AoI Analysis and Optimization in Systems with Computations-Intensive Updates

Saeid Sadeghi Vilni, Mohammad Moltafet, Markus Leinonen, and Marian Codreanu

***Abstract*—We consider a status update system consisting of a sampler, a controller, a processing unit, a transmitter, and a sink. The sampler generates a sample upon receiving a request from the controller and the sample requires further processing before transmission, hence is computation-intensive. This is mathematically modeled by a server called process server. After processing the sample, the status update packet is generated and sent to the transmitter for delivery to the sink. This is mathematically modeled by a server called transmit server. The service time of each packet at the transmit and process servers follow geometric distributions. Moreover, we consider that the servers serve packets under the blocking policy, i.e., whenever a server is busy at the arrival time of a new packet, the new arriving packet is blocked and discarded. We analyze the average age of information (AoI) for two fixed policies, namely, 1) zero-wait-one policy and 2) zero-wait-blocking policy. According to the former policy, the controller requests sampling when there is no packet in the system. According to the zero-wait-blocking policy, the controller requests a sample whenever the process server is idle. Furthermore, we develop an optimal control policy to minimize the average AoI using the tools of Markov decision process (MDP). In numerical results, we evaluate the performance of the policies under different system parameters. Moreover, we analyze the structure of the optimal policy.**

Index Terms: AoI, computation-intensive status update, Markov decision process, optimal status update control.

I. INTRODUCTION

EMERGING real-time applications, e.g., autonomous vehicles, wireless industrial automation, and health monitoring, in the upcoming generation of wireless communications rely heavily on the timely delivery of status updates [1], [2]. The age of information (AoI) [1], [2] is a metric used to evaluate the freshness of information in status update systems. AoI is the difference between the current and generation times of the last received status update packet at the destination [1], [2]. Each status update packet contains a timestamp representing the time when the sample was generated and the measured value of the monitored process. At time instant t, denoting the timestamp of the last received status update packet by $U(t)$, the AoI, $\Delta(t)$, is defined as $\Delta(t) = t - U(t)$ [1]–[4].

In some services with low latency requirements, e.g., autonomous driving, anomaly detection, and augmented reality (AR), the status updates need pre-processing before transmission. These systems are typically referred to as computation-intensive status update systems [5]–[7]. In these cases, in addition to the sampling and transmission, the processing time of each update also affects the information freshness [8]. Emerging techniques, such as edge, fog, or cloud computing, can be utilized to leverage the computing resources at the network [9]. However, enabling large-scale and distributed computing sources among heterogeneous devices requires the ability to pool their computing resources and computing resource management [10].

In this work, we consider a discrete-time status update system consisting of a controller, a sampler, a processing unit, a transmitter, and a sink (see Fig. 1). The sampler monitors a physical phenomenon and each sampled data needs to be processed before being transmitted. We mathematically model the processing and transmission processes of packets as servers, which are referred to as *process server* and *transmit server*, respectively. Upon receiving a request from the controller, the sampler generates a new sample which is then sent to the process server. The processed sample is turned into a status update packet and sent to the transmit server to be transmitted to the sink. After successfully receiving the packet, the sink sends an acknowledgment (ACK) to the transmit server, which is also overheard by the controller. The sampling time marks the beginning of the packets' age. We consider that the servers operate according to the blocking policy, i.e., if a new service requests arrive while the servers are busy (e.g., processing a previous sample at the process server or transmitting a previous status update at the transmit server), the newly arriving request is discarded. We assume that the service time of each packet at the transmit and process servers follow geometric distributions. We assume that the controller has global knowledge of the system, i.e., it can track the occupancy status of both processing and transmit servers.

The considered system model may represent a status update system providing health monitoring [11], [12]. In such a scenario, some data are generated by different sensors, in which the embedded information in data is not available until being processed. To this end, the data is uploaded to a process server, a pool of edge computing nodes, or a cloud computing node and then sent to a destination via a communication network for monitoring [13]. As the measured data

Manuscript received May 15, 2023; revised August 16, 2023; approved for publication by Yin Sun, Guest Editor, August 20, 2023.

This research has been financially supported by the Infotech Oulu, the Academy of Finland (grant 323698), and 6G Flagship program (grant 346208). The work of Markus Leinonen has also been financially supported in part by the Academy of Finland (grant 340171).

S. S. Vilni and M. Leinonen are with the CWC-RT, University of Oulu, 90014 Oulu, Finland. e-mail: {Saeid.SadeghiVilni, Markus.Leinonen}@oulu.fi.

M. Moltafet is with the Department of Electrical and Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064 USA. e-mail: mmoltafe@ucsc.edu.

M. Codreanu is with the Department of Science and Technology, Linkoping University, Sweden. e-mail: marian.codreanu@liu.se.

S. S. Vilni is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000040

are changed dynamically, the generated data needs different CPU cycles for processing [14]. In addition, each user uses shared computing and communication resources. Therefore, each packet encounters random delays for processing and transmission.

In the first part of the paper, we analyze the average AoI for two fixed policies, namely, 1) zero-wait-one policy, i.e., the controller takes a sample whenever the process and transmit servers are idle, and 2) zero-wait-blocking policy, i.e., the controller takes a sample whenever the process server is idle. We derive the closed-form expressions of the average AoI under these policies.

In the second part of the paper, we develop an optimal control policy to minimize the average AoI by finding the optimal sampling times. We cast the problem as a Markov decision process (MDP) problem and solve it numerically via the relative value iteration algorithm (RVIA) [15]. In numerical results, we evaluate the performance of the policies under different system parameters. Moreover, we analyze the structure of the optimal control policy.

The main contributions of the paper are summarized as follows:

- We consider a discrete-time computation-intensive status update system and study the AoI under a blocking policy.
- We derived closed-form expressions of the average AoI for two fixed policies: 1) zero-wait-one policy and 2) zero-wait-blocking policy.
- We develop an optimal control policy to minimize the average AoI using the relative value iteration algorithm.
- We investigate the structure of the optimal policy and show that zero wait policies are not necessarily age-optimal. Though counterintuitive, this shows that under certain circumstances (e.g., right after a status update was processed and delivered significantly faster than the average time) it is beneficial to insert idle slots instead of starting processing a new update whenever the system is empty. This phenomenon was first time pointed out in [16], and it was further investigated in [17], [18] under a different system setup.

A. Related Work

In this section, first, we review the works with a focus on AoI analysis or optimization in discrete-time status update systems. Then, the most related works with a focus on the computational-intensive status update system are presented. Finally, we present the main differences between our work and the most related works.

Prior works on the AoI in discrete-time status update systems can be categorized into two groups, namely, i) the works where the sampling process cannot be controlled and thus, the AoI is analyzed under fixed status update policies, e.g., [19]–[23], and ii) the works where the sampling process can be controlled and thus, the main goal is to determine the sampling time to optimize the system performance, e.g., [24]–[36]. These works use mathematical models such as MDP or Lyapunov technique to solve their optimization problem.

The authors of [19] studied the average AoI in a multi-source status update system with Bernoulli arrivals and a geometrically distributed service time. In [20] the authors derived closed-form expressions of average AoI in a random access-based status update system under different packet management policies. In [21], the authors derived a closed-form expression of the average AoI in a single-source rely-assisted status update system using the concept of Markovian jump linear systems. The authors of [22] analyzed non-linear functions of the AoI in a single-source status update system. In [23] the authors analyzed the stationary distribution of the AoI in a single-source status update system with Bernoulli arrivals and a generally distributed service time.

In [24], the authors proposed a power control policy to minimize the average AoI under an average power constraint. The authors of [25] proposed a near-optimal scheduling policy to minimize the average AoI in a two-way delay status update system. The work [26] studied a status update system under a two-way delay. Besides deriving the closed-form expressions of the average AoI for two fixed policies, they proposed an optimal control policy to minimize the average AoI. In [27], the authors proposed a transmission policy to minimize the AoI in a hybrid automatic repeat request (HARQ) based system with a non-orthogonal multiple access technique. The authors of [28] studied the freshness in a HARQ-based status update system in space–air–ground-integrated networks. They considered both HARQ Type I and Type III protocols and proposed a transmission policy for each protocol. The work [29] studied a multi-source status update system with an energy-harvesting-aided monitor which can request status updates from sources. The sources monitor the same physical process with different packet arrival rates. They proposed a scheduling policy to minimize AoI in the system. In [30], the authors studied the weighted sum AoI minimization problem in a multi-source status update system where the monitor decides which source sends the update. The AoI of the source is partially observable for the monitor. Using the partially observable MDP, they provided a scheduling policy to solve the problem. The authors of [31] studied a multi-user multi-sensor status update system, where the energy-harvesting sensors send an update by the users' demand. They proposed a control policy to minimize the average on-demand AoI. In [32], the authors considered a multi-user HARQ-based status update system with a generate-at-will source. They proposed several control policies to minimize the average AoI under an average number of transmissions constraint. The work [33] studied the average transmit power minimization with an average AoI constraint. They proposed a dynamic control policy to solve the problem, determining the power allocation, sampling times, and sub-channel assignment. The authors of [34] considered a system in which one energy-harvesting time-critical node and one data buffering node send packets over a shared channel to a destination. They proposed a control policy to minimize the average AoI of the time-critical node while keeping the average number of buffered packets less than that of the other node. In [35], the authors considered a status update system in which the forward and backward (for feedback) channels experience random delays. They proposed an optimal sampling policy that minimizes functions of the AoI subject to a constraint on the sampling rate. In [36], the authors studied a HARQ-based status update system with a random arrival source. They proposed several transmission scheduling policies for known and unknown environments.

The aforementioned works, i.e., [19]–[36], did not study the computational-intensive status updating. In the following, we review the most related works that study AoI minimization in computational-intensive status update systems.

The works [5]–[7], [37]–[39] studied the computation-intensive status update system in a continuous-time system for different queueing models with random arrivals. In [37], the authors studied a multi-source system. They calculated the moment-generating function of AoI under two different packet management policies using the stochastic hybrid systems technique. The AoI minimization in an IoT-based cellular network is studied in [38]. Each IoT device has a local processor and transmitter. They derived the closed-form expressions of the average AoI and an energy efficiency metric and then, minimized the expressions by finding the packet generation and compression rates. The work [39] considered a status update system where the IoT devices generate and offload data as a task to the edge or fog servers to extract the IoT devices' status updates. They modeled the task offloading as a two-stage tandem queue. They derived the closed-form expressions of the average AoI for different fixed policies. Moreover, using the closed-form expression of the average AoI, they proposed an optimal scheduling policy determining the packet generation rate, channel allocation, and computation resource allocation. In [5], the authors studied the status update system under three processing node models. They derived the closed-form expressions of the average AoI for a fixed policy under each processing node model. The author of [6] considered a two hops status update system. The buffer-assisted processing node receives the packets according to a Poisson process, and after processing, packets are sent to a sink via a buffer-assisted transmitter. They derived closed-form expressions of the average AoI and peak AoI (PAoI) for different fixed policies. The work in [7] studied a status update system where a source sends packets to an edge node. The communication and computing delays are modeled as two queues in tandem. They derived the distribution of the PAoI under different queueing models.

In contrast to the previous works, we consider a discrete-time computation-intensive status update system. Moreover, previous works on computation-intensive status update systems studied AoI when the sampling process can not be controlled, i.e., they considered systems with random arrivals and derived closed-form expressions of the average AoI or PAoI, whereas, in this paper, in addition to AoI analysis, we develop AoI optimal control policy for the system.

B. Organization

The rest of this paper is organized as follows. The system model is presented in Section II. In Section III, we analyze the AoI for the two fixed policies. The optimal control policy is presented in Section IV. Numerical results are presented in Section V. Finally, concluding remarks are made in Section VI.

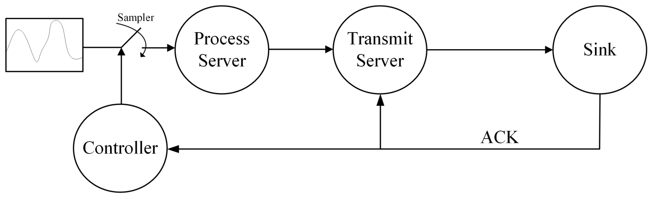


Fig. 1. The considered computation-intensive status update system model.

II. System Model

We consider a status update system consisting of a controller, a sampler, a (pre)processing unit, a transmitter, and a sink, as shown in Fig. 1. The processing and transmission components are modeled as servers and referred to as the process and transmit servers, respectively. The sampler monitors a random process and can generate a sample at the controller's request. The generated sample is sent to the process server, and the processing procedure starts. Upon finishing processing a sample, the process server generates a status update packet which is sent immediately to the transmit server. Whenever the sink receives successfully a packet, it feedbacks an acknowledgment (ACK) to the transmit server and it is also overheard by the controller. Note that, depending on the specific control policy, it is possible for a server to receive a new task while still serving the previous one. In such cases, we assume a blocking strategy, i.e., the new request is blocked and discarded whenever a server is busy. We assume the controller knows the utilization status of the process server, i.e., busy or idle. In addition, the controller knows the sink's reception status (i.e., receives a packet or not) and the utilization status of the transmit server via feedback. Therefore, the controller has a global knowledge of the system's status, i.e., the utilization status of the servers and the sink reception status. We consider a discrete-time system with unit time slots $t \in \{0, 1, 2, \cdots\}$.

At the beginning of each slot, the controller decides whether to generate a new sample or not. Let $a(t) \in \{0, 1\}$ denote the controller action at slot t, where $a(t) = 1$ indicates that the controller commands the sampler to generate a sample, otherwise sampler remains idle.

Let $g(t) \in \{0, 1\}$ denote the availability of the process server to start processing a new sample at the beginning of slot t, where $g(t) = 1$ indicates that the process server is available (i.e., it is idle), and $g(t) = 0$ otherwise. Note that, due to the discrete nature of time, if the server is idle at the beginning of slot t, (i.e., $g(t) = 1$), and a new sample is taken at slot t, the value of the server availability indicator is not changed at slot t, i.e., $g(t) = 1$ for the entire duration of slot t. We model the service times of the process server as independent and identically distributed geometric random variables with mean $1/\gamma$.

Let $y(t) \in \{0, 1\}$ denote the availability of the transmit server at the beginning of slot t, where $y(t) = 1$ indicates that the transmit server is available to send a new status update packet, and $y(t) = 0$ otherwise. We assume that the packet transmission times are independent and identically distributed and are modeled via a geometrically distributed service time with mean $1/p$.

III. Average AoI Analysis

In this section, we derive closed-form expressions of average AoI for two fixed policies, namely: 1) the zero-wait-one policy, where a new sample is taken just after the previous one was successfully delivered to the sink (i.e., when the system is empty) and 2) the zero-wait-blocking policy, where a new sample is taken whenever the process server becomes idle. Note that under the zero-wait-blocking policy, the transmit server may receive a new status update packet while serving the previous one; the new one is blocked and discarded whenever this happens.

Let $\Delta(t)$ denote the AoI at the sink at slot t, defined as the time elapsed since the sample associated with the most recently received packet was taken. Let $U(t)$ denote the *sampling* time of the most recently received packet (i.e., the time stamp), then, the AoI is given by $\Delta(t) = t - U(t)$. An example of AoI evolution over time is shown in Fig. 2. Let t_i denote the sampling time of the ith received packet at the sink. The processing of the packet is completed at time slot $t_i^{'}$, and the transmission of the packet is completed at time slot $t_i^{''}$. Let $\bar{\Delta}_\tau$ denote the average AoI until time slot τ, which is given as

$$\bar{\Delta}_\tau = \frac{1}{\tau}\sum_{t=1}^{\tau}\Delta(t). \tag{1}$$

Let $L(\tau) = \max\{i : t_i^{''} \leq \tau\}$ denote the number of successfully received updates until time slot τ. The average AoI is equal to the area under the AoI evolution curve depicted in Fig. 2. The area is a sum of disjoint areas determined by polygons $\underline{Q}$, $\{Q_i\}_{i=2}^{L(\tau)}$, and $\overline{Q}$. Thus, $\bar{\Delta}_\tau$ is calculated as

$$\bar{\Delta}_\tau = \frac{\underline{Q}+\overline{Q}}{\tau} + \frac{L(\tau)-1}{\tau}\frac{1}{L(\tau)-1}\sum_{i=2}^{i=L(\tau)} Q_i. \tag{2}$$

To analyze the AoI, we use a transition between the discrete AoI process and the AoI process. The AoI process is equivalent to the discrete AoI process, except that the age increases continually during a slot. To this end, let $\underline{Q}^{+}$, Q_i^{+}, and $\overline{Q}^{+}$ denote the trapezoids associated with $\underline{Q}$, Q_i, and $\overline{Q}$, respectively, as illustrated in Fig. 2. Using the trapezoid versions of $\underline{Q}$, $\{Q_i\}_{i=2}^{L(\tau)}$, and $\overline{Q}$ add a triangle of area $1/2$ at each slot to the area under $\Delta(t)$, thus, $\bar{\Delta}_\tau$ is derived as

$$\bar{\Delta}_\tau = \frac{\underline{Q}^{+}+\overline{Q}^{+}}{\tau} + \frac{L(\tau)-1}{\tau}\frac{1}{L(\tau)-1}\sum_{i=2}^{L(\tau)} Q_i^{+} - \frac{1}{2}. \tag{3}$$

Let $I_i = t_i - t_{i-1}$ denote the time interval between the sampling of packets $i-1$ and i, delivered to the sink. Let $T_i = t_i - t_i^{''}$ denote the total service time (i.e., process and transmit servers service times) of packet i. According to Fig. 2, Q_i^{+} is calculated as

$$Q_i^{+} = \underbrace{\frac{1}{2}(T_i+I_i)^2}_{\text{I}} - \underbrace{\frac{1}{2}(T_i)^2}_{\text{II}} = \frac{1}{2}I_i^2 + I_iT_i, \tag{4}$$

where part I in (4) represent the triangle with sides $I_i + T_i$, and part II represent the triangle with sides T_i. The long-term time average AoI is defined as $\bar{\Delta} = \lim_{\tau\to\infty}\bar{\Delta}_\tau$. Thus, the term $(\underline{Q}^{+} + \overline{Q}^{+})/\tau$ of (4) goes to zero when $\tau \to \infty$. We use the common assumption (see, e.g., [1], [4], [26]) that $\{(I_i, T_i)\}_{i\geq 1}$ is a stationary ergodic random process, i.e., $\lim_{\tau\to\infty}\frac{L(\tau)-1}{\tau}$ and $\lim_{\tau\to\infty}\frac{1}{L(\tau)-1}\sum_{i=2}^{i=L(\tau)} Q_i^{+}$ converge to their stochastic average as $1/\mathbb{E}\{I_i\}$ and $\mathbb{E}\{Q_i^{+}\}$, respectively. Therefore, the average AoI is given as

$$\bar{\Delta} = \frac{1}{\mathbb{E}\{I_i\}}\mathbb{E}\Big\{\frac{1}{2}I_i^2 + I_iT_i\Big\} - \frac{1}{2}. \tag{5}$$

A. AoI Analysis for Zero-wait-one Policy

Let $G_i \sim \text{Geo}(\gamma)$ denote the process server service time of the ith received packet and $Y_i \sim \text{Geo}(p)$ denote the transmit server service time of the ith received packet. AoI evolution under the zero-wait-one policy is illustrated in Fig. 3. Recall that under the zero-wait-one policy, the controller takes a sample whenever there is no packet in the system. Thus, processing of a new status update starts immediately after the previous one was successfully delivered (i.e., there are no slots where both process and transmit servers are idle simultaneously). Consequently, I_i and T_i are given as $I_i = G_{i-1} + Y_{i-1}$ and $T_i = G_i + Y_i$.

The mean of the inter-sampling time between two delivered packet i and $i-1$, $\mathbb{E}\{I_i\}$, is given as

$$\begin{aligned}\mathbb{E}\{I_i\} &= \mathbb{E}\{G_{i-1}\} + \mathbb{E}\{Y_{i-1}\}\\ &= \frac{1}{\gamma} + \frac{1}{p},\end{aligned} \tag{6}$$

where (6) follows from the fact that the mean service times of the process and transmit servers are $1/\gamma$ and $1/p$, respectively.

In order to derive the second moment of the inter-sampling time i, $\mathbb{E}\{I_i^2\}$, we first derive the second moment of the servers' service time, where $\mathbb{E}\{G_i^2\}$ is given as

$$\begin{aligned}\mathbb{E}\{G_i^2\} &= \sum_{j=1}^{\infty} j^2 \Pr(G_i = j)\\ &\overset{(a)}{=} \sum_{j=1}^{\infty} j^2\gamma\bar{\gamma}^{j-1}\\ &= \frac{\gamma}{\bar{\gamma}}\sum_{j=1}^{\infty} j^2\bar{\gamma}^{j}\\ &\overset{(b)}{=} \frac{\gamma}{\bar{\gamma}}\frac{\bar{\gamma}(1+\bar{\gamma})}{\gamma^3}\\ &= \frac{2-\gamma}{\gamma^2},\end{aligned} \tag{7}$$

where the equality (a) comes from $\Pr(G_i = j) = \gamma\bar{\gamma}^{j-1}$, and the equality (b) follows since $\sum_{j=1}^{\infty} j^2 l^j = l(l+1)/(1-l)^3$ for all $|l| < 1$. Following the same step, the second moment of the transmit server service time for packet i is given as

$$\mathbb{E}\{Y_i^2\} = \frac{2-p}{p^2}. \tag{8}$$

Using (7) and (8), the second moment of the inter-sampling time i, $\mathbb{E}\{I_i^2\}$, is given as

$$\mathbb{E}\{I_i^2\} = \mathbb{E}\{(G_{i-1} + Y_{i-1})^2\}$$

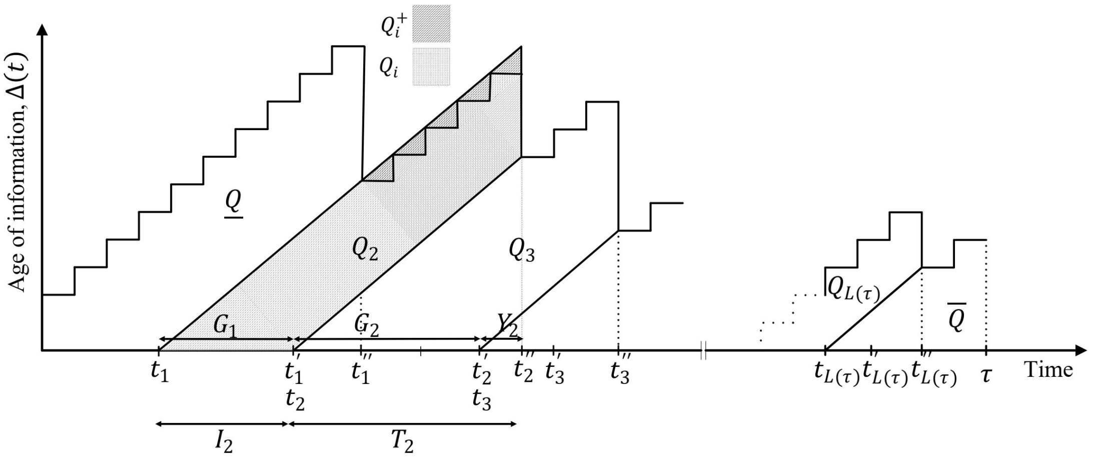


Fig. 2. An example of the evolution of the AoI.

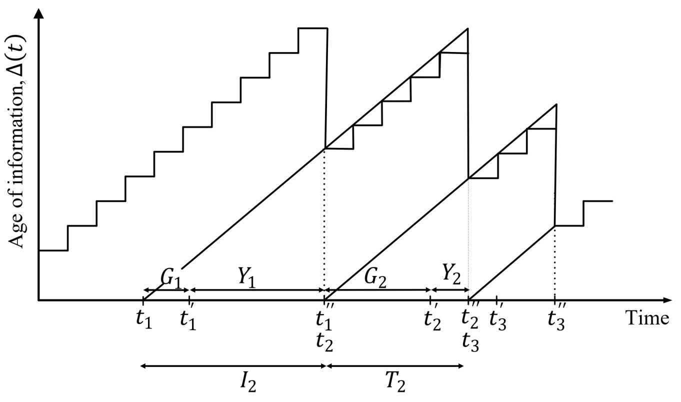


Fig. 3. AoI as a function of time under the zero-wait-one policy.

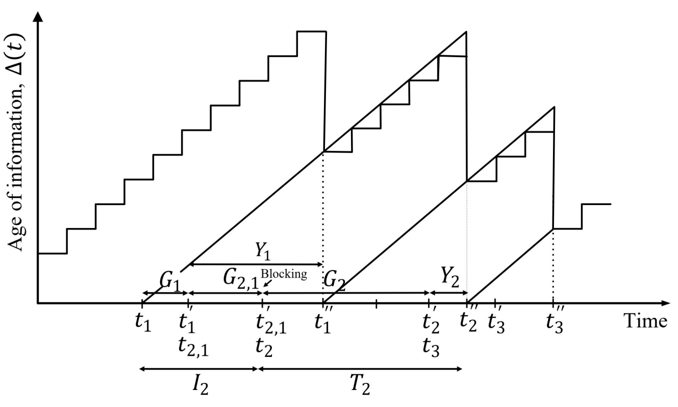


Fig. 4. An example of the evolution of the AoI under the zero-wait-blocking policy.

$$
\begin{aligned}
&\overset{(a)}{=} \mathbb{E}\{G_{i-1}^2\} + \mathbb{E}\{Y_{i-1}^2\} + 2\mathbb{E}\{G_{i-1}\}\mathbb{E}\{Y_{i-1}\} \\
&= \frac{2-\gamma}{\gamma^2} + \frac{2-p}{p^2} + \frac{2}{\gamma p},
\end{aligned} \tag{9}
$$

where the equality (a) comes from independence of G_{i-1} and Y_{i-1}. The value of $\mathbb{E}\{I_i T_i\}$ is given as

$$
\begin{aligned}
\mathbb{E}\{I_i T_i\} &= \mathbb{E}\{(G_{i-1}+Y_{i-1})(G_i+Y_i)\} \\
&= \mathbb{E}\{G_{i-1}G_i\} + \mathbb{E}\{Y_{i-1}Y_i\} \\
&\quad + \mathbb{E}\{G_{i-1}Y_i\} + \mathbb{E}\{Y_{i-1}G_i\} \\
&\overset{(a)}{=} \mathbb{E}\{G_{i-1}\}\mathbb{E}\{G_i\} + \mathbb{E}\{Y_{i-1}\}\mathbb{E}\{Y_i\} \\
&\quad + \mathbb{E}\{G_{i-1}\}\mathbb{E}\{Y_i\} + \mathbb{E}\{Y_{i-1}\}\mathbb{E}\{G_i\} \\
&= \frac{1}{\gamma^2} + \frac{1}{p^2} + \frac{2}{\gamma p},
\end{aligned} \tag{10}
$$

where the equality (a) follows from independence of G_{i-1}, G_i, Y_{i-1}, and Y_i.

Finally, substituting (6), (9), and (10) into (5), the average AoI is given as

$$
\bar{\Delta} = \frac{\frac{1}{2}\left(\frac{2-\gamma}{\gamma^2} + \frac{2-p}{p^2} + \frac{2}{\gamma p}\right) + \frac{1}{\gamma^2} + \frac{1}{p^2} + \frac{2}{\gamma p}}{\frac{1}{\gamma} + \frac{1}{p}} - \frac{1}{2}.
$$

B. AoI Analysis for Zero-wait-blocking Policy

Recall that under the zero-wait-blocking policy, the controller takes a sample whenever the process server is idle. If the transmit server receives a new packet while still serving the previous one, the newly arrived packet is blocked and dropped. Let $t_{i,j}$ denote the sampling time of the jth packet blocked at the transmit server by the $(i-1)$th transmitted packet. Let $t_{i,j}'$ denote the time instant that the blocked packet is generated by the process server. An example of the evolution of the AoI under the zero-wait-blocking policy is illustrated in Fig. 4.

Let $G_{i,j}$ denote the process server service time of the jth packet blocked by $(i-1)$th served packet. Let $\tilde{Y}_{i-1,k}$ denote the transmit server residual service time of the $(i-1)$th packet at the sampling time of the kth packet, it could be a blocked packet or a served packet. Let D_i represent the event where the packet i starts to be served at the transmit server, i.e., D_i happens if $\{G_i > \tilde{Y}_{i-1,i}\}$. Note that in the case that there is no blocking event between two served packets $i-1$ and i, we have $\tilde{Y}_{i-1,i} = Y_{i-1}$, and subsequently D_i happens if $\{G_i > Y_{i-1}\}$. Let $B_{i,j}$ represent the event where packet j is blocked at the transmit server by transmitted packet $i-1$, i.e., $B_{i,j}$ happens if $\{G_{i,j} < \tilde{Y}_{i-1,j}\}$. Since the processing of the new status update starts immediately after the previous one was processed, and a processed packet could be blocked by the transmit server, the inter-sampling time between two delivered packets $i-1$ and i is the process server service time of packet $i-1$, G_{i-1}, and the summation of the process server service time of the blocked packets by transmitted pack $i-1$. Let N denote a random variable representing the number of blocked packets at the transmit server by transmitted packet $i-1$. Then, the inter-sampling time is given as

$$I_i = G_{i-1} + G_{i,1} + \cdots + G_{i,N}. \tag{11}$$

The total service time for ith packet is given as

$$T_i = G_i + Y_i. \tag{12}$$

The mean of the ith inter-sampling time, $\mathbb{E}\{I_i\}$, is derived as

$$\begin{aligned}\mathbb{E}\{I_i\} &= \mathbb{E}\{\mathbb{E}\{I_i \mid N = n\}\}\\ &= \sum_{n=0}^{\infty} \mathbb{E}\{G_{i-1} + \sum_{k=1}^{n} G_{i,k}\}\Pr(B_{i,1}, \cdots, B_{i,n}, D_i)\\ &\overset{(a)}{=} \sum_{n=0}^{\infty} \frac{n+1}{\gamma}\Pr(B_{i,1}, \cdots, B_{i,n}, D_i),\end{aligned} \tag{13}$$

where (a) comes from the fact that the service time of packets in the process server are independent and identically distributed with mean $1/\gamma$. Note that $n = 0$ represents the case that blocking does not happen, i.e., $\{G_i > Y_{i-1}\}$. The remaining task is to characterize $\Pr(B_{i,1}, \cdots, B_{i,n}, D_i)$.

Due to the memoryless property of the geometric random variables, the residual service times have the same distribution as Y_{i-1} and are independent of each other. Consequently, events $D_i, B_{i,1}, \cdots, B_{i,n}$ become independent of each other. Thus, we have

$$\Pr(B_{i,1}, \cdots, B_{i,n}, D_i) = \Pr(B_{i,1}) \cdots \Pr(B_{i,n})\Pr(D_i).$$

In the following lemma, we derive the probability of event $B_{i,j}$.

Lemma 1. *Let $\bar{p} = 1-p$ and $\bar{\gamma} = 1-\gamma$, then, the probability of event $B_{i,j}$,* $\Pr(B_{i,j})$, *is given as*

$$\Pr(B_{i,j}) = \frac{\gamma\bar{p}}{1-\bar{\gamma}\bar{p}}. \tag{14}$$

Proof. Let z_1 denote the processing time of the jth blocked packet. The probability of event $B_{i,j} = \{G_{i,j} < \tilde{Y}_{i-1,j}\}$ is calculated as

$$\Pr(B_{i,j}) = \sum_{z_1=1}^{\infty} \Pr(G_{i,j} < \tilde{Y}_{i-1,j} \mid G_{i,j} = z_1)\Pr(G_{i,j} = z_1).$$

The probability that the processing time takes z_1 slots is given as $\Pr(G_{i,j} = z_1) = \gamma\bar{\gamma}^{z_1-1}$, and the probability that the residual transmission time takes more than z_1 slots is given as $\Pr(\tilde{Y}_{i-1,j} > z_1) = \bar{p}^{z_1}$. Thus, we have

$$\begin{aligned}\Pr(B_{i,j}) &= \sum_{z_1=1}^{\infty} \bar{p}^{z_1}\gamma\bar{\gamma}^{z_1-1}\\ &\overset{(a)}{=} \frac{\gamma\bar{p}}{1-\bar{\gamma}\bar{p}},\end{aligned} \tag{15}$$

where (a) follows since $\sum_{z_1=1}^{\infty} l^{z_1} = l/(1-l)$ for all $|l| < 1$. $\square$

Following the same steps as for the proof of Lemma 1, one can show that $\Pr(D_i) = 1 - \gamma\bar{p}/(1-\bar{\gamma}\bar{p})$. As $\Pr(B_{i,j})$ and $\Pr(D_i)$ are the same for different i and j, hereinafter, for simplicity of presentation, we use $P_B = \Pr(B_{i,j}),\ \forall i, j$, and $P_D = \Pr(D_i),\ \forall i$.

Using Lemma 1 and equation (13) the mean of the ith inter-sampling time is derived as

$$\begin{aligned}\mathbb{E}\{I_i\} &= \sum_{n=1}^{\infty} \frac{n}{\gamma} P_D \big(P_B\big)^{n-1}\\ &\overset{(a)}{=} \frac{P_D}{\gamma P_B}\frac{P_B}{P_D^2}\\ &= \frac{1}{\gamma P_D},\end{aligned} \tag{16}$$

where (a) follows since $\sum_{z_1=1}^{\infty} z_1 l^{z_1} = l/(1-l)^2$ for all $|l| < 1$, and $P_D = 1 - P_B$.

Using (11), the second moment of the inter-sampling time is derived as

$$\begin{aligned}\mathbb{E}\{I_i^2\} &= \mathbb{E}\{\mathbb{E}\{I_i^2 \mid N = n\}\}\\ &\overset{a}{=} \sum_{n=0}^{\infty} \mathbb{E}\{\big(G_{i-1}^2 + 2G_{i-1}\sum_{k=1}^{n} G_{i,k} + \sum_{k=1}^{n} G_{i,k}^2\\ &\quad + 2\sum_{k=1}^{n}\sum_{z1=1}^{k-1} G_{i,k}G_{i,z1}\}P_D P_B^n\\ &\overset{b}{=} \sum_{n=1}^{\infty} n\frac{2-\gamma}{\gamma^2}P_D P_B^{n-1} + \sum_{n=1}^{\infty} \frac{n(n-1)}{2}\frac{2}{\gamma^2}P_D P_B^{n-1}\\ &\overset{c}{=} \frac{2-\gamma}{\gamma^2}\frac{1}{P_D^2}P_D\\ &\quad + \sum_{n=1}^{\infty}\frac{n^2}{\gamma^2}P_D P_B^{n-1} - \sum_{n=1}^{\infty}\frac{n}{\gamma^2}P_D P_B^{n-1}\\ &\overset{d}{=} \frac{2-\gamma}{\gamma^2 P_D} + \frac{P_D}{P_B\gamma^2}\frac{P_B^2+P_B}{P_D^3} - \frac{P_D}{P_B\gamma^2}\frac{P_B}{P_D^2}\\ &= \frac{2-\gamma}{\gamma^2 P_D} + \frac{P_B+1}{\gamma^2 P_D^2} - \frac{1}{\gamma^2 P_D}\\ &= \frac{1-\gamma}{\gamma^2 P_D} + \frac{P_B+1}{\gamma^2 P_D^2},\end{aligned} \tag{17}$$

where (a) follows since $(\sum_{n=1}^{\infty} l_n)^2 = \sum_{n=1}^{\infty} l_n^2 + 2\sum_{n=1}^{\infty}\sum_{z1=1}^{n-1} l_n l_{z_1}$, (b) follows since $\sum_{z_1=1}^{Z}\sum_{z_2=1}^{z1-1} l = lZ(Z-1)/2$, (c) follows since $\sum_{z_1=1}^{\infty} z_1 l^{z_1} = l/(1-l)^2$ for all $|l| < 1$, (d) follows since $\sum_{z_1=1}^{\infty} z_1^2 l^{z_1} = l(l+1)/(1-l)^3$ for all $|l| < 1$.

The value of $\mathbb{E}\{I_i T_i\}$ is given as

$$\begin{aligned}\mathbb{E}\{I_i T_i\} &\overset{a}{=} \mathbb{E}\{I_i\}\mathbb{E}\{G_i + Y_i\}\\ &= \frac{1}{\gamma P_D}(\frac{1}{\gamma} + \frac{1}{p})\\ &= \frac{1}{P_D\gamma^2} + \frac{1}{P_D p\gamma},\end{aligned} \tag{18}$$

where (a) holds because I_i, G_i, and T_i are independent.

Finally, by substituting (16), (17), and (18) in (5), the average AoI for zero-wait-blocking policy is derived as

$$\bar{\Delta} = \frac{\frac{1}{2}(\frac{1-\gamma}{\gamma^2 P_D} + \frac{P_B+1}{\gamma^2 P_D^2}) + \frac{1}{\gamma^2 P_D} + \frac{1}{P_D p \gamma}}{\frac{1}{\gamma P_D}} - \frac{1}{2}$$
$$= \frac{1}{2}(\frac{1-\gamma}{\gamma} + \frac{P_B+1}{\gamma P_D}) + \frac{1}{\gamma} + \frac{1}{p} - \frac{1}{2}.$$

IV. Optimal Control Policy

In this section, we propose an optimal control policy. Specifically, we consider the expected long-term time average AoI, given as

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\{\Delta(t)\},$$

and find the optimal action, $a(t)$, at each slot to minimize the average AoI. Note that we use $\lim\sup$ instead of $\lim$ to ensure the limit exists. We formulate our problem as a MDP problem and solve it with the RVIA.

A. AoI Model

For the purpose of deriving an optimal control policy, we use the common assumption (see, e.g., [32], [33], [40]) that the age of the packets in the system can be upper bounded by a value $\Delta^{\max}$, i.e., $\Delta(t) = \min\{t - U(t), \Delta^{\max}\}$. Besides making the problem tractable, this also models the fact that once the available information about the process of interest becomes excessively stale (i.e., age reached $\Delta^{\max}$), further increasing the age would be irrelevant.

In order to characterize the evolution of AoI at the sink, it is useful to define the age of the under-serving packet at each server. Let $\Delta_s(t)$ denote the age of the under-processing sample at the process server at time t. In the following, we elaborate on the evolution of the age of packets at the servers.

Suppose first the process server is busy at slot t, i.e., $g(t) = 0$; in the next slot, there are three possible cases: 1) the server remains busy at the next slot, $g(t+1) = 0$, and the age of the under-processing sample at the next slot becomes $\min\{\Delta_s(t)+1, \Delta^{\max}\}$, 2) at the next slot the server becomes idle, $g(t+1) = 1$, and the controller takes a sample, $a(t+1) = 1$; then the age of the under-processing sample at the next slot becomes zero, and 3) at the next slot the server becomes idle, $g(t+1) = 1$, and the controller does not request a new sample, $a(t+1) = 0$; in this case, the server is empty and the age is undefined.

Consider now the case that at slot t, the process server is idle, $g(t) = 1$, and the controller takes a new sample, $a(t) = 1$; in the next slot there are three possible cases: 1) the process server does not finish the service, $g(t+1) = 0$, and, therefore, the age of the under-processing sample at the next slot becomes one, 2) the server finishes the service in one slot (i.e., slot t) and the process server becomes available, $g(t+1) = 1$, and the controller takes a new sample, $a(t+1) = 1$, then the age of the under-processing sample at the next slot becomes zero, and 3) as before, the server finishes the service at slot t and become available at slot $t+1$, $g(t+1) = 1$, but the controller does not request a new sample, $a(t+1) = 0$; in this case the server is empty and, therefore, the age is undefined. Note that, in general, if the process server is available and the controller does not request a new sample at a slot, the age is undefined at the slot. For the sake of conciseness, we introduce the short notation $I_s(t) = (g(t), a(t), g(t+1), a(t+1))$. The evolution of $\Delta_s(t)$ is given as

$$\Delta_s(t+1) = \begin{cases} \min\{\Delta_s(t)+1, \Delta^{\max}\}, & I_s(t) = (0,*,0,*) \\ 0, & I_s(t) = (0,*,1,1) \\ 1, & I_s(t) = (1,1,0,*) \\ 0, & I_s(t) = (1,1,1,1) \\ *, & I_s(t) = (*,*,1,0), \end{cases} \tag{19}$$

where the notation $*$ means that either 1) the variable's value does not affect the age value, or 2) the age value is undefined (i.e., the server is empty).

Let $\Delta_x(t)$ denote the age of the under-transmitting packet at the transmit server at time t. Let $n(t)$ be an indicator of a new packet arrival at the transmit server, i.e., $n(t) = 1$ indicates that a new packet arrived at the transmit server at slot t, and $n(t) = 0$ otherwise.

Suppose first that the transmit server is busy at slot t, $y(t) = 0$; at the next slot, there are three possible cases: 1) the transmit server remains busy, $y(t+1) = 0$, then the age of the under-transmitting packet becomes $\min\{\Delta_x(t)+1, \Delta^{\max}\}$, 2) the server becomes available, $y(t+1) = 1$, and receives a new packet, $n(t+1) = 1$, then the age of the under-transmitting packet becomes $\min\{\Delta_s(t)+1, \Delta^{\max}\}$, and 3) the server becomes available, $y(t+1) = 1$, and receives no packet, $n(t+1) = 0$, then as the transmit server is empty the value of the age becomes undefined.

Now consider the case that the transmit server is available at slot t, $y(t) = 1$, there are five possible cases: 1) the transmit server receives a new packet at slot t, $n(t) = 1$, and becomes busy at the next slot, $y(t+1) = 0$, then the age of the under-transmitting packet becomes $\min\{\Delta_x(t)+1, \Delta^{\max}\}$, 2) the transmit server receives a new packet at slot t, $n(t) = 1$, becomes available at the next slot, $y(t+1) = 1$, and receives a new packet at the next slot, $n(t+1) = 1$, then the age of the under-transmitting packet becomes one, 3) the transmit server receives a new packet at slot t, $n(t) = 1$, becomes available at the next slot, $y(t+1) = 1$, but it does not receive a new packet at the next slot, $n(t+1) = 0$, then the age of the under-transmitting packet becomes undefined, 4) the transmit server receives no packet at slot t, $n(t) = 0$, remains available at the next slot, $y(t+1) = 1$, and receives a new packet at the next slot, $n(t+1) = 1$, then the age of the under-transmitting packet becomes $\min\{\Delta_s(t)+1, \Delta^{\max}\}$, and 5) the transmit server receives no packet at slot t, $n(t) = 0$, becomes available at the next slot, $y(t+1) = 1$, and receives no packet at the next slot, $n(t+1) = 0$, then the age of the under-transmitting packet becomes undefined. In general, if the transmit server is available and does not receive a new packet at a slot, the age is undefined at the slot. For the sake of brevity, we introduce the short notion $I_x(t) = (y(t), n(t), y(t+1), n(t+1))$. The evolution of $\Delta_x(t)$ is given as

$$\Delta_x(t+1) = \begin{cases} \min\{\Delta_x(t)+1, \Delta^{\max}\}, & I_x(t) = (0,*,0,*) \\ \min\{\Delta_s(t)+1, \Delta^{\max}\}, & I_x(t) = (0,*,1,1) \\ \min\{\Delta_x(t)+1, \Delta^{\max}\}, & I_x(t) = (1,1,0,*) \\ 1, & I_x(t) = (1,1,1,1) \\ \min\{\Delta_s(t)+1, \Delta^{\max}\}, & I_x(t) = (1,0,1,1) \\ *, & I_x(t) = (*,*,1,0). \end{cases} \quad (20)$$

Having defined Δ_s and Δ_x, we now characterize the AoI at the sink. Let $r(t)$ be an indicator of a new packet arrival at the sink at slot t, where $r(t) = 1$ indicates that the sink receives a new packet at slot t, and $r(t) = 0$ otherwise. If the sink receives a new packet at slot t, AoI becomes $\min\{\Delta_x(t-1)+1, \Delta^{\max}\}$, otherwise AoI is incremented by one up to $\Delta^{\max}$. The evolution of $\Delta(t)$ is given as

$$\Delta(t+1) = \begin{cases} \min\{\Delta_x(t)+1, \Delta^{\max}\}, & r(t+1) = 1 \\ \min\{\Delta(t)+1, \Delta^{\max}\}, & r(t+1) = 0. \end{cases} \quad (21)$$

B. MDP Model

An MDP is defined by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, C\}$ of four elements: State space, action space, state transition probabilities, and an (immediate) cost, which are elaborated below:

State: Let $s(t) \in \mathcal{S}$ denote the state of the system at slot t, where $\mathcal{S}$ is the the state space. The state of the system is given as $s(t) = \{g(t), \Delta_s(t), y(t), \Delta_x(t), \Delta(t)\}$.

Action: The action at slot t is defined as $a(t) \in \mathcal{A} = \{0, 1\}$, where $\mathcal{A}$ is the action space.

Immediate cost: The immediate cost is the AoI, i.e., $C(t) = \Delta(t)$.

State transition probabilities: Let $\mathcal{P}_{ss'}(a) = \Pr(s' \mid s, a)$ denote the probability of moving from current state s to a next state s' under action a. Recall that we assume a blocking strategy, i.e., while the server is busy, any new request for service is blocked and discarded. In the following, we calculate the state transition probabilities for different actions and four general states: $s_1 = \{1,*,1,*,\Delta\}$, $s_2 = \{1,*,0,\Delta_x,\Delta\}$, $s_3 = \{0,\Delta_s,1,*,\Delta\}$, and $s_4 = \{0,\Delta_s,0,\Delta_x,\Delta\}$. Let us denote $\bar{\gamma} = 1-\gamma$, $\bar{p} = 1-p$. The state transition probabilities for different actions can be expressed as

$$\begin{aligned}
&\Pr\big(\{1,*,1,1,\Delta+1\} \mid s_1, a=1\big) = \gamma \\
&\Pr\big(\{0,1,1,*,\Delta+1\} \mid s_1, a=1\big) = \bar{\gamma} \\
&\Pr\big(\{1,*,1,*,\Delta+1\} \mid s_1, a=0\big) = 1 \\
&\Pr\big(\{1,*,1,1,\Delta_x+1\} \mid s_2, a=1\big) = \gamma p \\
&\Pr\big(\{1,*,0,\Delta_x+1,\Delta+1\} \mid s_2, a=1\big) = \gamma \bar{p} \\
&\Pr\big(\{0,1,1,*,\Delta_x+1\} \mid s_2, a=1\big) = \bar{\gamma} p \\
&\Pr\big(\{0,1,0,\Delta_x+1,\Delta+1\} \mid s_2, a=1\big) = \bar{\gamma}\bar{p} \\
&\Pr\big(\{1,*,1,*,\Delta_x+1\} \mid s_2, a=0\big) = p \\
&\Pr\big(\{1,*,0,\Delta_x+1,\Delta+1\} \mid s_2, a=0\big) = \bar{p} \\
&\Pr\big(\{1,*,1,\Delta_s+1,\Delta+1\} \mid s_3, a=0\big) = \gamma \\
&\Pr\big(\{0,\Delta_s+1,1,*,\Delta+1\} \mid s_3, a=0\big) = \bar{\gamma} \\
&\Pr\big(\{1,*,1,\Delta_s+1,\Delta_x+1\} \mid s_4, a=0\big) = \gamma p \\
&\Pr\big(\{1,*,0,\Delta_x+1,\Delta+1\} \mid s_4, a=0\big) = \gamma \bar{p} \\
&\Pr\big(\{0,\Delta_s+1,1,*,\Delta_x+1\} \mid s_4, a=0\big) = \bar{\gamma} p \\
&\Pr\big(\{0,\Delta_s+1,0,\Delta_x+1,\Delta+1\} \mid s_4, a=0\big) = \bar{\gamma}\bar{p}, \quad (22)
\end{aligned}$$

whereas the other cases are zero.

Theorem 1. *The defined MDP is unichain.*

Proof. An MDP is unichain if, under every deterministic policy, it induces a single recurrent class plus a possibly empty set of transient states [41]. It is sufficient to show that under any feasible deterministic policy, there exists a recurrent state, i.e., a state that is accessible from every other states [42, Exercise 4.3]. For the case that $\gamma \neq 1$ and $0 < p \leq 1$, the recurrent state is $\{0, \Delta^{\max}, 1, *, \Delta^{\max}\}$. This is due to the fact that the process server would have a packet, and the probability of the event of generating a packet in $\Delta^{\max}$ consecutive slots is non-zero. For the case that $p \neq 1$ and $0 < \gamma \leq 1$, the recurrent state is $\{1, *, 0, \Delta^{\max}, \Delta^{\max}\}$. This is because the probability of the event of generating a packet at one slot and unsuccessful transmission of a packet in $\Delta^{\max}$ consecutive slots are non-zero[1]. Thus, the defined MDP is unichain for every deterministic policy. □

Let π denote a policy that determines the action taken at each state. A stationary randomized policy is mapping from each state to a distribution over actions, $\pi(a \mid s) : \mathcal{S} \times \mathcal{A} \to [0,1]$, $\sum_{a \in \mathcal{A}_s} \pi(a \mid s) = 1$. A (stationary) deterministic policy chooses an action at a given state with probability one, which is a special case of the stationary randomized policy. With a slight abuse of notation, we denote the action taken in state s by a deterministic policy π with $\pi(s)$. Let $\bar{C}^{\pi} = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\{C(t)\}$ denote the average AoI cost obtained under policy π. Due to the unichain structure, the time average expected value of the cost is independent from the initial state [43, Proposition 8.2.1]. Having defined the MDP elements, the MDP problem is given as

$$\underset{\pi}{\text{minimize}} \quad \bar{C}^{\pi}. \quad (23)$$

The optimal value of problem (23) is denoted by C^*, and an optimal policy is denoted by π^*.

In the next subsection, we solve problem (23).

C. Optimal Control Policy

In this section, we solve MDP problem (23) using the RVIA [41, Section 4.3]. According to [43, Theorem 8.4.3, Theorem 8.4.4], for an MDP with 1) finite state and action space and 2) unichain structure, there exists a relative value function $h(s),\ s \in \mathcal{S}$, that satisfies

$$C^* + h(s) = \min_{a \in \mathcal{A}} \Big[C(s,a) + \sum_{s' \in \mathcal{S}} \Pr(s' \mid s, a) h(s')\Big], \ \forall s \in \mathcal{S}.$$

[1]Since in the case that $\gamma = p = 1$, the optimal policy is to sample at each slot, we do not study this case.

Algorithm 1: The RVIA to solve MDP problem (23)

```
   Input: $\mathcal{P}$, $s^{\rm ref}$, and $\epsilon$
   Initialize: 1) $i = 1$, 2) set $h^0(s) = 1$, $h^1(s) = 0$,
               $Q^0(s) = 0$ for all $s \in \mathcal{S}$
1  while $\max_{s\in\mathcal{S}} |h^i(s) - h^{i-1}(s)| \geq \epsilon$ do
2      $i = i + 1$
3      for $s \in \mathcal{S}$ do
4          $Q^i(s) = \min_{a\in\mathcal{A}} \left[C(s,a) + \sum_{s'\in\mathcal{S}} \Pr(s'|s,a)h^{i-1}(s')\right]$
5          $h^i(s) = Q^i(s) - Q^i(s^{\rm ref})$
6      end
7  end
8  for $s \in \mathcal{S}$ do
9      $\pi^*(s) = \arg\min_{a\in\mathcal{A}} \left[C(s,a) + \sum_{s'\in\mathcal{S}} \Pr(s' \mid s,a)h^i(s')\right]$
10 end
```

Subsequently, the optimal policy, π^*, is obtained as [43, Theorem 8.4.4]

$$\pi^*(s) = \arg\min_{a\in\mathcal{A}} \left[C(s,a) + \sum_{s'\in\mathcal{S}} \Pr(s' \mid s,a)h(s')\right], \ \forall s \in \mathcal{S}.$$

The RVIA is an iterative algorithm in which the relative value function is updated at each iteration $i \in \{0, 1, \cdots\}$. Let $Q^i(s)$ denote the value function given as

$$Q^i(s) = \min_{a\in\mathcal{A}} \left[C(s,a) + \sum_{s'\in\mathcal{S}} \Pr(s' \mid s,a)h^{i-1}(s')\right].$$

The relative value function is updated at each iteration as $h^i(s) = Q^i(s) - Q^i(s^{\rm ref})$, where $s^{\rm ref} \in \mathcal{S}$ is an arbitrarily chosen reference state which remains unchanged throughout the iterations.

The RVIA is presented in Algorithm 1, where $\epsilon \in (0, 1)$ is a small constant for the RVIA termination criterion.

Theorem 2. *For any state $s \in \mathcal{S}$ and initialization $Q^0(s)$, the sequences $\{h^i(s)\}_{i=1,2,\cdots}$ and $\{Q^i(s)\}_{i=1,2,\cdots}$ converge, i.e., $\lim_{i\to\infty} h^i(s) = h(s)$ and $\lim_{i\to\infty} Q^i(s) = Q(s)$.*

Proof. According to [41, Proposition 4.3.2], it is sufficient to show that the MDP under every deterministic policy induces a Markov chain that is unichain and aperiodic. In Theorem 1, it was proven that the MDP is unichain. If an state in a recurrent class is aperiodic then all the state in the class are aperiodic [42, Theorem 4.2.8]. As each recurrent state can move to itself with a non-zero probability, the MDP is aperiodic under every deterministic policy. □

V. Numerical Results

In this section, we evaluate the proposed optimal control policy and the fixed control policies: 1) zero-wait-one policy and 2) zero-wait-blocking policy. Furthermore, we numerically analyze the structure of the optimal policy. To run Algorithm 1, we set the maximum value of the AoI $\Delta^{\max} = 50$ and the stopping criterion threshold $\epsilon = 0.001$. The rest of the parameters are specified in each figure.

A. Comparison of the Policies

Fig. 5 illustrates the average AoI for the presented policies: 1) zero-wait-one, 2) zero-wait-blocking, and 3) optimal control policy, versus the transmit server service rate, p, with three process server service rates, γ. We also show the AoI obtained by the case where the servers are able to preempt the under-service sample/packet when they receive a new task. Since the service times are memoryless, it is clear that the optimal control policy is to always preempt the under-service sample/packet whenever a new one arrives; thus, we refer to this policy as the optimal policy preemption.

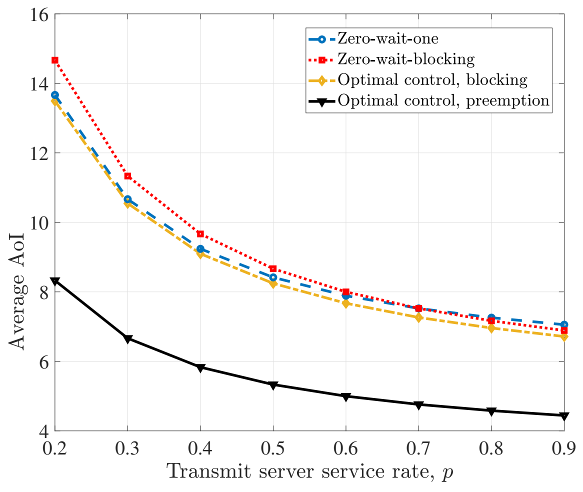
According to Fig. 5, when the servers' service rates increase, the average AoI decreases. This is because the controller can update the sink more frequently with a small number of blocking. As we expected, the optimal control policy outperforms the fixed policies. When the service rates are large (e.g., $p = 0.9$ and $\gamma = 0.7$), the zero-wait-blocking policy performance is very close to the optimal policy. It is because the probability of blocking a packet is considerably reduced, and thus the zero-wait-blocking policy frequently delivers packets to the sink. The zero-wait-one policy outperforms the zero-wait-blocking policy for a small transmit server service rate (e.g., $p = 0.2$), however, in the large service rate, the zero-wait-blocking policy outperforms the zero-wait-one policy. It is because in a low transmit server service rate, the probability of blocking a packet increases, and vice versa.

B. Structure of Optimal Policy

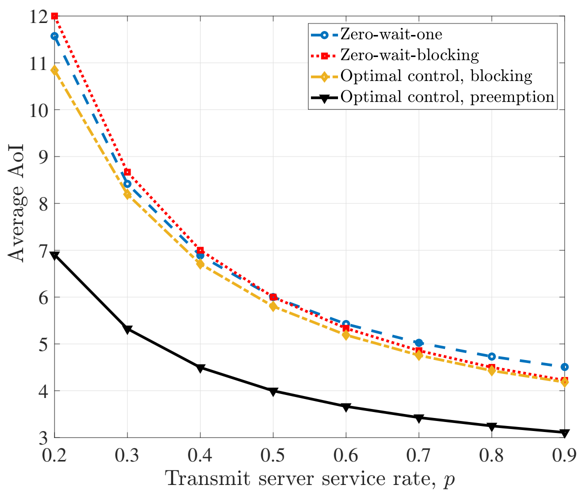
Fig. 6 demonstrates the structure of the optimal policy in the case that the system is empty, i.e., $s_1 = \{1, *, 1, *, \Delta\}$ by showing the optimal action for the different values of γ and p.

According to Fig. 6, the optimal policy in these states has a threshold structure with respect to the AoI, Δ. For example, when $p = 0.2$ and $\gamma = 0.3$, the threshold value for AoI is $\Delta = 4$: the optimal action for $\Delta \leq 4$ is to stay idle, $a = 0$, and the optimal action for $\Delta > 4$ is to take a sample, $a = 1$. The threshold value decreases when the process server service rate, γ, increases. This behavior is due to the fact that when the processing time of the packet is decreased for the same transmit server rate, the probability of blocking is increased and puts the process server out of access. Note that the optimal action after the threshold value remains the same. Overall, these results show that taking a sample when the system is idle for most of the cases is optimal. However, with low service rates when the AoI is small, in line with [16], the optimal action is to stay idle.

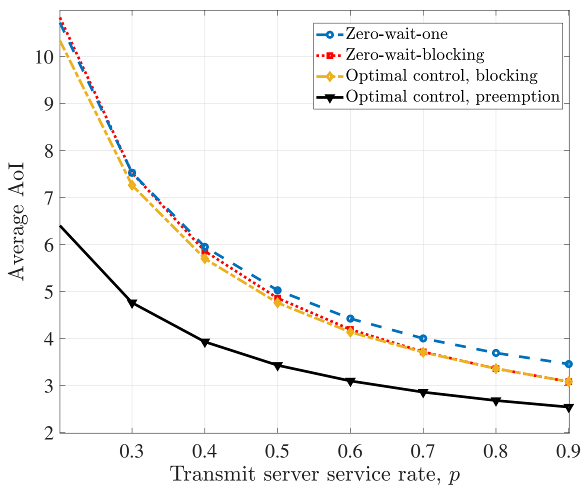
Furthermore, in Fig. 7, we analyze the optimal policy structure for the case that the transmit server is busy and the process server is idle, i.e., $s_2 = \{1, *, 0, \Delta_x, \Delta\}$. The results shows that the optimal policy in these states has a threshold structure with respect to the AoI at the transmit server, Δ_x. For example, when $p = 0.4$ and $\gamma = 0.5$, the threshold value for the AoI at the transmit process is $\Delta_x = 3$: the optimal



(a) $\gamma = 0.3$

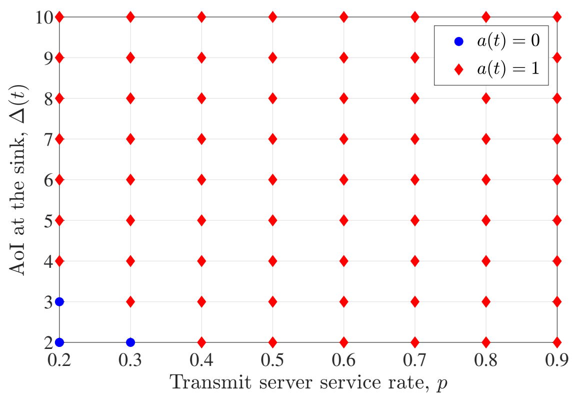


(b) $\gamma = 0.5$

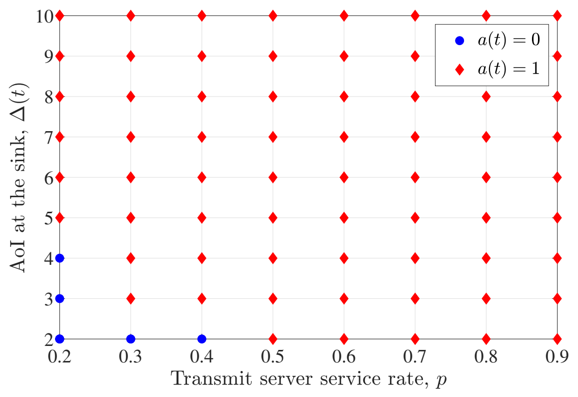


(c) $\gamma = 0.7$

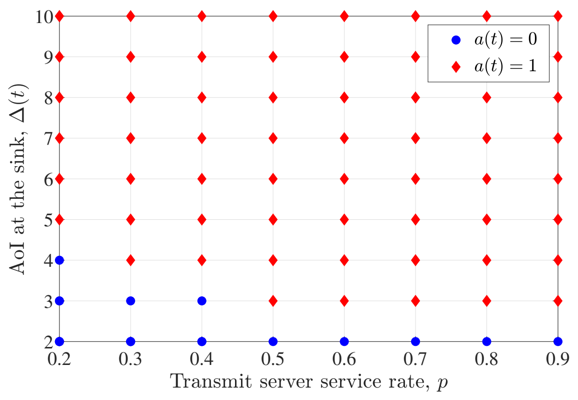
Fig. 5. Average AoI for different policies and the process server service rate with respect to the transmit server service rate with $\epsilon = 1e-3$.



(a) $\gamma = 0.7$



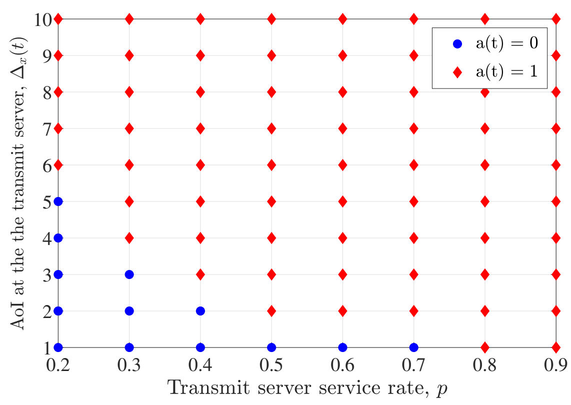
(b) $\gamma = 0.5$



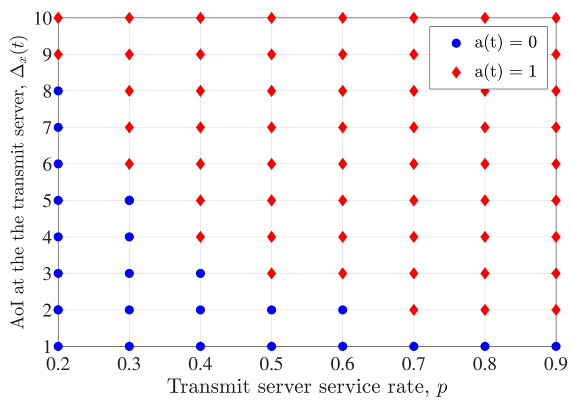
(c) $\gamma = 0.3$

Fig. 6. Structure of the optimal policy for state $s_1 = \{0, *, 0, *, \Delta\}$ with respect to the AoI, Δ, and the transmit server rate, p, for three different process server rates, γ.

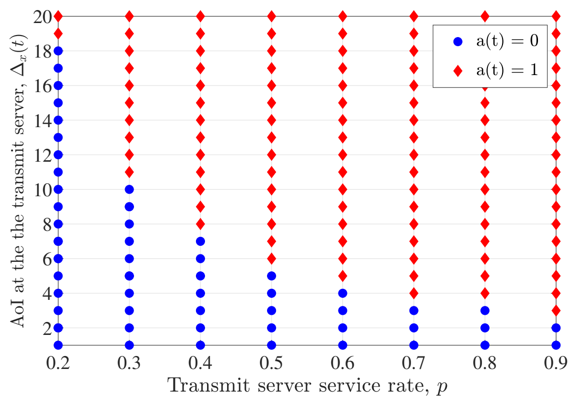
action for $\Delta \leq 3$ is to stay idle, $a = 0$, and the optimal action for $\Delta > 3$ is to take a sample, $a = 1$. When the process server rate, γ, decreases, the threshold value increases. This behavior is because in a low process server rate, taking a sample puts the process server for a bigger interval out of access. In addition, when the transmit server rate increases, the threshold values are decreased. It is because the probability of blocking is reduced. The optimal action remains unchanged for the values larger than the threshold value. Overall, it can be concluded that taking a sample when the process server is idle while the transmit server is busy is not optimal for the



(a) $\gamma = 0.7$



(b) $\gamma = 0.5$



(c) $\gamma = 0.3$

Fig. 7. Structure of the optimal policy for state $s_1 = \{0, *, 1, \Delta_x, \Delta\}$ with respect to the AoI at the transmit server, Δ_x, and the transmit server rate, p, for three different process server service rates, γ.

low service rates.

VI. Conclusion

We considered a time-slotted computation-intensive status update system. We derived the average AoI expression for two fixed policies: 1) zero-wait-one, i.e., the controller takes a new sample whenever both servers are available, and 2) zero-wait-blocking, i.e., the controller takes a sample whenever the process server is idle and block any newly arrived packet to the transmit server while it is busy. Furthermore, we proposed an optimal control policy to minimize AoI using the MDP technique. In numerical results, we analyze the optimal policy structure for two different states of the system: 1) the servers are empty, and 2) the process server is available while the transmit server is busy. The results reveal that, in the first state, it is not optimal to take a sample when AoI and the service rates are small. Similarly, for the second case, it is not optimal to take a sample when AoI at the transmit server and the service rates are small. Furthermore, we compare the performance of the optimal and fixed policies. It was shown that the zero-wait-one policy outperforms the zero-wait-blocking policy in the small service rates. However, in the larger service rates, the zero-wait-blocking policy outperforms the zero-wait-one policy and performs very close to the optimal policy.

References

[1] R. D. Yates *et al.*, "Age of information: An introduction and survey," *IEEE J. Select. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[2] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *IEEE Trans. Inform. Theory*, vol. 66, no. 9, pp. 5712–5728, Sep. 2020.

[3] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the role of age of information in the Internet of things," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 72–77, Dec. 2019.

[4] A. Kosta *et al.*, "Age of information: A new concept, metric, and tool," *Foundations and Trends® in Networking*, vol. 12, no. 3, pp. 162–259, Nov. 2017.

[5] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4353–4366, Feb. 2020.

[6] P. Zou, O. Ozel, and S. Subramaniam, "Optimizing information freshness through computation–transmission tradeoff and queue management in edge computing," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 949–963, Feb. 2021.

[7] F. Chiariotti, O. Vikhrova, B. Soret, and P. Popovski, "Peak age of information distribution for edge computing with wireless links," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3176–3191, Jan. 2021.

[8] A. Muhammad, I. Sorkhoh, M. Samir, D. Ebrahimi, and C. Assi, "Minimizing age of information in multi-access edge computing-assisted IoT networks," *IEEE Internet Things J.*, Dec. 2021.

[9] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, "Mobi-IoST: Mobility-aware cloud-fog-edge-IoT collaborative framework for time-critical applications," *IEEE Trans. Net. science Eng.*, vol. 7, no. 4, pp. 2271–2285, Sep. 2019.

[10] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Jan. 2020.

[11] K. Maharatna, E. B. Mazomenos, J. Morgan, and S. Bonfiglio, "Towards the development of next-generation remote healthcare system: Some practical considerations," in *Proc. IEEE ISCAS*, May 2012.

[12] A. H. Sodhro *et al.*, "Power control algorithms for media transmission in remote healthcare systems," *IEEE Acc.*, vol. 6, pp. 42384–42393, Jul. 2018.

[13] C. De Alwis *et al.*, "Survey on 6G frontiers: Trends, applications, requirements, technologies, and future research," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, Apr. 2021.

[14] L. Hu *et al.*, "Ready player one: UAV-clustering-based multi-task offloading for vehicular VR/AR gaming," *IEEE Network*, vol. 33, no. 3, pp. 42–48, May 2019.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[16] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *Proc. ISIT*, Jun. 2015.

[17] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inform. Theory*, vol. 63, no. 11, pp. 7492–7508, Aug. 2017.

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

[19] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of information of multiple sources with queue management," in *Proc. IEEE ICC*, Jun. 2015.

[20] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age of information performance of multiaccess strategies with packet management," *J. Commun. Netw.*, vol. 21, no. 3, pp. 244–255, 2019.

[21] M. Moradian and A. Dadlani, "Age of information in scheduled wireless relay networks," in *Proc. IEEE WCNC*, May 2020.

[22] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Non-linear age of information in a discrete time queue: Stationary distribution and average performance analysis," in *Proc. IEEE ICC*, Jun. 2020.

[23] J. Zhang and Y. Xu, "On age of information for discrete time status updating system with Ber/G/1/1 queues," in *Proc. IEEE ITW*, Oct. 2021.

[24] D. Qiao and M. C. Gursoy, "Age-optimal power control for status update systems with packet-based transmissions," *IEEE Wireless Commun. Lett.*, vol. 8, no. 6, pp. 1604–1607, Jul. 2019.

[25] C.-C. Wang, "How useful is delayed feedback in AoI minimization — A study on systems with queues in both forward and backward directions," in *Proc. IEEE ISIT*, Jun. 2022.

[26] M. Moltafet, M. Leinonen, M. Codreanu, and R. D. Yates, "Status update control and analysis under two-way delay," 2022, [Online]. Available: https://doi.org/10.48550/arXiv.2208.06177.

[27] S. Wu *et al.*, "Minimizing age-of-information in HARQ-CC aided NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1072–1086, Sep. 2022.

[28] Y. Wang *et al.*, "Age-optimal transmission policy with HARQ for freshness-critical vehicular status updates in space-air-ground integrated networks," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5719–5729, Apr. 2022.

[29] E. Gindullina, L. Badia, and D. Gündüz, "Age-of-information with information source diversity in an energy harvesting system," *IEEE Trans. Green Commun. Net.*, vol. 5, no. 3, pp. 1529–1540, Sep. 2021.

[30] J. Liu, R. Zhang, A. Gong, and H. Chen, "Optimizing age of information in wireless uplink networks with partial observations," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 4105–4118, Jul. 2023.

[31] 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, Sep. 2022.

[32] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks with HARQ," *IEEE J. Select. Areas Commun.*, vol. 39, no. 5, pp. 1412–1426, May 2021.

[33] 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, Jan. 2022.

[34] Z. Chen, N. Pappas, E. Björnson, and E. G. Larsson, "Optimizing information freshness in a multiple access channel with heterogeneous devices," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 456–470, Mar. 2021.

[35] J. Pan, A. M. Bedewy, Y. Sun, and N. B. Shroff, "Optimal sampling for data freshness: Unreliable transmissions with random two-way delay," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 408–420, Aug. 2023.

[36] S. S. Vilni, M. Moltafet, M. Leinonen, and M. Codreanu, "Average AoI minimization in an HARQ-based status update system under random arrivals," in *Proc. IEEE IoTaIS*, Nov. 2022.

[37] M. Moltafet, M. Leinonen, and M. Codreanu, "Moment generating function of the AoI in multi-source systems with computation-intensive status updates," in *Proc. IEEE ITW*, Nov. 2021.

[38] H. Hu, Y. Dong, Y. Jiang, Q. Chen, and J. Zhang, "On the age of information and energy efficiency in cellular IoT networks with data compression," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5226–5239, Nov. 2022.

[39] X. Qin *et al.*, "Timeliness of information for computation-intensive status updates in task-oriented communications," *IEEE J. Select. Areas Commun.*, vol. 41, no. 3, pp. 623–638, Dec. 2022.

[40] A. Maatouk, S. Kriouile, M. Assad, and A. Ephremides, "On the optimality of the Whittle's index policy for minimizing the age of information," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1263–1277, Oct. 2021.

[41] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. vol. 2, Bermont, MA: Athena Scientific, 2007.

[42] R. G. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013.

[43] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Saeid Sadeghi Vilni (Student Member, IEEE) received the M.Sc. (Tech.) degree in Electrical Engineering from the Tarbiat Modares University, Tehran, Iran, in 2017. He is currently pursuing the Ph.D. degree with the Centre for Wireless Communications, University of Oulu, Finland. His current research interests include age of information, machine learning for wireless applications, and radio resource allocation and optimization in wireless networks.

Mohammad Moltafet (S'16-M'21) received the M.Sc. degree in Communications Engineering from Tarbiat Modares University, Tehran, Iran, in 2016, and the Ph.D. degree in Communications Engineering from the University of Oulu, Finland, in 2021. He was a Visiting Ph.D. Researcher at Linköping University, Sweden, in 2019, and a Post-Doctoral Researcher at the University of Oulu and the Technical University of Munich, Germany, in 2021 and 2022, respectively. He is currently a Post-Doctoral Researcher with the University of California at Santa Cruz, Santa Cruz, CA, USA. His research interests include information freshness, information theory, stochastic optimization, and reinforcement learning for wireless applications.

Markus Leinonen (Member, IEEE) received the B.Sc. (Tech.) and M.Sc. (Tech.) degrees in Electrical Engineering and the D.Sc. (Tech.) degree in Communications Engineering from the University of Oulu, Finland, in 2010, 2011, and 2018, respectively. He completed his master's thesis with the title "Power Minimization in Single-Sink Data Gathering Wireless Sensor Network via Distributed Source Coding" in 2011. In 2010, he joined the Centre for Wireless Communications, University of Oulu, as a Master's Thesis Student. He directly continued to work on his doctoral research resulting in the thesis title as "Distributed Compressed Data Gathering in Wireless Sensor Networks." He received the Academy of Finland Post-Doctoral Researcher position for his project "Time-Critical Communications for Internet of Things Systems" from 2021 to 2024. In 2013, he was a Guest Researcher with the Technical University of Munich, Germany. In 2020, he was a Visiting Post-Doctoral Researcher with the University of California San Diego (UCSD). He was conferred the title of Docent (Adjunct Professor) with the University of Oulu in February 2023. He has published over 70 journal and conference papers in the area of wireless communications. His main research interests were time-critical and sparsity-aware wireless communications, including optimization and analysis of information freshness in status update systems and design of sparse signal recovery methods for channel estimation as well as for user activity detection in massive machine-type communications. He passed away on February 4, 2023.

Marian Codreanu (S'02-M'07) received the M.Sc. degree from the University Politehnica of Bucharest, Romania, in 1998, and the Ph.D. degree from University of Oulu, Finland, in 2007. His thesis was awarded as the Best Doctoral Thesis within the area of all technical sciences in Finland in 2007. In 2008, he was a Visiting Postdoctoral Researcher at the University of Maryland, College Park, USA. In 2013, the Academy of Finland awarded him a five-year Academy Research Fellow position. In 2019, Dr. Codreanu received a Marie Skłodowska-Curie Individual Fellowship and joined the Linköping University, where he is currently an Associate Professor. Dr. Codreanu published over 150 journal and conference papers in the areas of wireless communications and networking, statistical signal processing, mathematical optimization, and information theory. His current research focus is on information freshness optimization, machine learning for wireless networking, and sparse signal processing.