2024 IFIP Networking Conference (IFIP Networking)

Collaborative Optimization of the Age of Information under Partial Observability

Anam Tahir*, Kai Cui†, Bastian Alt†, Amr Rizk*, Heinz Koeppl†
*University of Duisburg-Essen, Germany
†Technische Universität Darmstadt, Germany

***Abstract*—The significance of the freshness of sensor and control data at the receiver side, often referred to as Age of Information (AoI), is fundamentally constrained by contention for limited network resources. Evidently, network congestion is detrimental for AoI, where this congestion is partly self-induced by the sensor transmission process in addition to the contention from other transmitting sensors. In this work, we devise a decentralized AoI-minimizing transmission policy for a number of sensor agents sharing capacity-limited, non-FIFO duplex channels that introduce random delays in communication with a common receiver. By implementing the same policy, however with no explicit inter-agent communication, the agents minimize the expected AoI in this partially observable system. We cater to the partial observability due to random channel delays by designing a bootstrap particle filter that independently maintains a belief over the AoI of each agent. We also leverage mean-field control approximations and reinforcement learning to derive scalable and approximately optimal solutions for minimizing the expected AoI collaboratively.**

***Index Terms*—partial observability, reinforcement learning, mean-field control, network resources, age-of-information**

I. Introduction

Age of Information (AoI) is a measure that quantifies the freshness of information of a sender, e.g., a sensor, calculated using the time elapsed since the last update message was received at the receiver. It is an important metric in real-time applications such as UAV-assisted communications, Internet-of-Things, sensor networks, information processing systems, and cooperative, connected automated mobility [1], [2], [29].

We consider AoI-based systems where multiple sensors use the same channel, which has limited resources, to send their messages. This results in the need for congestion control and scheduling algorithms to regulate the network traffic while minimizing the AoI. Various scheduling algorithms have been presented for this purpose, see [16]–[18], [24] and references therein. We do not only consider channels of limited capacity, but also the standard assumption that it induces some random delays on the messages that are transmitted such that these may arrive out of order [10], [21]. It is this *non-FIFO channel behavior* in combination with agents lack of knowledge of the state of other agents or the system, that results in the partial observability of the state of the system since the sensors/agents do not possess *instantaneous* information about the updated AoI at the receiver when they decide on transmitting the next message. Here, we present a bootstrap particle filter [7] which uses the delayed, out-of-order messages to maintain a belief over the AoI of the agent.

Partial observability in AoI-based systems has been considered in recent works, where the system is modelled as a (partially observable)-Markov decision process (PO-MDP) and then solved using methods from (deep) reinforcement learning (RL). Authors in [6] propose a proactive deep reinforcement learning algorithm to optimize the performance of a vehicle-to-vehicle network for AoI-aware radio resource management, where the partial observability arises due to agents working in a decentralized manner. In [13], and similarly [5], [14], the authors use the partially observable Markov game framework to model a decentralized wireless communication network and apply deep Q-learning to find the scheduling and power control policy for minimizing the average AoI. Decentralized POMDP (Dec-POMDP) is a state-of-the-art framework for modeling such coordination problems [19] by explicitly considering the uncertainty and partial observability of the environment. However, this framework can be computationally expensive, especially as the number of agents and/or states increases [3], which hinders its application [25]. A recently popular scalable method for multi-agent systems is mean-field approximation, which has now also been used to learn policies for AoI-based systems [2], [27], [31]. We note that mean-field control, which is the collaborative mean-field formulation, has successfully been used to model multi-agent systems in the past [4], [11], but not for AoI-based optimization systems.

In this work, we develop a collaborative algorithm adapted from our recent decentralized partially observable mean-field control framework [8] to model the multi-agent AoI-based system as a single agent MDP. Our main contributions are: (i) We model a system where agents obtain delayed acknowledgments of their AoI leading to partial observability; (ii) We design a particle filter to cater to this uncertainty by maintaining a belief over the true AoI at the receiver; (iii) We adapt a partially observable mean-field control framework to learn scalable, collaborative policies for the decentralized system. Note that an extended version of this work is available at [26].

II. System Model

We consider a multi-agent system having N sensors (called agents), that each send status updates as ordered messages, m_n for the message index $m \in \mathbb{N}$ and $n \in \mathcal{N}$ with $\mathcal{N} = \{1, \dots, N\}$, to a single receiver, through a finite capacity channel C_1. The AoI process $x_n(t) \in \mathbb{R}_{\geq 0}$ associated with the messages of agent n, describes how old is the last received fresh update message at the receiver from that agent. Our

978-3-903176-63-8/24/$31.00 ©2024 IEEE

system model is illustrated in Fig 1. Next, we illustrate the dynamics of all components of this figure.

a) Receiver Dynamics: Upon receiving a fresh message from agent $n \in \mathcal{N}$ at time t, the receiver updates the AoI, $x_n(t)$, associated with the messages of agent n, independently of the other agents [29]. A fresh message is defined as an arriving message m_n with an index m larger than all the previously received message indices of agent n. The age of information process of agent n, is given as $x_n(t) \coloneqq \{x_n(t) \mid t \in \mathbb{R}_{\geq 0}\}$ as

$$x_n(t) = t - \max_{m \in \mathbb{N}} (\tau_{n,m} \mid \tau'_{n,m} \leq t) \tag{1}$$

where $t \in \mathbb{R}_{\geq 0}$ denotes the current time, $\tau_{n,m}$ is the sending time of the message with index $m \in \mathbb{N}$ of agent n into channel C_1 and $\tau'_{n,m}$ denotes the reception time of the message m_n at the receiver. The message m is transmitted through channel C_1, which incurs a random delay that we assume to be i.i.d sampled per message from an exponential distribution with rate λ_1. We hence call this one way delay $d^f_{n,m} \sim \text{Exp}(\lambda_1)$, where superscript f indicates delay in the *forward* direction from the sensor agents to the receiver. This leads to the following reception time at the receiver of the message m by agent n: $\tau'_{n,m} = \tau_{n,m} + d^f_{n,m}$.

We assume that the channel does not enqueue the packets and the agent is able to directly transmit the message upon creation. Note that the reception times are not necessarily ordered, i.e., it might well be the case that $\tau'_{n,m} \geq \tau'_{n,m'}$, for $m' > m$. The channel delay causes out-of-order delivery of messages to the receiver from each agent, making it a non-FIFO system [21]. The receiver, however, discards the outdated messages and only uses the fresh ones to update the AoI using Eq. (1).

In the following, we assume a measurement model for AoI, $x_n(t)$, of agent n at the receiver at time points $\eta_n \coloneqq \{\eta_{n,j}\}_{j \in \mathbb{N}}$ of the reception of only fresh messages, i.e.,

$$\eta_{n,j} = \left\{ \tau'_{n,j} \middle| j = \arg\max_{m \in \mathbb{N}} (\tau_{n,m} \mid \tau'_{n,m} \leq t) \wedge \forall t \in \mathbb{R}_{\geq 0} \right\}, \tag{2}$$

where $j \in \mathbb{N}$. We assume noise-free measurements $\{z_{n,j}\}_{j \in \mathbb{N}}$ of the process $x_n(t)$: $z_{n,j}(t) = x_n(\eta_{n,j})$[1]. Upon receiving a fresh message the receiver immediately transmits an acknowledgment of the received message to the sender, containing the reception time, $\eta_{n,j}$, and the correspondingly updated AoI $x_n(\eta_{n,j})$. These acknowledgments y_n are received by the agent n at time points $\eta'_{n,j}$, which are detailed in Sect. III. Fig. 2 shows one realization of the AoI process, $x_n(t)$, of agent n along with its key components.

b) Channel Dynamics: The considered system consists of two independent finite capacity channels, a forward channel C_1 and a backward channel C_2 as depicted in Fig. 1. In congruence with the delay model, we consider a forward channel that consists of a number of paths H where each path can serve exactly one message before idling again. We assume a fixed channel utilization (overbooking ratio), specifically, in terms of the ratio of the number of sensors to the number of paths, i.e., $H = \kappa N$, where $\kappa \leq 1$. At any time, when all the paths in the channel C_1 are occupied, any new incoming message is dropped and the sending agent is informed. In the considered discrete-time agent action model the total number of messages admitted into the channel at every time step is at most equal to the total number of free paths. We note that *the system runs in continuous time*, however, the agents observe the system and correspondingly take actions only at discrete time points. Channel C_1 chooses messages to admit uniformly at random out of all incoming messages, meaning an agent can have more than one message in C_1 at any time. As mentioned, each path in the channel delivers its message to the receiver after a delay, $d^f_{n,m} \sim \text{Exp}(\lambda_1)$. At any time t, the state of a path in the channel is $s_h(t) \in \mathcal{S} = \{0, 1\}$, for $h \in \{1, \ldots, H\}$, where $s_h(t) = 0$ indicates that the path is free and $s_h(t) = 1$ indicates an occupied path. The channel load, i.e., the ratio of occupied paths at time t, is: $\nu_1(t) = \frac{1}{H} \sum_{h=1}^{H} \mathbb{1}_{s_h(t)=1}$. The second channel C_2 is a designated channel used by the receiver to only send back acknowledgments, y, to the respective sensor agents. The capacity C of channel C_2 is set equal to that of channel C_1 since that the receiver only acks fresh messages which cannot be greater than H. Now the delay model for C_2 is chosen similarly as $d^r_{n,m} \sim \text{Exp}(\lambda_2)$, where the superscript r denotes the delay in *reverse* direction.

c) Agent Dynamics: An agent n generates and immediately sends a fresh message to the receiver according to some policy that defines the inter-transmission times. Upon receiving a message, if it is not outdated due to out-of-order arrivals, the

[1]We use both terms interchangeably due to the lack of noise

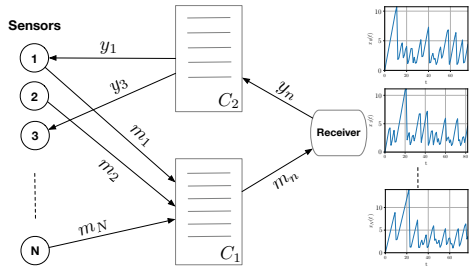

Fig. 1: Collaborative AoI system model having N sensors (agents), each sending its message m_N through channel C_1 and receiving acknowledgments y_N through channel C_2. The graphs on the right denote the AoI processes of the different sensors over time.

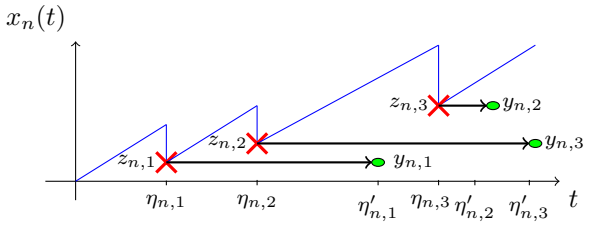

Fig. 2: AoI, $x_n(t)$, of agent n, showing out-of-order delivery of acknowledgements y_{n_i} back to agent n at times $\eta'_{n,i}$.

receiver updates the AoI for that agent at time points η_n. We assume that the only information an agent can access from the environment is the current channel load, $\nu_1(t)$.

Given the assumed delay distributions, the full state of agent n at time t is defined as $X_n(t) = (x_n(t), M_n^1(t), M_n^2(t))$, where $x_n(t)$ is the current AoI for the agent n at the receiver, $M_n^1(t)$ is the number of messages in channel C_1 from the agent n and $M_n^2(t)$ is the number of messages (acknowledgments) in channel C_2 from the receiver to agent n. The agent n receives observations, y_n, which essentially are the acknowledgments sent by the receiver. These acknowledgments, y_n, contain (i) the reception time, $\eta_{n,j}$, of the fresh message for that agent, and (ii) the corresponding value of the updated AoI, $x_n(\eta_{n,j})$, at the receiver.

Now, we assume that the agents *do not* have this full information, $X_n(t)$ and that it only has the count of the number of its unacknowledged messages, $u_n(t) = M_n^1(t) + M_n^2(t)$, and it can keep a belief over its AoI $x_{n,t}$ using the received observations y_n and the parameters of the channel delay model. As we will show later, we assume that the agent does not observe the continuous time processes u, x but rather samples its observations at discrete times and also takes actions on this discrete timescale. The actions, which will be illustrated in detail in the system design section, are mainly whether to generate and send a fresh message in this discrete time step or not. Finally, we assume that the agent n can only obtain the state of the forward channel when it sends to it, irrespective of whether its message is dropped or not. This leads to the considered system being partially observable.

d) Reward function: The agents aim to learn a policy π that decides whether to generate and transmit fresh messages, in order to maximize a reward. Since we have limited capacity channels being shared by all agents, we assume that the agents are willing to implicitly cooperate in order to maximize the global reward, which is to minimize the expected AoI over all agents. Hence, we fix the following reward function, $\Theta(t) = -\frac{1}{N}\sum_{n\in\mathcal{N}} x_n(t)$, where $\mathcal{N} = \{1, \ldots, N\}$. Note that other AoI related cost functions can be used here, as surveyed in [29]. For instance, one can simply penalize message drops in the reward function. Since the agents are working in an implicit cooperative manner to maximize this reward, we assume that this global reward is distributed to all agents.

III. AoI Minimization under Partial Observability

Next, we first illustrate the effect of the random channel delay on the observation model of the agents. We then propose a bootstrap particles filter for the agents in order to cater for the delayed observations. Lastly, we introduce the partially observable mean-field framework that is used to obtain decentralized policies for the agents to minimize their AoI.

a) Agents' Partial Observability: Recall, that the measurements from the receiver $\{z_{n,j}\}_{j\in\mathbb{N}}$ are not immediately observed by the corresponding agent n. The receiver sends the acknowledgment over the channel C_2, where each acknowledgment is subjected to a random i.i.d delay, $d^r_{n,j} \sim \text{Exp}(\lambda_2)$.

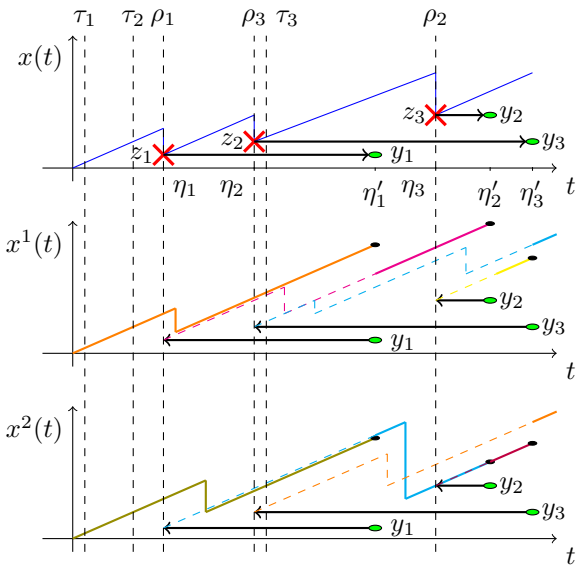

Fig. 3: Visualization of two particles, $(x^1(t), x^2(t)) \in \mathcal{X}$, for the true AoI process $x(t)$ of the agent from Fig. 2 (We drop the subscript denoting the agent for brevity).

The observations y_n, i.e., the delayed acknowledgment messages, are then received by agent n at time points, $\eta'_{n,j} = \eta_{n,j} + d^r_{n,j}$. Due to this channel delay, the acknowledgments received by the agents can also be out-of-order. And we make use of all these received observations (acknowledgments) to update our particle filter.

The ordered sending times from the receiver, $\rho_n = \{\rho_{n,j}\}_{j\in\mathbb{N}}$, for sent acknowledgments of agent n are then given as the order statistic of the sequence $\eta_n = \{\eta_{n,j}\}_{j\in\mathbb{N}}$: $\{\rho_{n,k}\} = \{\eta_{n,(k)}\} \quad \forall k \in \mathbb{N}$, where $(\cdot)_{n,(k)}$ denotes the kth order statistic, or the kth smallest value of the set η_n. Additionally, $l(k) : \mathbb{N} \to \mathbb{N}$ denotes the corresponding index of this value in the set η_n [9]. Also, the sequence of observations $y_n := \{y_{n,j}\}_{j\in\mathbb{N}}$ is given as: $\{y_{n,j}\}_{j\in\mathbb{N}} = \{z_{n,l(k)}\}_{k\in\mathbb{N}}$. Similarly, the ordered sequence of sending time points is then determined at the agent as: $\{\rho_{n,j}\}_{j\in\mathbb{N}} = \{\eta_{n,l(k)}\}_{k\in\mathbb{N}}$.

We use the information contained in the acknowledgments to feed the particle filter of each agent to independently update the belief over its own AoI, which is explained next. And we assume that the agent correctly knows the delay distributions. Finally, even though the agent only obtains the state of occupied paths in C_1 when it sends to it, the agent does not maintain a belief over the ratio of occupied paths, $\nu_1(t)$. Since calculating this belief is non-trivial and intractable, since each agent then needs to know the policy or the last timestep action of other agents.

b) Particle Filter: In order to make informed decisions while receiving delayed acknowledgments, the agents maintain a belief over their AoI, $x_n(t)$, using a particle filter [7], [22]. For ease of notation and without loss of generality, we have removed the subscript n from the following description, since each agent maintains its own belief independent of the others.

Let I denote the number of observations obtained by the

agent, i.e., the acknowledgments, until time point t, i.e., the (sender) agents' observation time points $\eta'_1 < \eta'_2 < \cdots < \eta'_I < t$ and the corresponding receiver side ordered sending times (of the acknowledgments) $\rho_1 < \rho_2 < \cdots < \rho_I < t$. We compute the posterior distribution of the latent path of the AoI process $x_{[0,t]}$ given the observations $y_{1:I} \coloneqq \{y_i\}_{i=1}^I$, their receiving times $\eta'_{1:I} \coloneqq \{\eta'_i\}_{i=1}^I$, and the sorted sending times $\rho_{1:I} \coloneqq \{\rho_i\}_{i=1}^I$. We denote the path of x between two time points $t < t'$ as $x_{[t,t']} \coloneqq \{x(s) \mid t \leq s \leq t'\}$. This posterior inference $p(x_{[0,t]} \mid y_{1:I}, \rho_{1:I}, \eta'_{1:I})$ is hard to compute, hence, we use a particle filter to represent this distribution in terms of a set of B particles, x^b, for $b = \{1, \ldots, B\}$. The likelihood, for our noise-free model, is given by a discrete distribution as $p(y_K \mid x(\rho_K)) = \delta(y_K - x(\rho_K))$, where K is the number of observations in the interval $[0, \eta'_I]$ and $x(\rho_K) = z_{(l(k))}$ is the ordered measurement.

We use the bootstrap particle filter in which the importance distribution is assumed to be the same as the prior distribution, so the weights only depend on the likelihood, see [7], [22] for more details. This is hard to achieve given the Dirac form of the likelihood when the filter updates the particles, especially if the state is continuous. Hence, the particles are reset to the last received observation, y_K, and their weights remain equally likely after every update.

The designed particle filter is explained using the example, depicted in Fig. 3. Here $x(t)$ is the true AoI not known to the agent and $x^1(t)$ and $x^2(t)$ are the two particles the agent uses to maintain a belief over $x(t)$. Given an observation y_k for the message $k \in \mathbb{N}$, containing the pair $\{z_k, \rho_k\}$, the agents' AoI at the receiver at time η_j is now known to the agent since $x\rho_k = z_k$. Once this observation is received, irrespective of the sequence order, all the particles of the agent are updated by going back in time to ρ_k and updating the value of all S particles to $z_{(l(k))}$.

In Fig. 3 we show two such particles $x^1(t)$ and $x^2(t)$. They both start at a time $t = 0$ using the initial belief, $x^1(0) = x^2(0) = 0$. The particles are *simulated* at a rate of 1 (as the AoI advances) until a time point where a sent message m_j may have been received by the receiver. The solid line is the simulation of the particle filter without observations and knowing the delay model distributions. The particles keep propagating in this manner until an observation y_j is received at the time η'_j. Upon receiving y_j, each particle in the set S is back-propagated in time to ρ_j and value $z_{(l(k))}$, (value of $x(\rho_j)$ at the receiver at time). The dashed lines are simulations based on a back propagated observation that was received until the current time t. Using the particle states, $x^b(t)$ where $b = \{1, \ldots, B\}$, the agent can compute its mean belief AoI, $\mu_B(t) = \frac{1}{B}\sum_{b=1}^B x^b(t)$, and the corresponding standard deviation, $\sigma_B(t) = \sqrt{\frac{1}{B}\sum_{b=1}^B (x^b(t) - \mu_B(t))^2}$, at any time t. Fig. 4 shows the performance of our particle filter for one agent over a time period of 50 seconds. It can be seen that the agents' belief of its AoI (red) is close to the true AoI at the receiver (black). This indicates that we can use the belief state to learn a good control policy for the agent actions.

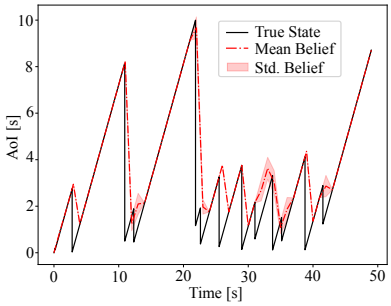

Fig. 4: Realization of our bootstrap particle filter estimate.

c) Partially Observable Mean-Field: Mean-field representations are a successful way of learning scalable solutions in multi-agent systems, both in cooperative [11] and competitive settings [28]. Our system model fits the former since we aim to reduce the expected AoI of the entire system. Note that learning a decentralized mean-field policy in partially observable systems still remains relatively uncharted, see [8].

We adapt the framework of [8] which presents a decentralized partially observable mean-field control (Dec-POMFC) model and assumes permutation-invariant agents to reduce an N-agent system to a single agent Markov decision Process (MDP). There, it is assumed that the problem dynamics, i.e. the distribution of AoI and messages in the channel, and rewards are Lipschitz in the mean field, which can be verified for our setting. A scalable policy for this MDP is then learned using the state-of-the-art single-agent RL policy gradient method known as Proximal Policy Optimization (PPO). In essence, the approach assumes the usage of an "upper-level" policy, $\tilde{\pi}$, during training. This policy assigns for any current mean field (full system state, see Sec. IV) the action probabilities for all agents with a particular current belief of AoI.

As stated in the agent dynamics above, the full state of the agent n at time t has three components $X_n(t) = (x_n(t), M_n^1(t), M_n^2(t))$ and the observations are denoted y_n. For the Dec-POMFC representation, a joint distribution, $\mu(t) = \frac{1}{N}\sum_n \delta_{X_n(t)}$, over the three components of the state $X_n(t)$, is considered of all agents. The number of total messages in channel C_1 at time t, i.e., $J_{C_1}(t)$, is instantaneously observed by an agent only when it sends a message to channel C_1, but can be calculated from the full state of all agents at any time t, as $J_{C_1}(t) = \sum_{i=1}^N M_i^1(t)$. From this, the ratio of currently occupied paths is obtained as $\nu_1(t) = \frac{J_{C_1}(t)}{H}$, and the empirical distribution over channel filling is calculated as $\nu = [\nu_0, \nu_1]$, where $\nu_0 = 1 - \nu_1$.

In contrast to [8], to learn the upper-level policy, $\tilde{\pi}$, for the Dec-POMFC model we make use of the mean and standard deviation of all three components of the state, $X_n(t)$ for all $n \in \mathcal{N}$ agents and $\nu(t)$ as a lower-dimensional and efficient input representation of the mean-field. Note that the mean, $\bar{\mu}$, and standard deviation, $\bar{\sigma}$, are just a specific function of the true distribution of the joint state distribution, $\mu(t)$, and can be a good approximation to the otherwise computationally

intractable $\mu(t)$. The learned upper-level policy, $\tilde{\pi}$, can then be used to extract the lower-level policy for each agent, $u_n(t) \sim \tilde{\pi}(\mu(t))$, i.e. the probability to send update messages given any current AoI, which decides on the agent actions $a_n(t) \sim u_n(t, x_n(t))$ based on its own (belief) AoI. Note that although the upper-level policy $\tilde{\pi}$ accesses the full system information $\mu(t)$ during training, we can perform decentralized execution, i.e., without knowledge of $\mu(t)$, after training the policy [8]. And, if agents don't have up-to-date information on their AoI, the particle filter is used to maintain a belief over it [22].

IV. System Design

Next, we give our system design in terms of the action space and observation space, the values of the parameters used, as well as, a description of the solver. We first explain our training process for learning the policy and how it is then evaluated on the system when the N is large.

*a) **Action space**:* The agents learn a policy, π, that dictates whether to generate and send a new message at the current time step or not. So, the action space is, $a \in \mathcal{A} = \{0, 1\}$, where $a = 1$ denotes a new message is generated and sent. The policy tells the agent what action to take based on the state in which the agent is, here, the state is the known (or the belief) AoI of each agent. As the AoI $x_n(t) \in \mathbb{R}_{\geq 0}$ is a continuous variable, learning a policy for this space state will need a long training time or will result in a non-optimal policy. Since values of AoI close to each other will have a similar policy, we quantize the AoI $x_n(t)$ into finite q levels, $\{[0, 1), [1, 2), \ldots, [q-2, q-1), [q-1, \infty]\}$, which dictates then the size of our action space for each agent n given as $\mathbf{a}_n = \{a_n^1, a_n^2, \ldots, a_n^q\}$, where $a_n^i \in \{0, 1\}$ for $i = 1, \ldots, q$, is an action for each quantized state level. Note that any arbitrary, finite number of quantization levels can be used.

*b) **Observation space**:* We consider the following three models for learning the policy, each differing in terms of the information used by the agents for policy training.

- **POMFC model**: This is the lower-dimensional representation of our partially observable mean-field model (Dec-POMFC), discussed in Sect. III-0c. To learn the POMFC policy the input observation is a vector containing the mean and standard deviation over the full state of all agents and channel state information, $\mathbf{o} = (\bar{\mu}(x_n(t)), \bar{\mu}(M_n^1(t)), \bar{\mu}(M_n^2(t)), \bar{\sigma}(x_n(t)), \bar{\sigma}(M_n^1(t)), \bar{\sigma}(M_n^2(t)), \nu_1(t))_{n=1,\ldots,N}$. Once an upper-level policy $\tilde{\pi}$ is learned it is used to extract the lower-level policy $u_n(t)$ for each agent from which the agent can choose its actions based on its individual AoI, $a_n(t) \sim u_n(x_n(t))$.
- **NA model**: This is the N-Agent scenario where the AoI of each agent is assumed known and directly used to learn the policy, leading to the input observation: $\mathbf{o} = (x_n(t))_{n=1,\ldots,N}$. Note that the input observation of this model is the same as that used by the lower-level policy $u_n(t)$ in our proposed POMFC model.
- **NA-Dec model**: This is the N-Agent scenario that uses the channel state information and the total unacknowledged messages of each agent to learn the policy. The input observation is: $\mathbf{o} = (x_n(t), e_n(t), \nu_1(t))_{n=1,\ldots,N}$, where $e_n(t) = M_n^1(t) + M_n^2(t)$ is the total unacknowledged messages of each agent.

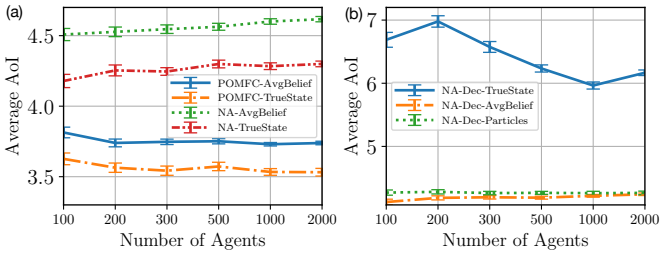

Fig. 5: Performance comparison of (a) both POMFC policies and (b) NA-Dec policies.

Based on the channel-induced delay we consider the following two variations of observations

- **TrueState** version: This is the scenario where we assume that the agent perfectly observes its true AoI, $x_n(t)$, without delay and also knows about the channel state, $\nu_1(t)$, at all times.
- **AvgBelief** version: This is the scenario where the channel-induced delay is taken into account and the agents only have access to delayed acknowledgments in the form of observations. Each agent then uses the above designed particle filter to maintain a belief over its true AoI which is used to calculate the average belief, $\bar{\mu}_{n,B}(t)$, and the standard deviation, $\bar{\sigma}_{n,B}(t)$, of each agent $n = 1, \ldots, N$. Additionally, for the channel state of C_1, we only use the last observation of each agent which they received upon the last attempt to use the channel.

For the NA-Dec model, in the AvgBelief version, we use as input observation all the particles directly, instead of their mean and standard deviation. The input observation for this **NA-Dec-Particles** version is then given as $\mathbf{o} = (x_{n,b}(t), e_n(t), \tilde{\nu}_{n,1}(t))_{n=1,\ldots,N, b=1,\ldots,B}$. We use this model to understand if using the lower-dimensional representation (mean and standard deviation) is enough to learn a well-performing policy.

*c) **Solver**:* In order to learn a policy π for each agent we use the state-of-the-art proximal policy optimization algorithm [23], which has shown promising results for both discrete and continuous state and action spaces. The reward function, Θ, used focuses solely on minimizing the average AoI of the system. To cater for partial observability of the channel and delayed observations of the acknowledgments, we use recurrent neural networks [30] with PPO. The experiments use the RLlib implementation of PPO [15].

*d) **Evaluation**:* Once a policy is learned it can then be used to extract the action $a_n(t)$ for an agent in a decentralized manner by using the (true or belief) AoI state, $x_n(t)$, of that agent. This only requires the quantized index $q_n(t)$ which reflects in which quantization level, $\mathbf{a}_n = \{a_n^1, a_n^2, \ldots, a_n^q\}$, the AoI lies for the agent n. The Python implementation of

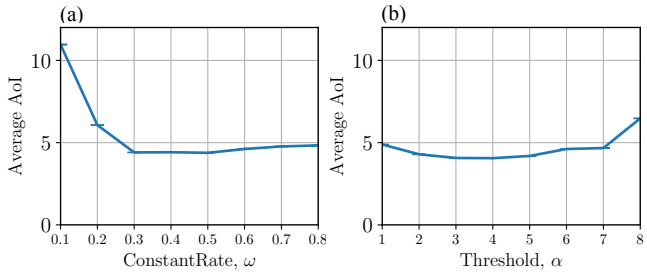

Fig. 6: AoI for a constant input rate ω into the channel C_1. (a) A non-trivial optimum for varying the channel load ω. (b) Effect of increasing the true AoI threshold, α.

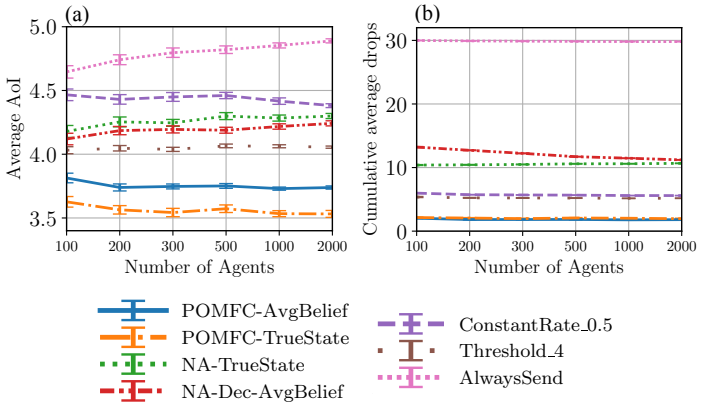

Fig. 7: (a) POMFC policies in comparison to the best performing (i) NA policies (NA-TrueState, Na-Dec-AvgBelief), (ii) ConstantRate policy, (iii) Threshold policy, and (iv) AlwaysSend. (b) The average drops of 100 Monte Carlo simulation are calculated at every time step and their cumulative total after 50 times steps is shown here.

our system and policies is available as open source [2].

V. Evaluation Results

Next, we present the results of our experiments. We use PPO to learn a policy that maximizes the given reward, Θ, and the training method falls in the category of centralized training and decentralized execution (CTDE) [12], with parameter sharing [20], in order to circumvent the non-stationarity issue common in the Multi-agent RL setup.

For all the presented results, the learned policy is evaluated on a range of agents: $N = \{10, 100, 200, 500, 1000, 2000\}$. For every N the evaluation comprises of 100 Monte Carlo simulations with corresponding confidence intervals. The POMFC policies were trained on $N = 100$ agents which can be considered enough to define a large system, although the POMFC framework is scalable, and a higher number of agents could have been used. The NA policies were trained using 10 agents only, due to the lack of scalability, as higher number of agents results in a much higher training convergence time.

[2]https://github.com/AnamTahir7/Collaborative-Optimization-of-the-Age-of-Information-under-Partial-Observability/tree/main

All the policies are compared based on the *average AoI* in the system, i.e., the average of the AoI of all N agents, which reflects the reward function, Θ.

In addition to the above introduced learned policies we have also compared to the following **fixed policies**: (i) **ConstantRate**: based on a fixed rate, ω, a subset of agents are randomly chosen to send their messages at every time step. (ii) **AlwaysSend**: every agent is sending a fresh message at every time step regardless of its AoI or channel state. This is bound to perform badly, especially in terms of message drops in the system. (iii) **Threshold**: every agent sends a fresh message if their true AoI exceeds certain threshold, α.

Fig. 5(a) shows that the learned POMFC-TrueState policy outperforms its finite system version NA policies as well as validating that it outperforms the POMFC policy using the average belief instead of the true state. Further, the figure shows that the POMFC-AvgBelief policy performs better than both NA policies, even given delayed observations. This result is an indication that the proposed particle filter works well, as also shown in Fig. 4. Hence, if the true AoI is *not instantaneously* available for learning the policy, then making use of the agents belief also obtains a well-performing policy.

Fig. 5(b) shows the performance comparison of the three versions of NA-Dec: TrueState, AvgBelief, and Particles. It can be seen that the policies using average belief and particles have a very similar performance and they both outperform the TrueState policy. Note that the TrueState policy in this scenario learns from the true values values of the AoI while in contrast the model that uses the particle filter uses additional information such as standard deviation over particles or the whole set of particles to learn a more explored and thus better policy. The particle filter is surely also able to maintain an accurate belief over the true state.

Fig. 6(a) shows the average AoI for the ConstantRate policy for a wide range of rates ω. The rate ω denotes a constant laod on the channel, i.e., a constant proportion of the agents (chosen uniformly at random at every time step) that send a message. Hence, $\omega = 1$ is equivalent to the AlwaysSend policy. It can be seen that there exists a non-trivial optimum over the rate ω. We have chosen the constant rate of $\omega = 0.5$ for further comparisons to other policies, i.e., at every time step half of the agents chosen uniformly at random will be sending new messages. Fig. 6(b) gives the performance comparison for using different AoI threshold values, α, where here too a non-trivial optimum exists. In further comparisons we use the value $\alpha = 4$. These two results show that such fixed policies cannot yield optimal performance and there is a need for learning a dynamic policy.

Finally, Fig. 7 shows the performance comparison of our proposed POMFC policies to the best performing (i) NA (NA-TrueState), (ii) NA-Dec (NA-Dec-AvgBelief) (iii) ConstantRate at $\omega = 0.5$ and (iv) Threshold at $\alpha = 0.4$, policies. It can be seen that our MDP-based POMFC-TrueState policy outperforms all other policies in terms of the average AoI of the system. Additionally, we show the comparison of the cumulative average number of message drops, where

the average is calculated at each time step. Here also our POMFC-TrueState policy has the least number of average drops, which intuitively goes hand in hand with minimizing the average AoI. Finally, our POMFC-AvgBelief policy, which uses our proposed particle filter, performs better than all the other learned and fixed policies (except when knowing the TrueState for POMFC), both in terms of average AoI and average message drops. This indicates that our particle filter performs well and is suitable for training where true AoI is not instantaneously available to the agents.

VI. Conclusion

We addressed the challenge of minimizing the expected Age of Information (AoI) in a multi-agent system using delay-inducing finite capacity forward and backward communication channels, which render the system partially observable. In particular, we considered agents that operate independently without global state awareness. Our approach leverages a bootstrap particle filter to update individual AoI beliefs per agent based on delayed acknowledgments. We employed a partially observable mean-field control (POMFC) framework to simplify the system into a single-agent MDP, which is solved using the PPO algorithm. The results show the superiority of the POMFC approach over fixed and other learned policies while underscoring its potential in complex multi-agent environments.

Acknowledgment

This work has been co-funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center, the ENVELOPE project under SNS JU, GA No 101139048 and the Collaborative Research Center (CRC) 1053–MAKI.

References

[1] Nehal Baganal-Krishna, Ralf Lübben, Eirini Liotou, Konstantinos V Katsaros, and Amr Rizk. A federated learning approach to qos forecasting in cellular vehicular communications: Approaches and empirical evidence. *Computer Networks*, page 110239, 2024.

[2] Andrea Baiocchi, Ion Turcanu, Nikita Lyamin, Katrin Sjöberg, and Alexey Vinel. Age of information in ieee 802.11 p. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1024–1031. IEEE, 2021.

[3] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[4] René Carmona, Mathieu Laurière, and Zongjun Tan. Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning. *arXiv preprint arXiv:1910.12802*, 2019.

[5] Elif Tuğçe Ceran, Deniz Gündüz, and András György. A reinforcement learning approach to age of information in multi-user networks with harq. *IEEE Journal on Selected Areas in Communications*, 39(5):1412–1426, 2021.

[6] Xianfu Chen, Celimuge Wu, Tao Chen, Honggang Zhang, Zhi Liu, Yan Zhang, and Mehdi Bennis. Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective. *IEEE Transactions on wireless communications*, 19(4):2268–2281, 2020.

[7] Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

[8] Kai Cui, Sascha Hauck, Christian Fabian, and Heinz Koeppl. Learning decentralized partially observable mean field control for artificial collective behavior. *arXiv preprint arXiv:2307.06175*, 2023.

[9] Herbert A David and Haikady N Nagaraja. *Order statistics*. John Wiley & Sons, 2004.

[10] Markus Fidler, Jaya Champati, Joerg Widmer, and Mahsa Noroozi. Statistical age-of-information bounds for parallel systems: When do independent channels make a difference? *IEEE Journal on Selected Areas in Information Theory*, 2023.

[11] Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. Mean-field controls with Q-learning for cooperative MARL: convergence and complexity analysis. *SIAM J. Math. Data Sci.*, 3(4):1168–1196, 2021.

[12] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[13] Shiyang Leng and Aylin Yener. Age of information minimization for wireless ad hoc networks: A deep reinforcement learning approach. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[14] Shiyang Leng and Aylin Yener. An actor-critic reinforcement learning approach to minimum age of information scheduling in energy harvesting networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8128–8132. IEEE, 2021.

[15] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[16] Praful D Mankar, Zheng Chen, Mohamed A Abd-Elmagid, Nikolaos Pappas, and Harpreet S Dhillon. Throughput and age of information in a cellular-based iot network. *IEEE Transactions on Wireless Communications*, 20(12):8248–8263, 2021.

[17] Ali Muhammad, Ibrahim Sorkhoh, Moataz Samir, Dariush Ebrahimi, and Chadi Assi. Minimizing age of information in multiaccess-edge-computing-assisted iot networks. *IEEE Internet of Things Journal*, 9(15):13052–13066, 2021.

[18] Mahsa Noroozi and Markus Fidler. Age-and deviation-of-information of hybrid time-and event-triggered systems: What matters more, determinism or resource conservation? *Performance Evaluation*, 164:102412, 2024.

[19] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[20] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[21] Amr Rizk and Jean-Yves Le Boudec. A palm calculus approach to the distribution of the age of information. *IEEE Transactions on Information Theory*, 2023.

[22] Simo Särkkä. *Bayesian filtering and smoothing*. Number 3. Cambridge university press, 2013.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] Yulin Shao, Qi Cao, Soung Chang Liew, and He Chen. Partially observable minimum-age scheduling: The greedy policy. *IEEE Transactions on Communications*, 70(1):404–418, 2021.

[25] Shinan Song, Zhiyi Fang, and Jingyan Jiang. Fast-drd: Fast decentralized reinforcement distillation for deadline-aware edge computing. *Information processing & management*, 59(2):102850, 2022.

[26] Anam Tahir, Kai Cui, Bastian Alt, Amr Rizk, and Heinz Koeppl. Collaborative optimization of the age of information under partial observability. *arXiv preprint arXiv:2312.12977*, 2023.

[27] Xuehe Wang and Lingjie Duan. Dynamic pricing and mean field analysis for controlling age of information. *IEEE/ACM Transactions on Networking*, 30(6):2588–2600, 2022.

[28] Batuhan Yardim, Semih Cayci, Matthieu Geist, and Niao He. Policy mirror ascent for efficient and independent learning in mean field games. In *International Conference on Machine Learning*, pages 39722–39754. PMLR, 2023.

[29] Roy D Yates, Yin Sun, D Richard Brown, Sanjit K Kaul, Eytan Modiano, and Sennur Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, 2021.

[30] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[31] Bo Zhou and Walid Saad. Age of information in ultra-dense iot systems: Performance and mean-field game analysis. *IEEE Transactions on Mobile Computing*, 2023.