a theoretical tool to model the pragmatics of the monitoring application, as it can seamlessly integrate the timing performance of the communication network with the underlying estimation [5]. However, VoI is often defined in a static manner: the value function is the Mean Square Error (MSE) of the system state, and there is an application constantly monitoring the process with a faster pace than the updates.

In many IoT scenarios, this is not true, as there may be *multiple* clients monitoring the same process through an edge node or gateway. Each client may potentially be interested in a different function of the system state at different times [6]. Different *queries* can then correspond to different functions of the state. A classic query is the sample average among the measured values, but non-linear functions and even order statistics can often be useful in industrial settings. For example, the number of state components that are within their normal operation parameters, or the maximum among the state components, can be helpful to trigger safety conditions and shut down machines or raise a warning to the operators. As another example, the sample variance can be useful when monitoring the strain on different components of a building or a structure, such as a bridge. In this case, the goal of the system is implicitly stated in the queries: instead of minimizing the estimation error on the state of the system as a whole, the objective is to reduce the error on the response to the *specific query* that a client makes. The dynamic nature of these queries, which may arrive at different times from different clients, is crucial in the optimization: a pragmatic transmitter needs to take into account not only the relevance of sensor information, but also which query might arrive next.

The general architecture we propose is shown in Fig. 1. The publish/subscribe model is already the standard for IoT applications [7], with the edge node acting as a message broker. In the system model in the figure, a Mobile Edge Computing (MEC)-enabled base station polls a set of sensors, which respond with their latest measurements. The edge node

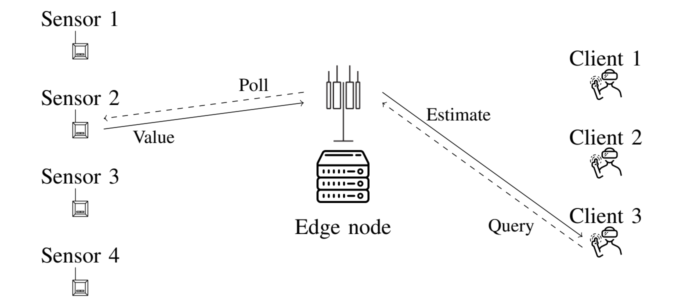
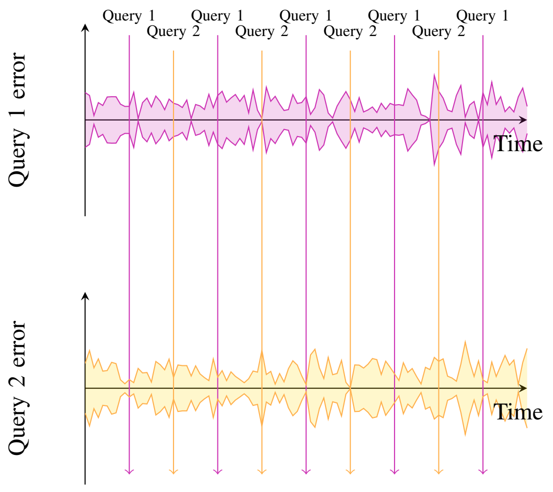


Fig. 1. Illustration of the considered monitoring scenario, where an edge node collects information from sensors to reply to queries from clients. Queries are indicated by dashed lines, while their responses are solid lines.

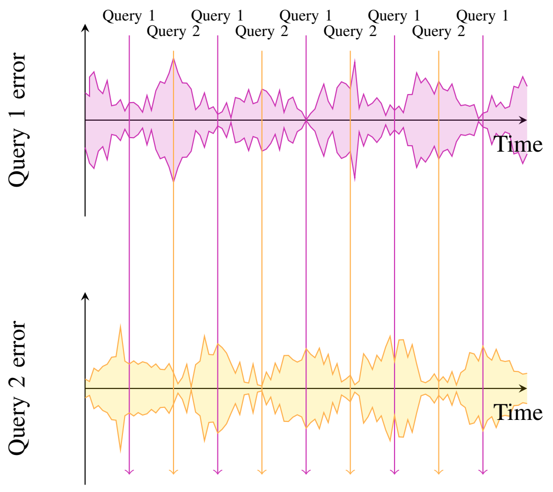
uses the data to estimate the overall state of the process measured by the sensors, and receives queries from client applications, which might be different functions of the state, e.g., the highest value among all sensors, or the number of sensors measuring values in a certain range [8]. In this work, we will consider that the edge node has enough computational power to run the estimation in real-time, but the interplay between the complexity of the scenario and the allocation of computing resources is an interesting problem that could complement our analysis [9].

This setup has a clear timing context for defining goal-oriented communication: the edge node should optimize its scheduling strategy to answer queries accurately, taking into account both the nature of the estimation process and the query process. The optimal VoI scheduling will then consider not only the *accuracy* of the estimate, i.e., the semantic value of the updates, but also *when* the estimation of a particular function of the state will be needed, i.e., the pragmatic relevance of the information to the receiver. In other words, the goal is not static, as in MSE minimization, but dynamic, as the edge node needs to anticipate upcoming queries and place more importance on updates that will help for those specific functions (e.g., if the edge node expects a client to ask for the highest temperature among all sensors, it should avoid polling sensors that have a high probability of having an extremely low value, even if doing so increases the overall estimation MSE over the state).

Fig. 2 shows a sketch of such an example. In the figure, we plot the instantaneous MSE over two hypothetical queries. Fig. 2a shows the effect of a VoI maximization strategy: the errors of both queries are approximately constant across time (as shown by the dashed line), with some fluctuations due to the dynamics of the system and channel errors, independently of the query process. On the other hand, a pragmatic system, as the one shown in Fig. 2b, can tolerate significant increases in the MSE of a query as long as the query is not active. Instead, it will aim to reduce the MSE of the next upcoming query, so that the error *at the time of the query* is minimized. While we expect some fluctuations due to the stochastic nature of these systems, the trend represented by the dashed lines is clear, and leads on average to better responses to specific queries, even if the error when measured at a random time instant is higher. Our previous work [6] devised a similar metric for Age of



(a) VoI-based sensor scheduling.



(b) Pragmatic sensor scheduling.

Fig. 2. Sketch of the effects of static VoI-based and pragmatic scheduling. The lines and colored areas represent the query response confidence interval.

Information (AoI), which can be measured at the arrival of a query instead of at any time. While the example is only an illustration of the ideal behavior of a query-aware pragmatic system, our results show that this behavior can indeed be obtained in practice.

In this work, we formulate a pragmatic[1] scheduling problem, considering both the process and the queries from multiple client applications. We define the problem as a Markov Decision Process (MDP), assuming that the edge node has the full statistical knowledge of the process, and solve it with Deep Reinforcement Learning (DRL), as the continuous state space prevents us from using simpler methods such as tabular Q-learning. However, the main goal of this work is not just to optimize in the specific setting we consider, but rather to provide insights on the novel problem of pragmatic communication, proposing a dynamic definition of VoI and using a relatively simple use case to understand it better. The contributions of the paper are the following:

[1]In the remainder of this paper, we use the terms *pragmatic* and *goal-oriented* interchangeably.

- We rigorously model the pragmatic sensor scheduling problem with multiple clients, different query functions, and different query generation processes, resulting in a dynamic objective function;
- We find a closed-form solution in a simple illustrative example, showing that the optimal policies for different queries can be extremely different and that using the wrong one can lead to significant performance loss;
- We frame the problem as a Partially Observable Markov Decision Process (POMDP), defining the state and observation space, the reward, and the actions available to the scheduler, and use DRL to obtain a scheduling solution, which is executable in real time even on an embedded edge processor;
- We provide simulation results and comparisons with static VoI policies and the traditional Maximum Age First (MAF) policy, which minimizes the average AoI of the system, for a scenario in which two query types are present, i.e., a maximum query asking the maximum value among the state components and a count range query asking how many state components have a value within a given interval. The performance of the system is strongly dependent on the queries that arrive to it, as the error on the queries depends on the exact function of the state that they request.

The rest of the paper is organized as follows: first, we discuss the state of the art on the topic in Sec. II. We then define the scheduling problem in Sec. III, and model it as an MDP in Sec. IV, which also describes the proposed DRL solution. Then, Sec. V describes the simulations comparing the proposed scheme to state-of-the-art policies and their results. Finally, Sec. VI concludes the paper and presents some possible avenues for future work.

II. Related Work

The problem of designing communication systems that communicate not only reliably, but also effectively and in a timely fashion, has recently received a significant amount of attention. During the last decade, AoI [10] and the associated metrics have received a significant attention in the communication engineering community. Most works in the literature study the average AoI in queues and networks of queues, using basic queuing theory to compute analytical performance curves [4]. However, the original definition of the AoI can result in suboptimal outcomes if some conditions are not met, and similar metrics that can extend the definition to more general scenarios and applications have been defined [11].

Firstly, AoI does not need to be linear: some recent works generalized the definition to measure any non-decreasing function of the age [12], [13], leading to different considerations in AoI optimization, as higher ages are penalized more or less depending on whether the aging function is super- or sub-linear. This can be further generalized by taking into account the actual value of the process tracked by the updates: since AoI is a proxy metric for the evolution of the tracking error over time, considering that error directly leads to better performance. The Age of Incorrect Information (AoII) [14] mixes a linear timing penalty with a multiplier based on the error in a discrete system, while VoI [15] directly measures the error (either real or expected) on the estimation process, whether over a continuous-valued variable or a Markov source [16].

The adaptive scheduling of sensors in IoT scenarios with the goal of minimizing the AoI or related metrics is a well-studied problem in the literature. The scheduling problem can be formulated both for multiple sources, in which case it involves balancing the ages of the different sources while avoiding interference [17], or for a single source with resource constraints: usually, these constraints are in the form of limited energy availability [18] or enforced duty cycles [19]. The inclusion of constraints on energy is often combined with energy-harvesting capabilities [20]. The most interesting works in this sense consider the VoI, measured as the expected tracking error of a Kalman filter [21], [22]. In general, the constraint on the accuracy is due to a communication bottleneck, which occurs due to limited bandwidth and energy, such that sensors need to reduce their transmissions as much as possible [23]. The selection of the subset of sensors that will transmit has been studied in this context [24], often considering the correlation between neighboring sensors' measurements [25]. Other scenarios in which VoI is used are data muling applications [26], in which drones, robots, or underwater vehicles need to physically move close to sensors to collect the information [27], and sensor placement problems, in which the issue is not to schedule transmissions, but rather to design the network to maximize accuracy and minimize cost [28]. Our own previous work [8] extends the definition of VoI from the MSE of the state to arbitrary functions, presenting a one-step optimal scheduling procedure. Another interesting development involves the modeling of the state of each sensor as a Markov chain, posing the polling problem as a POMDP [29] to identify sensors reporting abnormal values with the minimum energy expenditure [30]. For a more thorough review of the recent literature on VoI, we refer the reader to [31]. Semantic communication is a subject that is closely related to VoI, and has seen significant developments in the past few years: instead of a pre-defined value, information is evaluated and encoded based on *meaning*, which can only be derived implicitly, either from performance at a given task [32], [33] or by learning complex patterns in high-dimensional signals such as speech [34] or video [35].

An assumption that is shared by most of the works we discussed above is that information is always relevant, and that the application that tracks the process is always active. We can relax this assumption by considering a *query process*, as we did in our previous work [6], in which we defined the Age of Information at Query (QAoI): this metric is a sub-sampling of the AoI, only considering the instants when a query arrives from the application to be relevant for the optimization. A similar approach was adopted in defining the Effective Age of Information (EAoI) [36], with a slightly different set of assumptions, and our theoretical results were extended in [37], [38], which showed the different outcome of the AoI and QAoI minimization problems under an update-or-wait model. Another work also models requests in the optimization function [39], but it only deals with memoryless request processes,

TABLE I
MAIN NOTATION USED IN THE SYSTEM MODEL

Symbol	Dimension	Meaning	Symbol	Dimension	Meaning
N	1	Number of sensors	M	1	State dimension
$\mathbf{x}(t)$	$M \times 1$	System state	$\mathbf{A}$	$M \times M$	State transition matrix
$\mathbf{v}(t)$	$M \times 1$	Process noise	$\mathbf{\Sigma}_v$	$M \times M$	Process noise covariance
$\mathbf{y}(t)$	$N \times 1$	Observation	$\mathbf{H}$	$N \times M$	Observation matrix
$\mathbf{w}(t)$	$N \times 1$	Observation noise	$\mathbf{\Sigma}_w$	$N \times N$	Observation noise covariance
$\hat{\mathbf{x}}(t)$	$M \times 1$	Estimated state	$\boldsymbol{\psi}(t)$	$M \times M$	Estimate error covariance
$\hat{\mathbf{x}}_{\text{pri}}(t)$	$M \times 1$	*A priori* estimated state	$\boldsymbol{\psi}_{\text{pri}}(t)$	$M \times M$	*A priori* estimated error covariance
ε_n	1	Packet error probability for sensor n	$\mathbf{h}(t)$	$1 \times M$	Observation effect
$\mathbb{1}_{a(t)}$	$1 \times N$	Sensor selection vector	$y_{a(t)}(t)$	1	Received observation
$\mathbf{k}(t)$	$1 \times M$	Kalman gain	σ_w^2	1	Received observation variance
$\lambda(t)$	1	Reception indicator variable	C	1	Number of clients
$\mathcal{Q}_c$	1	Markov states of client c	$\tilde{\mathcal{Q}}_c$	1	Query states of client c
$\mathbf{T}_c$	$\lvert\mathcal{Q}_c\rvert \times \lvert\mathcal{Q}_c\rvert$	Transition matrix for client c	$z(\mathbf{x}(t))$	1	Query function
$\hat{z}(\hat{\mathbf{x}}(t))$	1	Query response	MSE_z	1	MSE for query z
S	1	Number of Monte Carlo samples	$\theta_{c,n}(t)$	1	VoI for sensor n and client c
$\mathcal{A}$	1	Scheduler action space	$\mathcal{S}$	$M^2 + M + C$	Scheduler state space
$\mathcal{O}$	$M^2 + M + C$	Scheduler observation space	π	$\mathcal{O} \to \Phi(\mathcal{A})$	Scheduler policy
$R(\pi)$	1	Long-term reward function	γ	1	Discount factor

which (as we will describe in the introduction) lead to a solution that is equivalent to standard AoI minimization. The extended version of that paper [40] considers more complex scenarios with partial battery knowledge, but still uses the same memoryless request model. Finally, a recent work by Xu *et al.* [41] also considers a memoryless request process, but considers a mix between traditional AoI and query-aware metrics. By only tracking the AoI when a query arrives from the application, the communication system considers not only the freshness of the received information, but also when it is needed: if, for example, the application works over discrete time intervals, transmitting more data close to the next query can reduce the bandwidth and energy usage, while maintaining the same or better accuracy from the application's point of view. We note that query awareness, and query prediction in particular, has been considered in the database literature [42], [43], and is related to the problem of predicting tasks in edge computing scenarios [44], [45]. However, contrary to our work, these do not consider the significance of the fetched data.

The problem of value-oriented scheduling has also been approached in distributed control: if the agents have communication capabilities, the most valuable piece of information is the one that will allow them to improve their performance in the task. The Urgency of Information (UoI) metric [46] directly considers how much an update would affect a known linear controller. If we consider Multi-Agent Reinforcement Learning (MARL) agents, the problem is more complex [47], as the communication policy is implicitly learned by the agents while they converge to the optimal control policy, and this approach has only been successful in simple problems [48] or with only one supporting agent communicating to a primary one [49]. However, because of the centralized nature of our pull-based setting, we limit our focus to the single-agent case, and refer the reader to [50] for a review of the cooperative MARL literature.

This work combines and extends some of the ideas on VoI sensor scheduling and query awareness, as well as concepts from the semantic communications literature, by considering a system with multiple queries arriving at different times, each of which requires different information on the state of the tracked process, represented by a different VoI function. To the best of our knowledge, this is the first work to consider this complete system model, and a significant step forward towards full-fledged goal-oriented communications.

III. SYSTEM MODEL

We consider a system in which an edge node receives information from a set of N sensors, indexed by $n \in \{1, 2, \ldots, N\}$, and has to respond to *queries* from a set $\mathcal{C}$ of remote clients, with $|\mathcal{C}| = C$. Time is divided into slots, denoted as $t = 0, 1, 2, \ldots$, and in each slot, the edge node can send a request to one sensor, and respond to queries from any number of clients. In turn, the sensors observe a linear dynamic system, whose state is denoted as $\mathbf{x}(t) \in \mathbb{R}^{M \times 1}$. The dimensionality of the process state is M, which can be different from the number of sensors N in the general case. The system evolves according to a (potentially time-varying) transition matrix $\mathbf{A}$, with an overlaid error perturbation modeled as a multivariate Gaussian noise:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t-1) + \mathbf{v}(t), \tag{1}$$

where $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}_M, \mathbf{\Sigma}_v)$. The noise $\mathbf{v}(t)$ is zero-mean, and its covariance matrix is $\mathbf{\Sigma}_v \in \mathbb{R}^{M \times M}$. The sensors then measure a vector $\mathbf{y}(t) \in \mathbb{R}^{N \times 1}$, which represents a linear observation of the state of the system with an added Gaussian measurement noise. We define an observation matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$, which defines the linear function of the state that each function observes:

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{w}(t), \tag{2}$$

where $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}_N, \mathbf{\Sigma}_w)$. The observation noise $\mathbf{w}(t)$ is also zero-mean, with a covariance matrix $\mathbf{\Sigma}_w \in \mathbb{R}^{N \times N}$. The main symbols in the model are listed in Table I, along with their dimensionality and meaning.

A. Remote Kalman Tracking

As the edge node does not know the real state $\mathbf{x}(t)$ of the monitored process, it needs to estimate it. In this work, we use the well-known Kalman filter [51], which is the optimal solution for linear dynamic systems. We assume that the edge node knows the matrices $\mathbf{A}$, $\mathbf{H}$, $\mathbf{\Sigma}_v$, and $\mathbf{\Sigma}_w$, and define vector $\hat{\mathbf{x}}(t) \in \mathbb{R}^{M\times 1}$ as the best estimate of the state available to the edge node. The Kalman filter also outputs a covariance matrix $\boldsymbol{\psi}(t)$, which corresponds to the expected value $\mathbb{E}[(\mathbf{x}(t)-\hat{\mathbf{x}}(t))^T(\mathbf{x}(t)-\hat{\mathbf{x}}(t))]$. We can then provide a recursive formula for updating the *a priori* estimate:

$$\hat{\mathbf{x}}_{\text{pri}}(t) = \mathbf{A}\hat{\mathbf{x}}(t-1) \tag{3}$$

$$\boldsymbol{\psi}_{\text{pri}}(t) = \mathbf{A}\boldsymbol{\psi}(t-1)\mathbf{A}^T + \mathbf{\Sigma}_v, \tag{4}$$

where the $_{\text{pri}}$ subscript in the estimates indicates that they are *a priori*.

As we stated above, the edge node can request the current value $y_n(t)$ from one sensor per timeslot, whose index is denoted by $a(t)$. We also consider communication errors, modeling the channel between sensor n and the edge node as a Packet Erasure Channel (PEC) with error probability ε_n. Considering the row vector $\mathbf{h}(t) \in \mathbb{R}^{1\times M}$:

$$\mathbf{h}(t) = \mathbb{1}_{a(t)}\mathbf{H}, \tag{5}$$

where $\mathbb{1}_{a(t)}$ is the row vector of length N whose elements are all 0, except for the one with index $a(t)$, which is equal to 1. We can then update the observation function in (2), getting the value $y_{a(t)}(t)$, which is the observation transmitted by the polled sensor $a(t)$:

$$y_{a(t)}(t) = \mathbf{h}(t)\mathbf{x}(t) + \mathbb{1}_{a(t)}\mathbf{w}(t) = \mathbb{1}_{a(t)}\mathbf{y}(t). \tag{6}$$

We indicate the outcome of the transmission at time t by the Bernoulli random variable $\lambda(t)$, which is equal to 1 if the transmission is successful and 0 otherwise. In the former case, the edge node receives observation $y_{a(t)}(t)$, while in the latter, it receives no observation for this time step. We can then give the Kalman gain row vector $\mathbf{k}(t) \in \mathbb{R}^{1\times M}$ as:

$$\mathbf{k}(t) = \boldsymbol{\psi}_{\text{pri}}(t)\mathbf{h}(t)^T\left(\mathbf{h}(t)\boldsymbol{\psi}_{\text{pri}}(t)\mathbf{h}(t)^T + \sigma_w^2(t)\right)^{-1}, \tag{7}$$

where $\sigma_w^2(t) = \mathbb{1}_{a(t)}\mathbf{\Sigma}_w\mathbb{1}_{a(t)}^T$. The update from the *a priori* estimate of the state to the *a posteriori* one is then given by:

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}_{\text{pri}}(t) + \lambda(t)\mathbf{k}(t)\left(y_{a(t)}(t) - \mathbf{h}(t)\hat{\mathbf{x}}_{\text{pri}}(t)\right) \tag{8}$$

$$\boldsymbol{\psi}(t) = \left(\mathbf{I}_M - \lambda(t)\mathbf{k}(t)\mathbf{h}(t)\right)\boldsymbol{\psi}_{\text{pri}}(t), \tag{9}$$

where $\mathbf{I}_M$ is the $M\times M$ identity matrix. We highlight that the *a priori* and *a posteriori* estimates are the same if $\lambda(t)=0$, e.g., no observation is received by the edge node [52], as the *a priori* estimate is the best estimate that the edge node can obtain with the information it has received.

B. The Query Process

We consider a *query* to be a request for either the state $\mathbf{x}(t)$ itself, or the value of a function $z(\mathbf{x}(t))$ of it. The edge node receives queries and responds with an estimate $\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$ based on the current state of the Kalman filter.

The temporal element of the query process can be modeled as a Markov chain. We assume that each client c follows an independent Markov chain, whose state at time t is $q_c(t) \in \mathcal{Q}_c$, with a known transition matrix $\mathbf{T}_c$. Each client c always requests the same function z_c anytime its Markov chain is in a subset of states, which we denote as $\tilde{\mathcal{Q}}_c$. Naturally, the state of each client is unknown to the edge node, which can only know which clients are currently subscribed and when did they send their last query. In some cases (e.g., periodic queries), this information is sufficient to predict the next query perfectly, as we will discuss below, but in the general case, the information about the query process available to the edge node entails some randomness and uncertainty.

C. Responding To Queries

The objective of the edge node is to respond to queries as accurately as possible, i.e., to minimize the error of its responses. The MSE for client c is defined as:

$$\text{MSE}_{z_c} = \mathbb{E}\left[\left(\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t)) - z(\mathbf{x}(t))\right)^2\right]. \tag{10}$$

The edge node can act in two ways to minimize the MSE: the first is to optimally use its knowledge of the state by using an Minimum Mean Square Error (MMSE) estimator to obtain $\hat{z}(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t))$, and the second is to poll sensors according to the expected VoI of their readings, i.e., schedule the sensor that can help the most in reducing the MSE of the next queries. The former problem is relatively simple, while the latter will be tackled in Sec. IV.

We can give the definitions and MMSE estimators for some of the most intuitive queries as follows:

1) *State*: in this case, the request is for the direct value of $\mathbf{x}(t)$;
2) *Sample mean*: in this case, the function that the client requests to the edge node is the sample average, represented by:

$$z_{\text{avg}}(\mathbf{x}(t)) = \frac{1}{M}\sum_{m=1}^{M} x^{(m)}(t), \tag{11}$$

where $x^{(m)}(t)$ is the m-th element of $\mathbf{x}(t)$;
3) *Sample variance*: in this case, the sample variance is computed as:

$$z_{\text{var}}(\mathbf{x}(t)) = \frac{1}{M-1}\sum_{m=1}^{M}\left(x^{(m)}(t) - z_{\text{avg}}(\mathbf{x}(t))\right)^2; \tag{12}$$

4) *Maximum*: this query returns the maximum component of the state:

$$z_{\text{max}}(\mathbf{x}(t)) = \max_{m\in\{1,\ldots,M\}} x^{(m)}(t); \tag{13}$$

5) *Count range*: this function counts how many components of the state are inside a given interval $[a,b]$. The count range query is defined as:

$$z_{\text{cnt}}(\mathbf{x}(t)) = \sum_{m=1}^{M}\mathbb{1}\left(x^{(m)}(t) \in [a,b]\right), \tag{14}$$

where $\mathbb{1}(\cdot)$ is the indicator function, equal to 1 if the condition inside the parentheses is verified and 0 otherwise. Note that the definition of the function $z_{\text{cnt}(\cdot)}$ should include the interval $[a,b]$, but here we omit it for the sake of readability. Furthermore, a and b are assumed to be fixed for the same query process.

The one-step optimal scheduler for the first three queries can be computed analytically by finding the choice that minimizes the MSE of the estimate at the next step, and we derive the one-step optimal schedulers for several queries in our previous work [8]. However, the optimal scheduler, and even the MMSE estimator, for more complex queries like the latter two, with highly non-linear functions, is hard to compute analytically. In the case of order statistics, a closed-form MMSE estimator might not even be achievable [53], as the extreme values of high-dimensional multivariate Gaussian variables are computed only as limiting distributions in the relevant literature. In order to compute the MSE of a given response to the count range query, we have to define the region $\mathcal{Z}(m)$:

$$\mathcal{Z}(m) = \left\{ \mathbf{x} \in \mathbb{R}^{M\times 1} : z_{\text{cnt}}(\mathbf{x}) = m \right\}. \tag{15}$$

We can then define the probability that $z_{\text{cnt}}(\mathbf{x}(t))$ is equal to m, corresponding to the integral of the multivariate Gaussian random variable $\mathbf{x}(t) \sim \mathcal{N}(\hat{\mathbf{x}}_t(t), \boldsymbol{\psi}(t))$ in $\mathcal{Z}(m)$:

$$\mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m) = \int\limits_{\mathcal{Z}(m)} \frac{e^{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}}_t(t))^T \boldsymbol{\psi}^{-1}(t)(\mathbf{x}-\hat{\mathbf{x}}_t(t))}}{\sqrt{2\pi^M |\boldsymbol{\psi}(t)}} d\mathbf{x}. \tag{16}$$

The MMSE estimator for the count range query is then given by:

$$\hat{z}_{\text{cnt}}(t) = \sum_{m=0}^{M} m \mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m). \tag{17}$$

The corresponding MSE is defined as follows:

$$\text{MSE}_{\text{cnt}}(t) = \sum_{m=0}^{M} (m - \hat{z}_{\text{cnt}}(t))^2 \mathcal{P}(z_{\text{cnt}}(\mathbf{x}(t)) = m). \tag{18}$$

The integral in (16) can only be computed numerically, and is extremely hard to tabulate.

We can then consider the VoI, which is used to evaluate the quality of a scheduler. If a query of type z_c arrives at step t, we define the value of the information available to sensor n as the expected reduction in the MSE for that query with respect to the *a priori* estimate. The VoI $\theta_{c,n}(t)$ is then given by:

$$\begin{aligned}\theta_{c,n}(t) = (1-\varepsilon_n)\mathbb{E}\left[(\hat{z}_c(\hat{\mathbf{x}}_{\text{pri}}(t), \boldsymbol{\psi}_{\text{pri}}(t)) - z_c(\mathbf{x}(t)))^2 \right] \\ -(1-\varepsilon_n)\mathbb{E}\left[(\hat{z}_c(\hat{\mathbf{x}}(t), \boldsymbol{\psi}(t)) - z_c(\mathbf{x}(t)))^2 \middle| a_t = n \right].\end{aligned} \tag{19}$$

The one-step optimal VoI scheduler for any query function can be easily approximated using Monte Carlo methods [54] by drawing samples from the *a priori* distribution. The detailed algorithm for Monte Carlo-based scheduling is given in our previous work [8]. While Monte Carlo estimates are not MMSE, they approach the optimal estimator as the number of samples grows to infinity, at the cost of computational complexity. If we consider S samples, the complexity of one Monte Carlo estimate is $O(SM^2)$.

IV. The Scheduling Problem

In the previous section, we defined the system model and determined the optimal estimator for common query functions, along with a Monte Carlo strategy for general functions. However, the most complex problem is not to reply directly to a query, but to consider future queries in a foresighted manner, scheduling sensor transmissions so as to minimize the MSE on future responses. This requires to consider not only the monitored system, but also the query process and the interplay between different query functions. For example, two clients which request the maximum and minimum will need very different parts of the state to be estimated accurately, and balancing between their needs will be complex. The polling decisions made by the edge node also affect the future state of the Kalman filter, requiring a dynamic strategy.

We can model the scheduling problem for the edge node as a POMDP, in which the edge node must decide which sensor to poll at each time slot. The action space is then simply $\mathcal{A} = \{1, \dots, N\}$, while the state space is more complex. The state of the Kalman filter just before the update, described by $\hat{\mathbf{x}}_{\text{pri}}(t)$ and $\boldsymbol{\psi}_{\text{pri}}(t)$, is included in the state, and so should all the states of the clients. The state space for a system with C clients is then $\mathcal{S} = \mathbb{R}^{M^2+M} \times \prod_{c=1}^{C} \mathcal{Q}_c$, as the state is given by the tuple $s(t) = (\hat{\mathbf{x}}_{\text{pri}}(t), \boldsymbol{\psi}_{\text{pri}}(t), q_1(t), \dots, q_C(t))$. However, the edge node does not know the state $q_c(t)$ of each client, but only the time that has passed since the last query, which we define as $\tau_c(t) \in \mathbb{N}$. We then have an observation tuple $o(t) = (\hat{\mathbf{x}}_{\text{pri}}(t), \boldsymbol{\psi}_{\text{pri}}(t), \tau_1(t), \dots, \tau_C(t))$, belonging to the observation space $\mathcal{O} = \mathbb{R}^{M^2+M} \times \mathbb{N}^C$. The matrices $\mathbf{A}$, $\mathbf{H}$, $\boldsymbol{\Sigma}_v$, and $\boldsymbol{\Sigma}_w$, as well as the error probability vector $\boldsymbol{\varepsilon} = [\varepsilon_n]$ and the query functions z_c, should also be known *a priori* to the edge node, but are not part of the state.

Note that the problem reduces to a fully observable MDP if the time since the last transmission is sufficient to determine the next query, i.e., if the following condition is true:

$$\begin{aligned}\mathcal{P}(q_c(t+1) \in \tilde{\mathcal{Q}}_c | q_c(t) = q) = \mathcal{P}(q_c(t+1) \in \tilde{\mathcal{Q}}_c | \tau_c(t)), \\ \forall q \in \mathcal{Q}_c, \tau_c(t) \in \mathbb{N}.\end{aligned} \tag{20}$$

Two special cases of this are the memoryless process, in which the Markov chain only has two states (query and no query), and the deterministic chain with $|\tilde{\mathcal{Q}}_c| = 1$, which leads to a periodic query process. In the general case, the state of the query process depends on external factors (e.g., a human operator), and is not directly knowable by the edge node: if a stochastic transition can lead to a state in which (20) is not verified, the problem is partially observable.

The transition probability $P(s, s'|a)$ from one state to the next for a given action is then determined by the Markov chains of each client, along with the Kalman filter equations in (8) and (9). The final parameter to define the POMDP is then the reward function $r(t)$:

$$r(t) = -\sum_{c\in\mathcal{C}} \alpha_c \text{MSE}_{z_c}(t) \mathbb{1}(q_c(t) \in \tilde{\mathcal{Q}}_c), \tag{21}$$

where $\alpha_c > 0$ is a weight parameter representing the relative importance of each client, whose value is given by the system designers, and is thus known *a priori* by the edge node. The reward is always negative, as the objective is to minimize the error on all queries.

We then define a *policy* $\pi : \mathcal{O} \rightarrow \Phi(\mathcal{A})$, where $\Phi(\mathcal{A})$ is a probability distribution over the action space $\mathcal{A}$. In other words, the policy maps observed states to the probability of selecting each sensor. We can then define the long-term reward function $R(t|\pi)$:

$$R(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(t) \Big| s_0, \pi\right], \tag{22}$$

where $\gamma \in [0, 1)$ is an exponential discount factor. The objective of the scheduling problem is then to find the optimal policy π^*, which maximizes the long-term reward:

$$\pi^* = \underset{\pi:\mathcal{O}\rightarrow\Phi(\mathcal{A})}{\arg\max}\ R(\pi). \tag{23}$$

The case for $\gamma = 0$ is a special case, in which future steps are never counted, and only performance in the next step matters: this case was solved analytically in our previous work [8].

A. A Simple Example: The Effect of Queries on the Optimal Policy

We can first consider a simple example, in which a system with $N = 2$ sensors observes a process with $M = 2$ and needs to reply to a single client (i.e., $C = 1$). The communication is assumed to be error-free, and each sensor m observes an independent binary Markov chain with state $\mathbf{x}(t) \in \{0, 1\}^M \forall t$ and $\mathbf{H} = \mathbf{I}_2$. At each time step, the state changes with probability p_m and remains the same with probability $1 - p_m$, so that the transition matrix $\mathbf{T}_m$ is given by:

$$\mathbf{T}_m = \begin{pmatrix} 1 - p_m & p_m \\ p_m & 1 - p_m \end{pmatrix}. \tag{24}$$

We know that the observation of the state is error-free, so after Δ_m steps from the last observation o_m, the *a posteriori* state probability distribution of Markov chain m is given by:

$$P_m(\Delta_m, o_m) = \mathbf{T}_m^{\Delta_m} \begin{pmatrix} 1 - o_m & o_m \end{pmatrix}^T. \tag{25}$$

We assume that a query arrives at every step from the client, but define two types of clients with different query functions: the first client asks for a maximum query, while the second asks for a count query, i.e., how many sensors measure a value of 1. If at least one of the sensors has a value of 1, the value of the other sensor is useless for the maximum query; on the other hand, it is still relevant for the count query. We can compute the MMSE response to each query:

$$\hat{z}_{\text{max}}(\boldsymbol{\Delta}, \mathbf{o}) = 1 - P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2); \tag{26}$$

$$\hat{z}_{\text{cnt}}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,1}(\Delta_1, o_1) + P_{2,1}(\Delta_2, o_2), \tag{27}$$

where $P_{m,i}(\Delta_m, o_m)$ is the *a posteriori* probability that chain m will be in state i, given the latest observation and its age, and $\boldsymbol{\Delta}$ and $\mathbf{o}$ are the vectors of ages and observed values, respectively. We highlight that, aside from a few special cases, the query response will be a value between 0 and 1, corresponding to the probability of the correct response being 1, as this is the MMSE estimator for Bernoulli random variables. We can also compute the MSE for both queries:

$$\text{MSE}_{\text{max}}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2) \tag{28}$$

$$\times (1 - P_{1,0}(\Delta_1, o_1) P_{2,0}(\Delta_2, o_2)); \tag{29}$$

$$\text{MSE}_{\text{cnt}}(\boldsymbol{\Delta}, \mathbf{o}) = P_{1,0}(\Delta_1, o_1) + P_{2,0}(\Delta_2, o_2) \tag{30}$$

$$- (P_{1,0}(\Delta_1, o_1))^2 - (P_{2,0}(\Delta_2, o_2))^2. \tag{31}$$

We can note that, if we observe one of the two states and $o = 1$, the response to the maximum query is always correct, as the probability of that component being equal to 0 is 0. In order to maximize the long-term reward from (22), we can adopt the classical policy iteration method, as described in [55, Ch. 4] after truncating the POMDP by setting a maximum age Δ_{max}. While policy iteration is not directly applicable to POMDPs, we can recast the problem as a fully observable MDP whose states fully describe the history of observations [56]. In our case, the time since the last observation of each state variable and the values of those observations are enough to enjoy this property. The expanded state is then defined as $s = (\boldsymbol{\Delta}, \mathbf{o}) \in \mathcal{S}$, over which we can apply policy iteration. The transitions from one state to the other are extremely simple, and we can easily derive the transition probability $\mathcal{P}(s(t + 1)|s(t), \pi(s(t)))$.

Policy iteration has two steps, called evaluation and improvement. The algorithm is initialized with an approximate value $V_0(s)$ for each state and a policy π_0, which can be set as all zeros. It then repeats the two steps iteratively until the policy converges. In the first step at iteration t, the value function $V_t(s)$ is updated as follows:

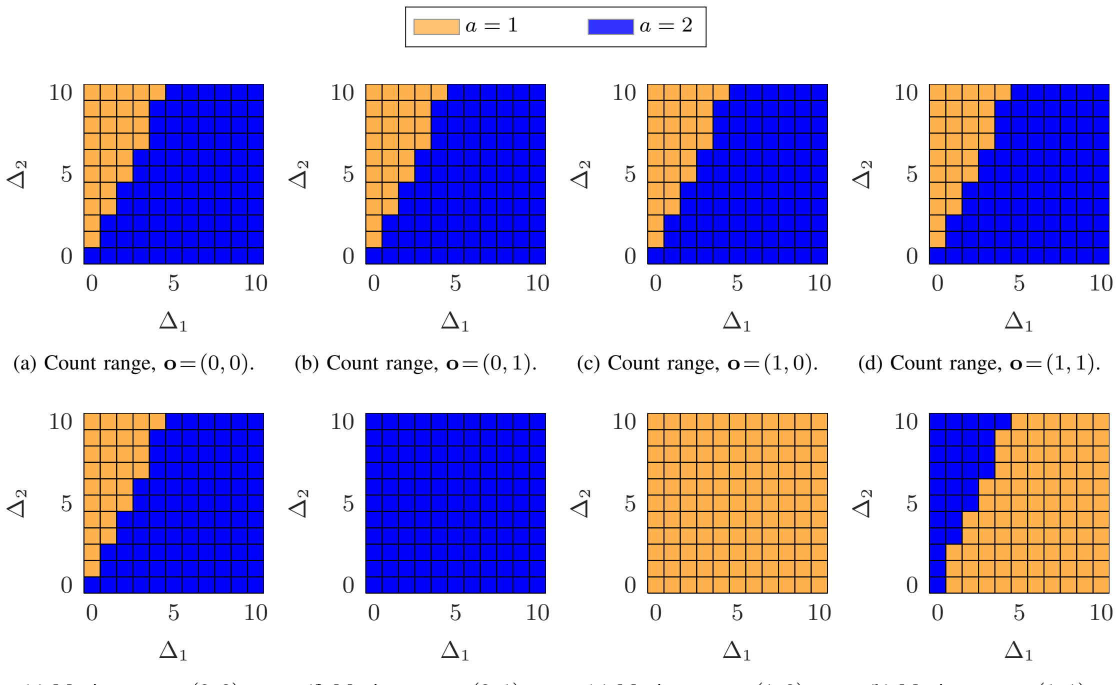
$$V_{t+1}(s) = -\text{MSE}(s, \pi_t(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi_t(s)) V_t(s'). \tag{32}$$

Naturally, the definition of the MSE depends on the type of query. After the value has been updated for all states, the policy is updated:

$$\pi_{t+1}(s) = \underset{a\in\{1,2\}}{\arg\min} -\text{MSE}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) V_{t+1}(s'). \tag{33}$$

Policy iteration is guaranteed to converge to the optimal policy in finite-state MDPs with finite reward [57]. We can also note that the formulation corresponds to maximizing the VoI $\theta_{c,m}(t)$ for the selected query, as defined in (19).

The results for $p_1 = 0.1$ and $p_2 = 0.2$ are given in Fig. 3. As Fig. 3a-d show, the policy is the same for any observation, and only depends on the age of the two measurements, since the MSE of the count query is the same for any observation. The level of uncertainty determines the action: the second component of the state, which can vary more often due to the higher state change probability, is the one that is polled, unless the age of the latest observation of the other component is approximately double. The maximum query has a more complex policy: if the last observations of each component are the same, i.e., $\mathbf{o} = (0, 0)$ or $\mathbf{o} = (1, 1)$, the policy is the same as for the count range query. On the other hand, if one of the last observations is 1, while the other is 0, the



(a) Count range, $\mathbf{o} = (0, 0)$. (b) Count range, $\mathbf{o} = (0, 1)$. (c) Count range, $\mathbf{o} = (1, 0)$. (d) Count range, $\mathbf{o} = (1, 1)$.

(e) Maximum, $\mathbf{o} = (0, 0)$. (f) Maximum, $\mathbf{o} = (0, 1)$. (g) Maximum, $\mathbf{o} = (1, 0)$. (h) Maximum, $\mathbf{o} = (1, 1)$.

Fig. 3. Optimal policies for the count range query (upper) and maximum query (lower) with $p_1 = 0.1$ and $p_2 = 0.2$.

component with value 1 is always polled. This is reasonable, as one observation from a sensor that contains a 1 gives perfect certainty on the overall maximum. Giving a higher priority to the component with the highest probability of being equal to 1 is then beneficial, even if the uncertainty on the other component becomes extremely high.

Naturally, this is only a simple example, and introducing a query process will complicate the system, but it highlights the strong dependence between the function that determines each query and the respective polling policy. While the optimal strategy to minimize the AoI would always poll the sensor with the highest age, and a strategy that minimizes the uncertainty of the count range query (which, in this simple system, is almost equivalent to minimizing the MSE) weighs each sensor's age by the speed of the corresponding process, the strategy for the maximum query actually depends on the current value of each sensor, and is starkly different from the others. Mixing different types of query in the same system will then lead to non-trivial trade-offs, particularly when the functions are highly non-linear.

B. Reinforcement Learning Solution and Learning Architecture

While policy iteration has strong convergence guarantees, it is infeasible to use when the state space is large, which is the case for the considered scheduling problem. Instead, we resort to approximate solutions, and consider a Reinforcement Learning (RL) approach to the scheduling problem. RL is a machine learning approach in which an agent learns from experience, updating its estimate of the value function by trial and error. The agent makes decisions and receives immediate rewards from the environment, without any prior knowledge of the reward function or the consequences of actions. For a more thorough introduction to reinforcement learning, we refer the reader to [55]. While RL is not directly applicable to partially observable scenarios, the observation space in the full problem is a sufficient statistic to respect the Markov property: as the Kalman filter is the optimal estimator for linear systems, and all the historic information of past observations is available in the state and covariance estimates, the system is equivalent to a fully observable MDP and we can directly apply standard solutions such as RL [56].

We implement the Deep Q Network (DQN) architecture [58], which uses a deep neural network to approximate the value function. In order to avoid instability, we need to use a *replay memory* to store the agent experience and select batches of uncorrelated samples. Each batch contains B uncorrelated samples, and each experience sample is a tuple $e = (s(t), a(t), r(t), s(t + 1))$. We maintain two neural networks for increased stability: a *target network* and an *update network*. In order to estimate the long-term reward $R(\pi)$ from an experience sample, we use the target network's prediction $Q_t(s, a)$:

$$Q(e) = r(t) + \gamma \max_{a \in \mathcal{A}} Q_t(s(t + 1), a). \tag{34}$$

The use of the long-term reward estimates to update future estimates follows the well-established bootstrap method, and the use of a greedy update policy follows the Q-learning model implemented by the DQN. The estimates $Q(e)$ are then used as labels for the backpropagation operation on the update network, whose output predictions are used in the action policy to select the next action. The action policy we use implements

TABLE II
DQN ARCHITECTURE.

Parameter	Layer 1	Layer 2	Layer 3
Input size	M^2+M+C	$2.5M$	M
Output size	$2.5M$	M	N
Activation function	ReLU	ReLU	ReLU
Dropout	0.1	0.1	0

the well-known softmax function:

$$\pi(s,a)=\frac{e^{\frac{Q_u(s,a)}{\tau}}}{\sum_{a'\in\mathcal{A}}e^{\frac{Q_u(s,a')}{\tau}}}, \tag{35}$$

where the temperature parameter τ is to balance between exploration and exploitation. Lower values of τ make the outcome closer to the greedy policy, as the probability of selecting suboptimal actions decreases, while higher values of τ increase exploration. In any case, exploration with the softmax function is *directed*: actions that are assumed to be highly suboptimal will be picked less frequently, while the agent will prefer actions that have an estimated long-term reward just below the maximum.

The update network is updated at every step with a new batch of samples, while the target network is only updated every U steps by copying the update network's weights. As we stated above, the use of separate target and update networks allows the system to converge, avoiding numerical and stability issues. In the rest of this work, we implement a DQN with 3 layers, whose parameters are given in Table II. The first two layers have a dropout probability $p_d = 0.1$ during the training, and the network is relatively simple, as the input is highly redundant. The hyperparameters above were found after a grid search optimization process.

C. Computational Complexity

We can now discuss the computational complexity of the learning solution. The following refers to the complexity of a trained model, i.e., of a single decision on the next action: while training can be performed offline in a simulation environment or even passively on existing data, actions need to be real-time for the system to work, and the time to make decisions is critical.

If we consider a single layer with ℓ_i inputs and ℓ_o outputs, there are three operations that the network needs to perform to compute each output:

1) Multiply each input ℓ_i by the appropriate weight (equivalent to ℓ_i multiplication operations);
2) Sum all the results (equivalent to ℓ_i sums);
3) Apply the non-linear activation function.

If we consider the activation function as the result of k basic operations, the total number of basic operations for a single layer is then $\ell_o(2\ell_i + k)$. If we consider our architecture as a vector $\boldsymbol{\ell}$ of layer sizes, where the first element is the cardinality of the input (i.e., the observed state) and the last element is N (corresponding to the N possible actions), the total complexity is:

$$\mathcal{C}_f(\boldsymbol{\ell})=\sum_{i=1}^{|\boldsymbol{\ell}|-1}\ell_{i+1}(2\ell_i+k). \tag{36}$$

TABLE III
DQN PARAMETERS.

Parameter	Description	Value
γ	Exponential discount factor	0.9
T_e	Time steps in each episode	100
E_{train}	Training episodes	100
E_{test}	Test episodes	10
p_d	Dropout probability	0.1
R_m	Replay memory size	10000
B	Batch size	128
t_{up}	Target net update period	10
L_o	Learning rate optimizer	Adam
L_0	Initial learning rate	10^{-4}

We remark that $\mathcal{C}_f(\boldsymbol{\ell})$ is the total number of basic operations per layer, and as such, If we consider our architecture for $C = 2$ and $N = 20$, which is given above, we have $k = 1$, as the Rectified Linear Unit (ReLU) activation function is extremely simple, and the total number of operations for each step is then $\mathcal{C}_f = 96\,570$. The backpropagation algorithm required to train the neural network has the same complexity as the forward pass, but it must be run for each sample in a training batch [59]. For a training batch size of B, the total complexity for a single training step is then given by:

$$\mathcal{C}_b(\boldsymbol{\ell})=B\mathcal{C}_f(\boldsymbol{\ell}). \tag{37}$$

In our architecture, we have $B = 128$, and consequently, $\mathcal{C}_b = 12\,360\,960$. This number of operations should be entirely within the capabilities of an edge node, as even simple embedded processors can deal with much more complex architectures that require billions of operations in less than 100 ms [60]. As most of the required calculations in training and evaluation are vector operations, each layer might only require a single clock tick on modern processors, particularly when the processor is a GPU or designed for hardware-assisted learning.

V. Simulation Settings and Results

The performance of the RL-based query-aware scheme is verified by Monte Carlo simulation, considering a specific scenario. Its performance is measured in terms of the MSE on its query responses. The evaluation is performed over $E_{\text{test}} = 10$ independent episodes, each of which consists of $T_{\text{max}} = 100$ time steps. The parameters of the DQN agent are the same for all considered scenarios, and are given in Table III.

A. Scenario and Benchmark Policies

We consider a system with $M = 20$ sensors, each observing a different component of the state $\mathbf{x}(t)$, so that $M = N$ and $\mathbf{H} = \mathbf{I}$. The dynamic system that the edge node observes is defined as follows:

$$\mathbf{A}^{(i,j)}=\begin{cases}\frac{3}{4}, & \text{if } i=j;\\ -\frac{1}{8}, & \text{if } i\neq j \wedge \operatorname{mod}(i-2j,7)=6.\end{cases} \tag{38}$$

The edge node knows $\mathbf{A}$, as well as the process and measurement noise covariance matrices, which are given by:

$$\Sigma_v^{(i,j)} = \begin{cases} \frac{11+\text{mod}(i-1,10)}{5}, & \text{if } i = j; \\ 1, & \text{if } i \neq j, \text{mod}(i-j,6) = 0, \end{cases} \tag{39}$$

$$\Sigma_w = \mathbf{I}. \tag{40}$$

The error probability ε_n for each sensor is given by:

$$\varepsilon_n = 0.02 \left\lceil \frac{n}{10} \right\rceil . \tag{41}$$

We consider a case with $C = 2$ clients with the same importance, i.e., $\alpha_1 = \alpha_2 = 1$. Client 1 requests a *count range* query with interval $[-5, 0]$, while client 2 makes a *maximum* query. The two query types are described in more detail in Sec. III-C, and we also refer the reader to our previous work [8] for a deeper discussion on the derivation of MMSE estimators for specific queries. It is possible to add more clients with other queries and varying importance. This adds one parameter (i.e., the time since the last query from that client) to the DQN, but the problem is not guaranteed to scale: the added complexity necessarily makes the training longer, requiring an adjustment to the exploration and learning profiles as well. In this work, we also include the results for a scenario with 4 clients, including an *average* and a *state* query as well.

We consider 5 different benchmarks for the query-aware policy:

- *MAF*: The MAF policy, which minimizes the average AoI of the system regardless of the value of sensors' readings. This legacy approach represents a value-neutral lower bound, as it aims at minimizing the AoI for all sensors regardless of the relevance of their data or their expected effect on the accuracy of the state estimate;
- *Cnt*: The one-step optimal policy for client 1, which follows the procedure from [8] to minimize the MSE of the count range query in the current step;
- *Max*: The one-step optimal policy for client 2, which does the same for the maximum query;
- *RL (Cnt)*: The foresighted policy learned by a RL agent with $\alpha_2 = 0$, which only minimizes the MSE of the response to the count range query;
- *RL (Max)*: The foresighted policy with $\alpha_1 = 0$, which does the same for the maximum query.

We also consider two different query processes, both of which are observable by the edge node, slightly simplifying the problem:

- *Periodic queries*: queries are generated every $T_q = 6$ steps. In this case, the Markov chain is deterministic, going from state 0 (in which a query is generated) to state 1 with probability 1, then increasing until 5, after which the chain goes back to 0 with probability 1;
- *Memoryless queries*: in this case, the Markov chain only has 2 states, and the rows of the transition matrix are identical. The time between two subsequent queries is geometrically distributed, with an expected value $\mathbb{E}[T_q] = 6$ steps.

We consider three combinations of these query processes: the case in which both clients have periodic queries, the case in which they both have memoryless queries, and the mixed case in which client 1 has periodic queries, while client 2 follows a memoryless process.

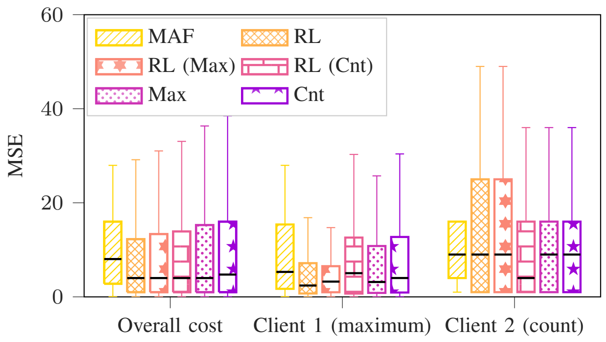


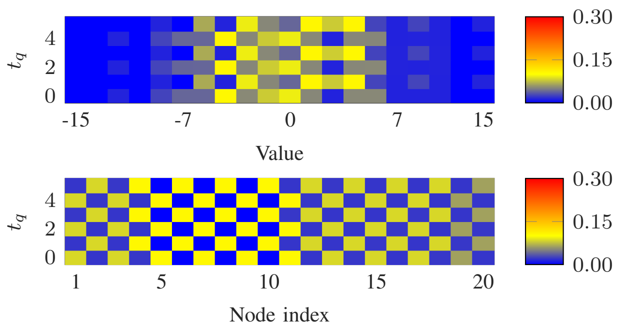
Fig. 4. MSE cost of the different policies for both types of query in the periodic query scenario.

B. Periodic Query Scenario

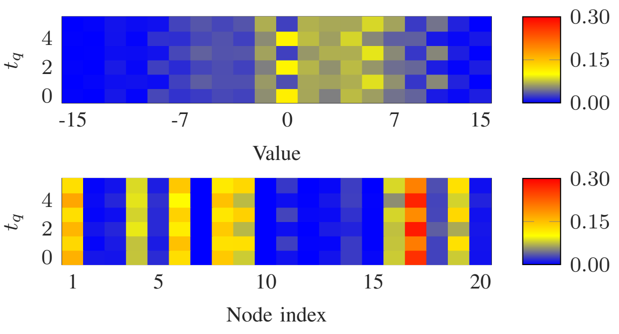
In this subsection, we show results for the case where both queries arrive periodically, each with a period of 6 steps. The two queries are out of sync, with the maximum query starting at 0 and the count range query starting at time 2.

This is the easiest case for the edge node, as queries are generated deterministically and the optimal policy can act on deterministic knowledge of the query pattern. Fig. 4 shows a boxplot of the MSE for both types of queries, as well as the overall cost, which is defined as the opposite of the average reward as given in (21). In other words, the overall cost is a weighted sum of the MSEs for all queries, using the weight vector $\boldsymbol{\alpha}$. In our simulations, queries all have the same period and weight, so the overall cost corresponds to the average of the MSEs over all queries. We can note that the RL policy considering both types of queries obtains a lower cost than all others (as shown in the group on the left of the figure), with a much lower average and only slightly worse performance at the 95th percentile than an AoI-oriented approach. In particular, the choice made by the RL policy is to privilege the maximum query, with results that end up being similar to only optimizing for it. All other approaches tend to reduce the MSE of the count range query more, although they end up having a higher error on the maximum query. The count range query is penalized by the fact that it arrives only 2 slots after the maximum query: reducing its MSE would require losing accuracy in the response to the maximum query, increasing the overall cost. On the other hand, the 4 slots between a count range query and the subsequent maximum query allow the RL policy to improve the accuracy significantly. The effect of the discount factor γ is also important: since a count range query arrives 2 slots after the maximum query before it, and $\gamma = 0.9$, its MSE only accounts for 81% of the reward for the steps before the maximum query. A higher value of γ, or a different weighting of the two query types by adapting α_1, would produce a more balanced outcome.

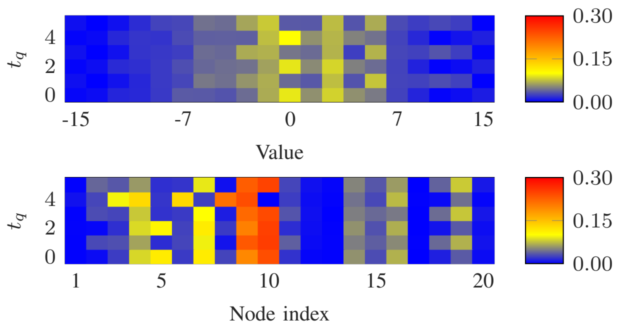
We can also note that, in this case, the other RL-based policies outperform their greedy versions on the metric that



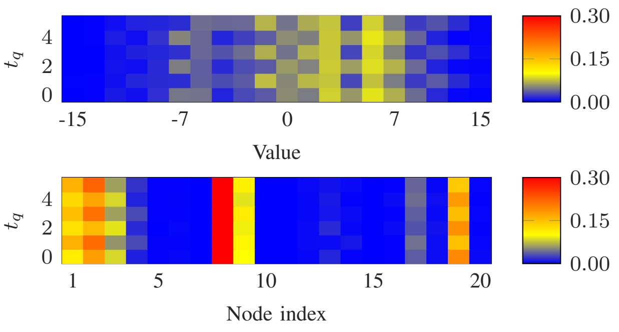
(a) MAF policy.



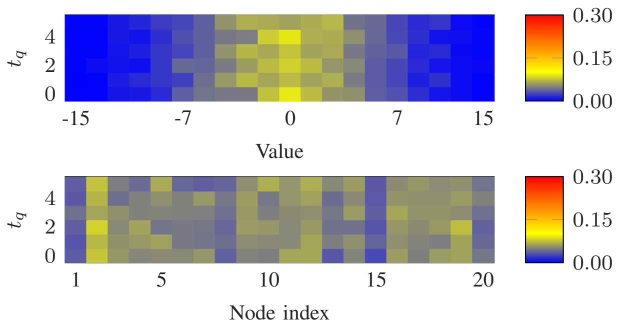
(b) RL policy.



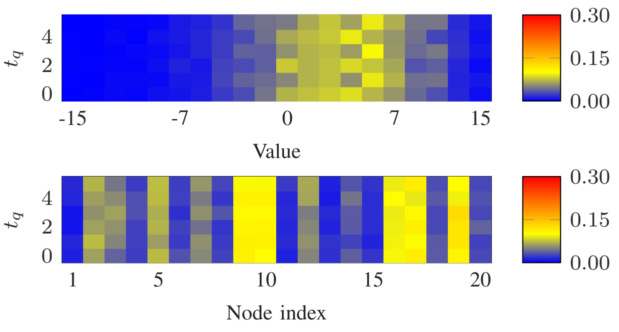
(c) Foresighted RL (Cnt) policy.



(d) Foresighted RL (Max) policy.



(e) One-step optimal Cnt policy.

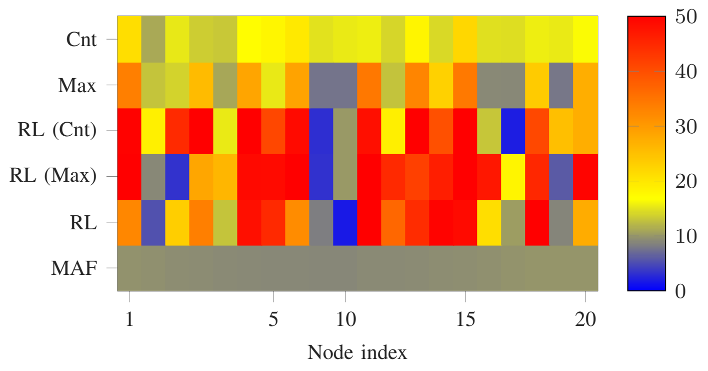


(f) One-step optimal Max policy.

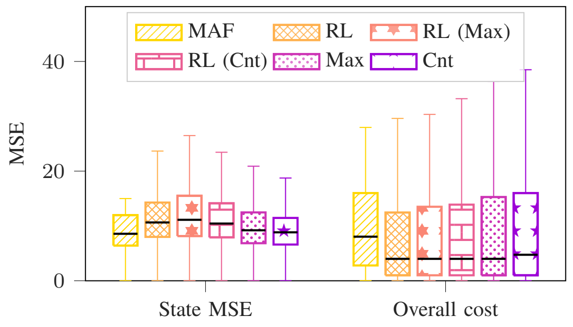
Fig. 5. Colormap representing the histogram of choices, in terms of sensor value and index, for each of the policies. The vertical axis represents the time t_q (count range queries arrive at time 0).

they optimize for, but no such pattern exists for the other type of query, which these policies completely disregard. As noted in previous works on VoI, the AoI-based approach taken by the MAF policy provides a middle ground for performance, never failing too badly by polling all sensors equally. The choices made by the various policies can be analyzed more in depth by considering the distribution of the sensors that are polled. Fig. 5 shows two colormaps for each policy, in which the y-axis represents the step in each query period, i.e., the index of the slot modulus 6. As a reminder, the maximum query is generated at $t_q = 0$ and the count range query is generated at $t_q = 2$. The two colormaps differ by the value represented on the x-axis: in the first one, the x-axis represents the value $x^{(m)}(t)$ measured by the chosen sensor, while in the second, the value is simply the index of the sensor. The color of each cell represents the empirical probability of each combination in our test episodes.

We can first look at the MAF policy, in Fig. 5a: the distribution of values is almost symmetrical, and values between -5 and 5 are polled with approximately the same frequency. On the other hand, the index colormap shows a checkerboard pattern, caused by the round robin-like pattern of updates (which is shifted by 2 steps at every cycle, as N is not a multiple of the query period). The RL policy has a different pattern: we can note that in even time slots, corresponding to the query instant, the distribution of values is bimodal: sensors whose value is close to 0 are polled very often, as are sensors whose value is very high, between 7 and 12. These two peaks correspond to the two queries: values close to 0 are at the edge of the interval that is relevant for the count range query, while very high values are obviously interesting for the maximum query. The indexes of the sensors that are polled are also much more concentrated: sensors 1, 6, 8, and 17 are polled extremely often, while other sensors are rarely polled: this is due to the



(a) Average AoI for each node.



(b) State and query MSE for all policies.

Fig. 6. Average AoI and error (in terms of state and query MSE) for different policies in the periodic query scenario.

nature of the problem, as some sensors are more valuable to answer the queries due to the evolution of the state.

The two single-query RL policies, whose choices are represented in Fig. 5c-d, can further shed light on the behavior of the joint policy: we can easily see that some of the nodes that are often polled by RL are also polled by its Max and Cnt versions, and that the two peaks in the distribution are close to a superposition of the two peaks of RL (Max) and RL (Cnt). Finally, we can note that the one-step greedy policies, shown in Fig. 5e-f, do not have any dependence on the time step, as they are unaware of the query process: the basic features, such as the Cnt policy choosing values clustered around 0 and the Max policy choosing values on the highest end of the range, are maintained, but the policies are inherently noisier than their RL-based versions, which can exploit their knowledge of the query process to improve performance at the right moment.

The two plots in Fig. 6 can further clarify the difference between simple AoI minimization, VoI minimization, and query-aware VoI. Fig. 6a shows the average AoI for each sensor for different policies. Naturally, the MAF policy maintains the minimum AoI, with an average only slightly over 10 for all sensors. As the policy tries to minimize the age for all sensors, the average is very similar across all sensors, although not identical (sensors with a higher packet error rate will be polled slightly more often as the poll is repeated after each packet loss). All other policies have a higher AoI for some sensors, polling them less often as they have less useful information, and the RL-based ones have the highest difference, with some sensors being polled extremely often and others almost never: as information useful for the queries can be reconstructed from the correlation between different sensors and the model of the dynamic system, the RL policies rarely poll sensors whose values are less useful or informative for the specific queries they are trained for. The plot in Fig. 6b, showing boxplots of the MSE on the state estimation for each policy clearly shows this: the RL-based policies actually have a *higher* MSE than simple MAF, as parts of the state are disregarded, but make a significantly smaller error when replying to queries, as the relevant information for the clients is given more importance in the scheduling.

Finally, we consider a more complex scenario, with 4 clients asking periodic queries with a period of 12 steps. Along with the maximum and count range queries, the two additional clients make state and sample mean queries, as defined in Sec. III-C. In this case, we also consider RL-based and one-step optimal strategies aimed at these queries, and the overall performance in terms of the reward and the specific queries (all of which have the same importance) is shown in Fig. 7. We note that the average query is easy to respond to, as errors in opposite directions over different components of the state tend to compensate: the RL solution outperforms all others in terms of the overall cost (i.e., the MSE over all queries made by clients), as while each query-specific strategy performs best on its own objective function, the RL policy manages to balance different queries, achieving a low error on all of them. We also highlight that legacy VoI optimization, even when foresighted (i.e., the RL (MSE) strategy in the figure), cannot effectively deal with maximum or count range queries, which depend only on specific parts of the state, as it does not take into account this importance, but statically aims at minimizing the error over all state components indiscriminately.

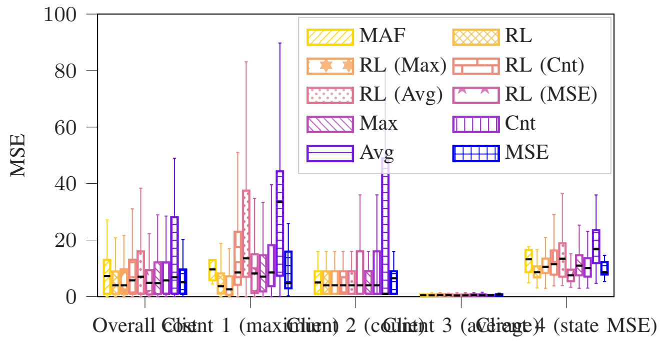


Fig. 7. MSE cost of the different policies for both types of query in the periodic query scenario with $C = 4$.

C. Geometric Query Arrival

We can consider a second scenario, in which at each step a query of either type is generated with probability $1/6$. The average frequency of queries is the same as for the previous scenario, but instead of a deterministic, periodic

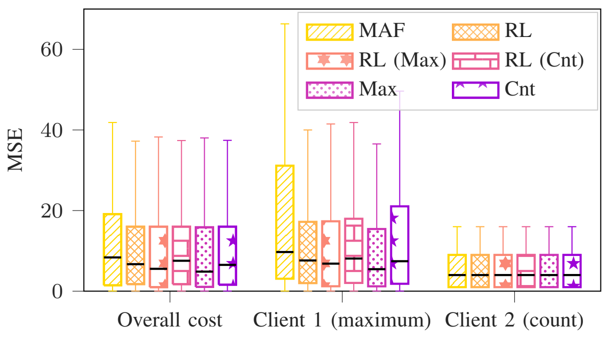


Fig. 8. MSE cost of the different policies for both types of query in the geometric query scenario.

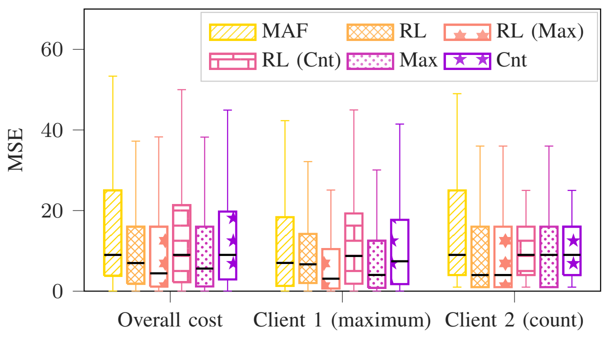


Fig. 9. MSE cost of the different policies for both types of query in the mixed query scenario.

sequence, queries follow a memoryless random process with geometrically distributed inter-query times. We remark that queries of both types may arrive to the edge node at the same time, and that in this case, no knowledge is available at the edge node: as the query process is memoryless, knowing the arrival times of past queries provides no information on future query arrivals. In this case, the advantage of a query-aware system is naturally diminished. The time since the last query of each type is still maintained as part of the input to the RL algorithm, so as to maintain the same architecture for all cases, but in this case, the RL algorithm needs to learn that this information is useless. This case also required more training than the other scenarios we considered.

Fig. 8 shows the performance boxplots for this scenario: in this case, performance is almost uniform, and all policies have a similar overall cost. The RL policy still has a small gain in terms of the overall cost, but it performs worse than the greedy Max policy on the maximum query. Performance on the count range query is almost uniformly good, and all differences between the policies are on the worst-case performance of the maximum query.

D. Mixed Query Arrival

Finally, we consider a third scenario: in this case, the maximum query follows a memoryless process with a probability $1/6$ of generating a query at each time step, while count range queries are periodically generated every 6 slots. In this case, the policy needs to adapt to the possibility of a maximum query arriving, while also preparing for the foreseen count range queries.

The performance of the considered policies is shown in the form of boxplots in Fig 9, as for the previous cases. The figure clearly shows that this case is much more complex, and the RL policy does not manage to outperform the strategies that are oriented exclusively toward the maximum query. Since the maximum query is entirely unpredictable, the full RL policy would need more training to deal with this scenario: the simpler strategy learned by the RL (Max) scheme turns out to be better on average, while RL (Cnt) performs about as well as RL. In most cases, the error on the count range queries tends to be higher for all policies. We note, however, that the RL strategy still outperforms all others in terms of worst-case performance, as the 95th percentile whisker is particularly low for the count query, resulting in better overall worst-case performance. A better strategy could be learned with more training, and we note that the complexity of the scenario has a significant impact on the amount of training required, with mixed scenarios with deterministic and stochastic query processes being the most difficult.

By knowing the instants in which count range queries will arrive, the RL strategies can limit the worst-case error, although this comes at the cost of a slightly higher worst-case error on the maximum query (which is hard to optimize for, as its arrival process is completely unpredictable). In this case, as in the geometric query arrival scenario, the one-step greedy policy for the maximum query is actually performing almost as well as the RL version, as there is no long-term information to be learned on the query process. As for QAoI, awareness of the query process is more useful if the latter is deterministic, or at least partially predictable.

VI. Conclusions and Future Work

In this work, we have presented a framework for query-aware sensor scheduling, in which an edge node needs to choose the most relevant information to respond to external user queries, which may be different functions of the system state. This type of scenario is closely linked to semantic and task-oriented communications in the IoT, approaching the problem from a different angle: in our system, communications are pull-based, and the bottleneck of the system is medium access rather than rate, so that the solution is semantic, VoI-based scheduling rather than encoding. Our work shows that query-aware scheduling can lead to profoundly different choices, depending on the specific functions that queries ask for and on the query arrival process for each client, and that RL-based strategies can provide a significant advantage in more predictable scenarios, while unpredictable query processes do not provide any useful information to improve scheduling past one-step greedy strategies.

There are several open avenues of research to extend this work, both on the scheduling itself and on the process estimation. Firstly, scheduling is currently limited to a single sensors, and communication is entirely pull-based: a scenario in which multiple sensors can be polled at once, or sensors can transmit urgent information without being polled first, can make scheduling strategies more interesting. Furthermore, extending the problem from simple numeric values to richer types of information such as images or point clouds could prove useful to several applications, such as cooperative driving or robot swarm management, which require the integration of data-heavy information from multiple sources. This also leads to the second line of future work that we are exploring, i.e., the substitution of the Kalman filter with more complex estimators, such as deep networks, which can deal with much more complex functions and system models, and do not require prior knowledge of the system dynamics. Finally, the combination of a control system with the remote estimation would represent another step forward toward a fully task-oriented communication system.

References

[1] P. Popovski, O. Simeone, F. Boccardi, D. Gündüz, and O. Sahin, "Semantic-effectiveness filtering and control for post-5g wireless connectivity," *J. Indian Inst. Sci.*, vol. 100, no. 2, pp. 435–443, Apr. 2020.

[2] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. University of Illinois Press, Sep. 1949.

[3] A. A. Soderlund and M. Kumar, "Optimization of multitarget tracking within a sensor network via information-guided clustering," *J. Guidance, Contr., Dynamics*, vol. 42, no. 2, pp. 317–334, Feb. 2019.

[4] R. D. Yates, Y. Sun, D. Richard Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of Information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, Mar. 2021.

[5] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør *et al.*, "A perspective on time toward wireless 6G," *Proc. IEEE*, vol. 110, no. 8, pp. 1116–1146, Jul. 2022.

[6] F. Chiariotti, J. Holm, A. E. Kalør, B. Soret *et al.*, "Query age of information: Freshness in pull-based communication," *IEEE Trans. Commun.*, vol. 70, no. 3, pp. 1606–1622, Jan. 2022.

[7] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled Software Defined Networking for efficient and scalable IoT communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, Sep. 2015.

[8] F. Chiariotti, A. E. Kalør, J. Holm, B. Soret, and P. Popovski, "Scheduling of sensor transmissions based on Value of Information for summary statistics," *IEEE Netw. Letters*, vol. 4, no. 2, pp. 92–96, May 2022.

[9] J. Du, C. Jiang, A. Benslimane, S. Guo, and Y. Ren, "SDN-based resource allocation in Edge and Cloud computing systems: An evolutionary Stackelberg differential game approach," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1613–1628, Feb. 2022.

[10] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 2731–2735.

[11] E. Uysal, O. Kaya, A. Ephremides, J. Gross *et al.*, "Semantic communications in networked systems: A data significance perspective," *IEEE Network*, vol. 36, no. 4, pp. 233–240, Oct. 2022.

[12] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *J. Commun. Netw.*, vol. 21, no. 3, pp. 204–219, Jul. 2019.

[13] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "The cost of delay in status updates and their value: Non-linear ageing," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4905–4918, Apr. 2020.

[14] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The age of incorrect information: A new performance metric for status updates," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2215–2228, Jul. 2020.

[15] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proc. ACM/IEEE 10th Int. Conf. Cyber-Phys. Sys. (CPS/IoT)*, Apr. 2019, pp. 109–117.

[16] N. Pappas and M. Kountouris, "Goal-oriented communication for real-time tracking in autonomous systems," in *Proc. IEEE Int. Conf. Auton. Syst. (ICAS)*, Aug. 2021.

[17] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 15–28, Dec. 2019.

[18] M. Moltafet, M. Leinonen, M. Codreanu, and N. Pappas, "Power minimization for Age of Information constrained dynamic control in wireless sensor networks," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 419–432, Nov. 2021.

[19] M. Emara, H. ElSawy, and G. Bauch, "A spatiotemporal model for peak AoI in uplink IoT networks: Time versus event-triggered traffic," *IEEE Internet of Things J.*, vol. 7, no. 8, pp. 6762–6777, 2020.

[20] M. Hatami, M. Leinonen, Z. Chen, N. Pappas, and M. Codreanu, "On-demand AoI minimization in resource-constrained cache-enabled IoT networks with energy harvesting sensors," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7446–7463, 2022.

[21] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251–260, Feb. 2006.

[22] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu, "Randomized greedy sensor selection: Leveraging weak submodularity," *IEEE Trans. Autom. Contr.*, vol. 66, no. 1, pp. 199–212, Mar. 2020.

[23] A. Nayak, A. E. Kalør, F. Chiariotti, and P. Popovski, "A decentralized policy for minimization of Age of Incorrect Information in Slotted ALOHA systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2023.

[24] S. Bharti, K. K. Pattanaik, and P. Bellavista, "Value of information based sensor ranking for efficient sensor service allocation in service oriented wireless sensor networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 2, pp. 823–838, Jan. 2019.

[25] A. Zancanaro, G. Cisotto, and L. Badia, "Modeling value of information in remote sensing from correlated sources," *Comp. Commun.*, vol. 203, pp. 289–297, Apr. 2023.

[26] V.-P. Bui, S. R. Pandey, F. Chiariotti, and P. Popovski, "Scheduling policy for Value-of-Information (VoI) in trajectory estimation for Digital Twins," *IEEE Commun. Lett.*, Apr. 2023.

[27] R. Duan, J. Du, J. Ren, C. Jiang, Y. Ren, and A. Benslimane, "VoI based information collection for AUV assisted underwater acoustic sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020.

[28] S. M. Hoseyni, F. Di Maio, and E. Zio, "VoI-based optimal sensors positioning and the sub-modularity issue," in *Proc. Int. Conf. System Rel. Saf. (ICSRS)*, Nov. 2019, pp. 148–152.

[29] G. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis, "Autonomous maintenance in IoT networks via AoI-driven deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Commun. Wkshp. (INFOCOM WKSHPS)*, May 2021.

[30] G. J. Stamatakis, N. Pappas, A. Fragkiadakis, and A. Traganitis, "Semantics-aware active fault detection in IoT," in *Proc. IEEE 20th Int. Symp. Model. Optim. Mobile, Ad hoc, Wireless Netw. (WiOpt)*, Sep. 2022.

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

[32] F. Pase, D. Gündüz, and M. Zorzi, "Rate-constrained remote contextual bandits," *IEEE J. Sel. Areas Inform. Theory*, 2022.

[33] D. Gündüz, Z. Qin, I. E. Aguerri, H. S. Dhillon *et al.*, "Beyond transmitting bits: Context, semantics, and task-oriented communications," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 5–41, Nov. 2022.

[34] Z. Weng, Z. Qin, X. Tao, C. Pan, G. Liu, and G. Y. Li, "Deep learning enabled semantic communications with speech recognition and synthesis," *IEEE Trans. Wireless Commun.*, Feb. 2023.

[35] L. Xia, Y. Sun, C. Liang, D. Feng *et al.*, "WiserVR: Semantic communication enabled wireless —Virtual Reality delivery," *IEEE Wireless Commun.*, vol. 30, no. 2, pp. 32–39, Apr. 2023.

[36] B. Yin, S. Zhang, Y. Cheng, L. X. Cai *et al.*, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 109–117.

[37] M. E. Ildiz, S. Avşar, and E. Uysal, "An inequality for Query Age of Information and Age of Information," in *30th IEEE Signal Process. Commun. Appl. Conf. (SIU)*. IEEE, May 2022.

[38] M. E. Ildiz, O. T. Yavascan, E. Uysal, and O. T. Kartal, "Query Age of Information: Optimizing AoI at the right time," in *Proc. Int. Symp. Inf. Theory (ISIT)*. IEEE, Jun. 2022, pp. 144–149.

[39] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, "Age-aware status update control for energy harvesting IoT sensors via reinforcement learning," in *Proc. IEEE 31st Ann. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Aug. 2020.

[40] M. Hatami, M. Leinonen, and M. Codreanu, "AoI minimization in status update control with energy harvesting sensors," *IEEE Trans. Commun.*, vol. 69, no. 12, pp. 8335–8351, Sep. 2021.

[41] C. Xu, X. Wang, H. H. Yang, H. Sun, and T. Q. Quek, "AoI and energy consumption oriented dynamic status updating in caching enabled IoT networks," in *Proc. Int. Conf. Comput. Commun. Wkshp. (INFOCOM WKSHPS)*, Jul. 2020, pp. 710–715.

[42] I. T. Bowman and K. Salem, “Optimization of query streams using semantic prefetching,” *ACM Trans. Database Syst.*, vol. 30, no. 4, pp. 1056–1101, Dec. 2005.

[43] K. Ramachandra and S. Sudarshan, “Holistic optimization by prefetching query results,” in *Proc. ACM SIGMOD Int. Conf. Mgmt. Data*, May 2012, pp. 133–144.

[44] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, “Cache-aided noma mobile edge computing: A reinforcement learning approach,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6899–6915, 2020.

[45] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, “Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading,” *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Apr. 2017.

[46] X. Zheng, S. Zhou, and Z. Niu, “Urgency of Information for context-aware timely status updates in remote control systems,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7237–7250, Jul. 2020.

[47] C. Zhang and V. Lesser, “Coordinating multi-agent reinforcement learning with limited communication,” in *Proc. ACM Int. Conf. Auton. Agents Multi-Agent Syst. (AAMAS)*, May 2013, pp. 1101–1108.

[48] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2016, pp. 2145–2153.

[49] 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 Commun.*, vol. 39, no. 8, pp. 2590–2603, Jun. 2021.

[50] A. Oroojlooy and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” *Applied Intell.*, pp. 1–46, Oct. 2022.

[51] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.

[52] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, “Kalman filtering with intermittent observations,” *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1453–1464, Sep. 2004.

[53] F. Amram, “Multivariate extreme value distributions for stationary Gaussian sequences,” *J. Multivariate Anal.*, vol. 16, no. 2, pp. 237–240, Apr. 1985.

[54] D. Luengo, L. Martino, M. Bugallo, V. Elvira, and S. Särkkä, “A survey of Monte Carlo methods for parameter estimation,” *EURASIP J. Advances Signal Process.*, vol. 2020, pp. 1–62, Dec. 2020.

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

[56] E. Hansen, “An improved policy iteration algorithm for partially observable MDPs,” vol. 10, Dec. 1997.

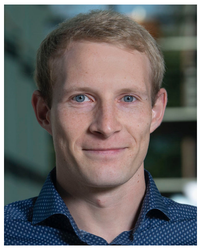
[57] R. A. Howard, *Dynamic programming and Markov processes*. John Wiley, Feb. 1960.

[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[59] B. Widrow and M. A. Lehr, “30 years of adaptive neural networks: perceptron, madaline, and backpropagation,” *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, Sep. 1990.

[60] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” *IEEE Access*, vol. 6, pp. 64 270–64 277, Oct. 2018.

Federico Chiariotti (S’15–M’19) is currently an assistant professor at the Department of Information Engineering, University of Padova, Italy, where he also received his Ph.D. in 2019. Between 2020 and 2022, he worked as a post-doctoral researcher and as an assistant professor at the Department of Electronic Systems, Aalborg University, Denmark. He has authored over 60 published papers on semantic communication, Age of Information, Smart Cities, and transport layer protocols. He was a recipient of the Best Paper Award at several conferences, including the IEEE INFOCOM 2020 WCNEE Workshop. His current research interests include network applications of machine learning, transport layer protocols, Smart Cities, bike sharing system optimization, and adaptive video streaming. He is currently an Associate Editor of the IEEE Networking Letters.

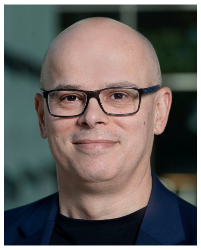


Anders E. Kalør (S’17–M’22) received the B.Sc. and M.Sc. degrees in computer engineering in 2015 and 2017, respectively, and the Ph.D. degree in wireless communications in 2022, all from Aalborg University. He is currently a postdoctoral researcher at The University of Hong Kong, supported by an individual International Postdoc grant from the Independent Research Fund Denmark. Concurrently, he is affiliated with the Connectivity section at Aalborg University, Denmark. In 2017, he was a visiting researcher at Bosch, Germany, and in 2020 at King’s College London, UK. He was awarded the Spar Nord Foundation Research Award for his Ph.D. project (2023). His current research interests include communication theory and the intersection between wireless communications, machine learning and data mining for IoT.

Beatriz Soret (M’11–SM’21) received her M.Sc. and Ph.D. degrees in Telecommunications from the University of Malaga, Spain, in 2002 and 2010, respectively. She is currently a Senior Research Fellow at the Telecommunications Research Institute, University of Malaga, and a part-time Associate Professor at Aalborg University. She has also held industrial positions in Nokia Bell Labs and GomSpace. She received a best paper award in IEEE Globecom 2013 and a Beatriz Galindo senior grant in Spain in 2020. Her current research interests include semantic communications and AoI, LEO satellite communications, and intelligent IoT environments.

Josefine Holm (S’19) received her B.Sc and M.Sc. degrees in mathematical engineering from Aalborg University in 2016 and 2018, respectively. She recently obtained her Ph.D. degree at the Connectivity Section at Aalborg University. Her research interests include wireless communication and IoT networks.

Torben Bach Pedersen is a professor with the Center for Data-Intensive Systems (Daisy), Aalborg University, Denmark. His research concerns Predictive, Prescriptive, and Extreme-Scale Data Analytics with Digital Energy as the main application area. He is an ACM Distinguished Scientist, an IEEE Computer Society Distinguished Contributor, a member of the Danish Academy of Technical Sciences, and holds an honorary doctorate from TU Dresden.



Petar Popovski (S'97–A'98–M'04–SM'10–F'16) is a Professor at Aalborg University, where he heads the section on Connectivity and a Visiting Excellence Chair at the University of Bremen. He received his Dipl.-Ing and M. Sc. degrees in communication engineering from the University of Sts. Cyril and Methodius in Skopje and the Ph.D. degree from Aalborg University in 2005. He is a Fellow of the IEEE. He received an ERC Consolidator Grant (2015), the Danish Elite Researcher award (2016), IEEE Fred W. Ellersick prize (2016), IEEE Stephen O. Rice prize (2018), Technical Achievement Award from the IEEE Technical Committee on Smart Grid Communications (2019), the Danish Telecommunication Prize (2020) and Villum Investigator Grant (2021). He is a Member at Large at the Board of Governors in IEEE Communication Society, Vice-Chair of the IEEE Communication Theory Technical Committee and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He is currently an Editor-in-Chief of IEEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. Prof. Popovski was the General Chair for IEEE SmartGridComm 2018 and IEEE Communication Theory Workshop 2019. His research interests are in the area of wireless communication and communication theory. He authored the book "Wireless Connectivity: An Intuitive and Fundamental Guide", published by Wiley in 2020.