Optimal Policies for Status Update Generation in an IoT Device With Heterogeneous Traffic

George Stamatakis, Nikolaos Pappas and Apostolos Traganitis

Tweet

LIU LINKÖPING UNIVERSITY

Optimal Policies for Status Update Generation in an IoT Device with Heterogeneous Traffic

George Stamatakis, Nikolaos Pappas *Member, IEEE*, and Apostolos Traganitis

Abstract—A large body of applications that involve monitoring, decision making, and forecasting require timely status updates for their efficient operation. Age of Information (AoI) is a newly proposed metric that effectively captures this requirement. Recent research on the subject has derived AoI optimal policies for the generation of status updates and AoI optimal packet queueing disciplines. Unlike previous research we focus on low-end devices that typically support monitoring applications in the context of the Internet of Things. We acknowledge that these devices host a diverse set of applications some of which are AoI sensitive while others are not. Furthermore, due to their limited computational resources they typically utilize a simple First-In First-Out (FIFO) queueing discipline. We consider the problem of optimally controlling the status update generation process for a system with a source-destination pair that communicates via a wireless link, whereby the source node is comprised of a FIFO queue and serves two applications, one that is AoI sensitive and one that is not. We formulate this problem as a dynamic programming problem and utilize the framework of Markov Decision Processes to derive the optimal policy for the generation of status update packets. Due to the lack of comparable methods in the literature we compare the derived optimal policies against baseline policies, such as the zero-wait policy. Results indicate that baseline policies fail to capture the complex system dynamics which determine the relationship between the frequency of status update generation and the resulting queueing delay and thus perform poorly. To the best of our knowledge, the derived optimal policy does not exhibit a simple structure; thus, we utilized the baseline policies, whose operation is intuitive, to gain insight into the inner workings of the optimal policy.

I. Introduction

Applications that offer monitoring, informed decision making, and forecasting services in cyber-physical systems, often rely on timely status updates [1], [2]. A large number of such applications has been developed in the context of Internet of Things with examples that include, but are not limited to, smart cities, smart factories and grids, smart agriculture, parking and traffic management, water management, e-Health, environment monitoring and education [1]. The proliferation of these applications is expected to have a profound impact on key sectors of economy, and this has spurred research on their particular operational requirements [3], [4]. A key result in the field was the realization [5] that the objective of timely status updating is not captured by metrics such as utilization and delay, which are typically used in network design and management.

G. Stamatakis and A. Traganitis are with the Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH) email:{gstam,tragani}@ics.forth.gr.

N. Pappas is with the Department of Science and Technology, Linköping University, Norrköping SE-60174, Sweden email:nikolaos.pappas@liu.se.

To alleviate this problem, a new metric, called Age of Information (AoI), was introduced in [5] to effectively capture the requirement for timely status updating. As an example, consider a scenario where a transmitter samples the state of a stochastic process $Z(k)$, that evolves over discrete time k, and sends status updates to a destination which is typically a monitor or a controller. Status updates are transmitted through a queue which introduces random delays in their delivery times. At any time k, if the freshest status update delivered at the destination was timestamped $T_M(k)$, then the AoI at the destination is,

$$\Delta(k) = k - T_M(k) \qquad (1)$$

Hence, the AoI is the amount of time elapsed since the moment that the freshest delivered update was generated.

In order to achieve a low AoI value at the destination one must ensure that the timestamp $T_M(k)$ is as close to the current time k as possible, i.e., that information at the destination is fresh. The authors in [5] showed that this depends on the interplay between two different time intervals. The first one is the time interval between the generation of successive status updates and the second one is the transmission delay between the sensor and the destination. The first one is within the control of the transmitter while the second one is generally a random variable whose distribution depends on the status update generation process. By shortening the time interval between the generation of successive status updates, one generates status updates more frequently and, due to congestion, transmission delays will increase. On the other hand, by increasing the interval between the generation of successive status updates, congestion decreases and consequently transmission delays will decrease. To minimize the expected value of AoI over time one has to find the optimal balance between the status update generation interval and the resulting transmission delays as presented in [5]. The role of AoI in IoT networks is considered in [6].

Further improvements, in terms of AoI, may be obtained by opting for a closed-loop solution for the generation of status updates instead of the open loop control scheme presented above, which is based on the selection of a single value for the interval between successive status update generation [7, Section 1.2, The Role and Value of Information]. A closed loop solution assumes the form of a policy for the generation of status updates, which, given the current state of the status update system, dictates whether a new status update should be generated or not. An example of such a policy is the zero-wait policy, first presented in [8] whereby a new status update is generated only when the queue is empty, i.e., the previous status update has been delivered successfully at the destination.

In this work, we study a system that is similar to the one presented above, with the main difference that the aforementioned queue is shared between a flow of status update packets and a flow of non-status update packets. Considering the complexity of monitoring applications in the context of IoT we expect that the deployed network equipment will definitely serve both AoI-sensitive and AoI-insensitive applications via the same queue. In our scenario generation of status update packets is fully controlled by the transmitter while non-status update packets are generated by an application which is beyond its control. All transmissions are subject to failure and upon a failed transmission attempt, the head-of-line packet will be retransmitted up to a maximum number of times after which it will be dropped. Furthermore, we assume that the AoI of the system is constrained to be less than a predefined threshold value. In case this constraint is not satisfied the source node will change temporarily the queue's default First In First Out (FIFO) service policy, and its transmission scheme, so that the delivery of a fresh status update to the destination is guaranteed and all outdated status update packets stored in the queue are dropped.

To the best of our knowledge, this is the first work to consider the design of an optimal controller for the generation of status updates for the wireless system under consideration. Optimality here is taken with respect to a cost function that is additive over time and depends on both the AoI of the system and the cost related to the use of the mechanism that guarantees a successful packet transmission. We formulate the problem at hand as a dynamic programming problem and utilize the framework of Markov Decision Processes (MDP) to derive optimal policies. Finally, we show by comparison that for a wide range of scenarios well known policies from the literature, such as the zero-wait policy, perform poorly for the system under consideration.

The remainder of the paper is organized as follows. In Section VII we present recent work related to the problem described above. In Section II we present the system model considered in this work. In Section III we formulate this problem as a dynamic programming problem. In Section IV we show that the dynamic program constitutes an MDP and present the algorithms we use to derive the AoI optimal policies. Finally, in Section VI we present numerical results for the evaluation of the derived policies. Our conclusions are in Section VIII.

II. SYSTEM MODEL

We consider a system that is comprised of a source node that transmits data to a destination node D through a wireless link. The source node consists of a sensor that generates data packets with status update information, an application that generates data packets with *non*-status update information, a finite queue, and a transmitter S. Subsequently, we will use the term *status updates* to refer to packets conveying status update information and the term *application packets* to refer to packets with non-status update information. The system model along with its state variables, which will be properly defined in Section III, is depicted in Fig. 1. We
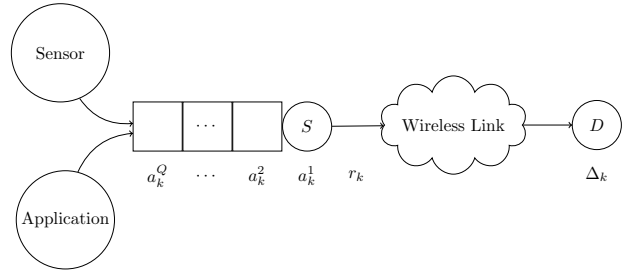

Fig. 1. System model.

assume that time is slotted and the transmission of a single packet occupies one time-slot. At the beginning of the k-th time-slot, S will commence the transmission of the head of line packet. The transmission may succeed with probability P_s or fail with probability $1 - P_s$, independently of the transmission outcomes in previous time-slots. We assume that all packet transmissions are acknowledged so that the success or failure of the transmission will be known to the source node by the end of the k-th time-slot. In the case of a failed transmission, a retransmission counter r_k will be incremented and the packet will be retransmitted during the next time-slot. The server will make up to $r_{max} - 1$ retransmission attempts and it will stop in case of a success. In case of r_{max} failed transmission attempts the packet will be dropped.

Within the duration of a time-slot, the application in Fig. 1 will generate a single application packet per time-slot with probability P_a, while the source node, which is in full control of the sensor, has to *decide* whether to generate a fresh status update or not. All packets generated within the duration of a time-slot will be enqueued unless the queue is full, in which case they will be dropped.

Finally, we assume that the source node must satisfy a hard constraint on AoI, i.e., Δ_k should always be less than a threshold value Δ_{max}. In case Δ_k becomes equal to Δ_{max}, the queue's service policy will change temporarily from its default FIFO operation so that the source node may be able to apply the following three actions: 1) The head of line packet is dropped. 2) All status update packets currently in the queue are dropped. 3) A fresh status update packet is sampled and transmitted with success probability 1 by the transmitter. We emphasize that this type of transmission is available to the source node only when AoI reaches the threshold value. This assumption is justified, from a technological point of view, by the tremendous attention that both academia and industry show in the support for Ultra Reliable Low Latency communications (URLLC) within the 5G framework. URLLC communication links are expected to have a successful packet delivery rate of up to $1 - 10^{-9}$ [9] which is extremely close to our assumed value of 1. Furthermore, since this level of high reliability will most probably be achieved through redundancy and utilization of excessive resources, such as transmission power, it is only natural to assume that it comes with a high cost. Further details regarding the conditions that determine the set of available decisions to the source node will be presented in Section III.

At the end of the k-th time-slot a cost g_k is induced to the system which is either equal to the AoI at the destination, or,

in case $\Delta_k = \Delta_{\max}$ equal to a fixed cost value. Our objective is to derive an optimal policy π^* that decides on the generation of a fresh status update at the beginning of each time-slot so that the expected value of the total discounted cost over an infinite number of time-slots is minimized, i.e.,

$$\pi^* = \arg\min_{\pi \in \Pi} \lim_{N \to \infty} \mathbb{E}\left\{ \sum_{k=0}^{N-1} \gamma^k g_k | x_0 \right\}, \qquad (2)$$

where x_0 represents the initial state of the system, which corresponds to an empty queue, $0 < \gamma^k < 1$ indicates that the importance of the induced costs reduces with time and expectation is taken with respect to the joint sequence of the system's random variables which represent the arrival of an application packet and the successful transmission of the head of line packet.

III. PROBLEM FORMULATION

In this section we formulate a dynamic programming problem for the system considered. We begin with the description of the state, control and random variable spaces and proceed with the system transition function, the state transition costs and the system cost function, which is additive over time. Finally, we give a short description about the optimal policy and its characteristics.

State Space Description: We utilize the AoI metric, denoted with Δ_k, $k = 0, 1, \ldots$ in Fig. 1, to characterize the freshness of status updates at destination D. AoI was defined in [5] as the time that has elapsed since the generation of the last status update that has been received by D. More specifically, let τ'_m denote the generation time-stamp of the m-th status update, τ'_m be the time-slot that the m-th status update arrived at destination D and M_k be the index of the last status update that D has received by the k-th time-slot, i.e., $M_k = \max\{m | \tau'_m \leq k\}$, then $\Delta_k = k - \tau_{M_k}$. This representation of Δ_k, for a centralized control model, has the drawback that one must keep timestamps, which are typically represented either as large integers or as vectors comprised of multiple sub-fields, as part of the description of the system's state which can be computationally inefficient.

We can derive a different expression for Δ_k by noting that when k equals the time-slot of the last status update arrival, i.e., $k = \tau'_{M_k}$, we have $\Delta_{\tau'_{M_k}} = \tau'_{M_k} - \tau_{M_k} + 1$. The increment by one in the latter equation is due to the slotted time assumption, whereby we account for the next time-slot in advance. We will subsequently use this definition for $\Delta_{\tau'_{M_k}}$. For the system of Fig. 1 the time interval, $\tau'_{M_k} - \tau_{M_k}$, is equal to the total time that the M_k-th status update spent waiting in queue and under service by the transmitter. We define $a_{M_k} = \tau'_{M_k} - \tau_{M_k}$ and thus $\Delta_{\tau'_{M_k}} = a_{M_k} + 1$. The AoI will increase by one unit for each time-slot that passes by beyond τ'_{M_k}, i.e., $\Delta_k = a_{M_k} + 1 + \delta_k$, $\tau'_{M_k} \leq k \leq \tau'_{M_{k+1}}$, where $\delta_k = k - \tau'_{M_k}$. For the source node to have knowledge of a_{M_k} for each status update delivered to D, we associate with each queue position a counter a_k^q, $q = 2, \ldots, Q$ (see Fig. 1) that holds the total time spent waiting in queue for the status update currently occupying the queue position. For the status update currently under service, counter a_k^1 holds the aggregate

time it has spent waiting in the queue and under service. The process of updating the values of a_k^q, $q = 1, 2, \ldots, Q$ as packets move from one queue position to the next will be presented later in this section. Furthermore, delay information for the application packets is irrelevant for the AoI of the system, thus whenever an application packet occupies the q-th position of the queue we assign counter a_k^q the special value of -1. An additional advantage of this assignment is a significant reduction in the size of the state space. We note here that counters a_k^q, $q = 1, 2, \ldots, Q$, corresponding to empty queue-slots will assume a value of zero. Finally, we assume that whenever Δ_k becomes equal to $\Delta_{\max} \in \mathbb{Z}^+$ the source node will preemptively drop the packet currently under service along with all queued status updates and it will transmit a fresh status update through an error free but expensive channel. As a consequence of this assumption, Δ_k will be bounded above by $\Delta_{\max}$, and the state space will be finite, as will become apparent subsequently.

The state of the system of Fig. 1, at the beginning of the k-th time slot, is determined by the values of Δ_k, r_k and a_k^q, $q = 1, 2, \ldots, Q$. We use column vector $x_k = [\Delta_k, r_k, a_k^1, \ldots, a_k^Q]^T$, where T is the transpose operator, to refer collectively to the state of the system and X to denote the set of all system states.

Control/Decision Space Description: Given its current state x_k the source node has to make a decision regarding the generation of a fresh status update, the dropping of the head of line packet and the preemptive transmission of a fresh status update. We denote the set of all possible decisions, termed controls, with,

$$U = \{(u^s, u^d, u^p) : u^s, u^d, u^p \in \{0, 1\}, \neg u^p \vee (u^p \wedge u^s \wedge u^d)\} \qquad (3)$$

where logical operators $\neg$, $\vee$ and $\wedge$ denote negation, inclusive disjunction and conjunction respectively, while u^s is a binary variable indicating whether the sensor should generate a status update, u^d is a binary variable indicating whether the head of line packet should be dropped at the end of the current time-slot in case of another failed transmission and u^p is a binary variable indicating whether all status updates within the source node should be dropped and a fresh status update should be preemptively transmitted to D. Predicate $\neg u^p \vee (u^p \wedge u^s \wedge u^d)$ will evaluate to true either for $u^p = 0$ ($\neg u^p = 1$) along with all combinations $(u^s, u^d) \in \{0, 1\}^2$, or for $(u^s, u^d, u^p) = (1, 1, 1)$. The latter control involves generating a fresh status update ($u^s = 1$), dropping the head of line packet ($u^d = 1$) as well as dropping all queued status updates and preemptively transmitting the fresh status update by using the costly, yet error free channel ($u^p = 1$).

At each system state x only a subset of the controls in U will be available to the source node. This subset is typically called the constraint control set and is denoted with $U(x) \subseteq U$. Table I categorizes the states based on their attributes and presents the corresponding constraint control sets. For notational convenience we drop the time index k since constraint control sets do not change over time.

System Random Variables: At the beginning of the $(k+1)$-th time-slot the system will make a transition to a new state x_{k+1} as a result of the selected control u_k and two random

TABLE I
SET OF AVAILABLE CONTROLS AT DIFFERENT STATE SUBSETS.

Subset of States	Constraint Control Set $U(x)$	Description
$\{x \in X : \Delta = \Delta_{\max}\}$	$\{(1, 1, 1)\}$	Generate a fresh status update, drop the head of line packet and all queued status updates, preemptively transmit a fresh status update through the expensive channel.
$\{x \in X : \Delta \neq \Delta_{\max}, r = r_{\max}, a^Q \neq 0\}$	$\{(0, 1, 0)\}$	The source node will not generate a fresh status update due to the full queue ($a^Q \neq 0$), the head of line packet will be dropped at the end of the current time-slot in case the $r_{\max}$-th retransmission fails.
$\{x \in X : \Delta \neq \Delta_{\max}, r = r_{\max}, a^Q = 0\}$	$\{(0, 1, 0), (1, 1, 0)\}$	The source node may or may not generate a fresh status update while the head of line packet will be dropped in case the $r_{\max}$-th retransmission fails.
$\{x \in X : \Delta \neq \Delta_{\max}, r \neq r_{\max}, a^Q \neq 0\}$	$\{(0, 0, 0)\}$	The sensor cannot generate a fresh status update due to the queue being full.
Otherwise	$\{(0, 0, 0), (1, 0, 0)\}$	The source may or may not generate a fresh status update.

events. The first one is the arrival of an application packet which is represented by the binary random variable W_k^a and the second one is the successful transmission of the head-of-line packet which is represented by the binary random variable W_k^s. As mentioned in section II, we assume that the application in Fig. 1 will generate a single packet per time-slot with probability P_a. Furthermore, the transmitter will deliver a packet successfully with probability P_s independently of the transmission outcome in any previous time-slot. The probability distributions of W_k^a and W_k^s are assumed to be independent of previous time-slots and identically distributed for all time-slots. We use the random vector $W_k = [W_k^a, W_k^s]^T$ to collectively refer to the random variables of the system.

State Transition Function: Given x_k, u_k and the values for W_k^s and W_k^a, which will be known to the source node by the end of the k-th time-slot, the system will make a transition to a new state $x_{k+1} = [\Delta_{k+1}, r_{k+1}, a_{k+1}^1, \ldots, a_{k+1}^Q]^T$. This transition is determined by the discrete-time system $x_{k+1} = f(x_k, u_k, W_k)$. Next we present the elements that comprise $f(\cdot)$. We begin with Δ_{k+1} which is given by the following expression,

$$\Delta_{k+1} = \begin{cases} 1, & \text{if } x_k \in X_{\Delta_{\max}} \\ \Delta_k + 1, & \text{if } x_k \notin X_{\Delta_{\max}} \text{ and } (W_k^s = 0 \text{ or } a_k^1 = -1) \\ a_k^1 + 1, & \text{if } x_k \notin X_{\Delta_{\max}} \text{ and } W_k^s = 1 \text{ and } a_k^1 \neq -1, \end{cases} \quad (4)$$

where $X_{\Delta_{\max}} = \{x \in X : \Delta = \Delta_{\max}\}$. Expression (4) shows that Δ_{k+1} will be set to one whenever AoI becomes equal to the maximum acceptable value of $\Delta_{\max}$. This is due to the transmission of a fresh status update through an error free channel. Furthermore, from (4) we see that Δ_k will be incremented by one in the cases of an unsuccessful packet transmission and that of a successful transmission of an application packet. Finally, in the case of a successful transmission of a status update, Δ_{k+1} will be set to $a_k^1 + 1$ which is equal to $(\tau'_{M_k} - \tau_{M_k}) + 1$.

Assuming that the queue in Fig. 1 can store at least one more packet besides the one currently under service, i.e., $Q > 1$, the

value of the retransmission counter r_k is updated as follows,

$$r_{k+1} = \begin{cases} 0, & \text{if } a_k^1 = 0 \text{ and } u_k^s = w_k^a = 0 \\ 0, & \text{if } (w_k^s = 1 \text{ or } u_k^d = 1) \text{ and } a_k^2 = u_k^s = w_k^a = 0 \\ 1, & \text{if } (u_k^d = 1 \text{ or } w_k^s = 1) \text{ and} \\ & \quad (a_k^2 \neq 0 \text{ or } u_k^s = 1 \text{ or } w_k^a = 1) \\ r_k + 1, & \text{if } a_k^1 \neq 0 \text{ and } w_k^s = 0 \text{ and } u_k^d = 0. \end{cases} \quad (5)$$

From (5) we see that r_{k+1} will be set to zero when there is no packet for the transmitter to transmit at the beginning of the $(k + 1)$-th time-slot. This may occur in two cases. Firstly, in case there wasn't a packet under service ($a_k^1 = 0$) and, additionally, there were no packet arrivals ($u_k^s = w_k^a = 0$) during the k-th time-slot. Secondly, in case the packet under service was either successfully transmitted or dropped ($w_k^s = 1$ or $u_k^d = 1$), the queue was empty ($a_k^2 = 0$ implies that all queue positions with $q \geq 2$ were also empty) and there were no packet arrivals ($a_k^2 = u_k^s = w_k^a = 0$) during the k-th time-slot.

On the other hand, r_{k+1} will be set to one if the packet being transmitted at the k-th time-slot departed from the source node either by being successfully transmitted or by being dropped and there exists another packet for the transmitter to transmit at the beginning of the $(k + 1)$-th time-slot. This scenario will occur either if the queue position with $q = 2$ was occupied by a packet during the k-th time-slot, i.e., $a_k^2 \neq 0$, or in case it was empty and a new packet arrived at the source node during the k-th time-slot. Finally, the value of r_{k+1} will be incremented by one if there exists a packet under service $a_k^1 \neq 0$ which is neither transmitted successfully, nor is it dropped by the source node.

Now, let $N_k^m \in \{0, \ldots, Q\}$ be zero, in case the queue is empty, and equal to the index value q, of the last queue position which is occupied by a packet,

$$N_k^m = \begin{cases} 0, & \text{if } \{q \in 1, \ldots, Q : a_k^q \neq 0\} \text{ is empty} \\ \max\{q \in 1, \ldots, Q : a_k^q \neq 0\}, & \text{otherwise.} \end{cases} \quad (6)$$

Furthermore, let N_k^p denote the number of application packets in queue at the k-th time-slot.

We can distinguish three groups of expressions related to updating the queue delay counter values a_{k+1}^q, $q = 1, \ldots, Q$. The first group of expressions applies to the case where $x_k \in X_{\Delta_{\max}}$ and is presented in Table II. The second group

a_{k+1}^q	Conditions for transition	Description
-1	$q = 1, \ldots, N_k^P$	The first N_k^P queue positions will be occupied exclusively by application packets since all status updates would have been dropped.
-1	$w_k^a = 1$ and $q = N_k^P + 1$	In the case of an application packet arrival, the new packet will be placed in the $(N_k^P + 1)$-th queue position, and $a_{k+1}^{N_k^P+1}$ will be set to -1.
0	$w_k^a = 0$ and $q = N_k^P + 1$	In the case of no application packet arrival, $a_{k+1}^{N_k^P+1}$ will be set to zero.
0	$q = N_k^P + 2, \ldots, Q$	For all remaining queue positions, up to the Q-th slot, a_{k+1}^q will be set to zero to indicate that they are empty.

of expressions applies when both $x_k \notin X_{\Delta_{\max}}$ and the packet that was transmitted at the k-th time-slot departed from the system either due to a successful transmission or because it was dropped by the transmitter ($u_k^d = 1$ or $w_k^s = 1$) and is presented in Table III. The third group of equations presented
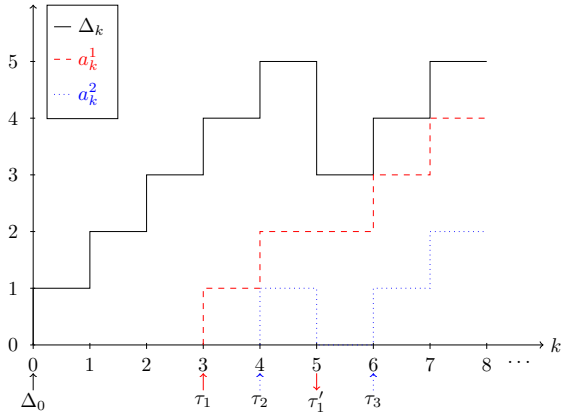

Fig. 2. Evolution of Δ_k, a_k^1, and a_k^2 over time for an example scenario with three status update arrivals at τ_1, τ_2, and τ_3 and one departure at τ_1' when Δ_0 is set to one.

in Table IV, applies in the case where both $x_k \notin X_{\Delta_{\max}}$ and the packet that was transmitted at the k-th time-slot did not depart from the source node which may occur if the packet was neither transmitted successfully nor dropped. Fig. 2 depicts the evolution of Δ_k, a_k^1 and a_k^2 over time for the following example scenario. At times $0, 1$ and 2 the queue is empty of packets and the AoI is incremented by 1 at the beginning of each new time-slot, starting from an initial value of 1. At time τ_1 a status-update enters the queue. By the end of the same time-slot the status update transmission has failed and both AoI and a_k^1 are incremented by 1. At time τ_2 a second status-update packet enters the queue and a_k^2 is set to 1. By the end of this latter time-slot, at time τ_1', the transmission of the head-of-line packet has been successful. This results in the second status-update to become the head-of-line packet, $\Delta_5 \leftarrow a_4^1 + 1$, $a_5^1 \leftarrow a_4^2 + 1$ and $a_5^2 \leftarrow 0$, where ($\leftarrow$) is the assignment operator. At time τ_3 a new status-update enters the queue. From time 6 up to 8 no successful packet transmission has occurred and all three counters are incremented by 1 at the beginning of each time-slot.

Transition cost and additive cost functions: With every state transition, according to control u_k, we associate a transition

cost $g(x_k, u_k, w_k)$ which is defined as,

$$g(x_k, u_k, w_k) = \begin{cases} G_{\Delta_{\max}}, & \text{if } x_k \in \Delta_{\max} \\ \Delta_{k+1}, & \text{otherwise,} \end{cases} \tag{7}$$

where w_k is the realization of random vector W_k at the k-th time-slot and $G_{\Delta_{\max}}$ is a virtual cost associated with the employment of the expensive channel whenever $x_k \in X_{\Delta_{\max}}$. The value of Δ_{k+1} is completely determined by values x_k, u_k and w_k, which are all known to the source node by the end of the k-th time-slot.

We are interested in minimizing the total cost accumulated over an infinite time horizon which is expressed as follows,

$$J_\pi(x_0) = \lim_{N \to \infty} \mathbb{E}_{\substack{W_k, \\ k=0,1,\ldots}} \left\{ \sum_{k=0}^{N-1} \gamma^k g(x_k, u_k, w_k) | x_0 \right\}, \tag{8}$$

where x_0 is the initial state of the system, expectation $\mathbb{E}\{\cdot\}$ is taken with respect to the joint probability distribution of random variables W_k, $k = 0, 1, \ldots$ and γ is a discount factor, i.e., $0 < \gamma < 1$, indicating that the importance of the induced cost decreases with time. Finally, π represents a policy, i.e., a sequence of functions $\pi = \{\mu_0, \mu_1, \ldots\}$, where each function μ_k maps states to controls for the k-th stage. For a policy π to belong to the set of all admissible policies Π, functions μ_k must satisfy the constraint that for time-slot k and state x_k controls are selected exclusively from the set $U(x_k)$.

In order to minimize (8), we must find an optimal policy π^* that applies the appropriate control at each state. This is a non-trivial problem since control decisions cannot be viewed in isolation. One must balance the desire for low cost in the short-term with the risk of incurring high costs in the long run. For example, a short sighted source node would avoid adding a fresh status update in a queue that already includes a status update. This is because the delay counter associated with the fresh status update will start incrementing immediately after its generation and this will have a negative impact on cost once the packet reaches the destination. However, this decision may lead to a queue filled with application packets and the AoI becoming equal to $\Delta_{\max}$, an event that will lead to the excessive penalty $G_{\Delta_{\max}}$.

IV. AGE OPTIMAL POLICIES

The dynamic program presented in section II is characterized by finite state, control, and probability spaces. Fur-

TABLE III
EXPRESSIONS TO UPDATE DELAY COUNTERS WHEN $x_k \notin X_{\Delta_{\max}}$ AND THE HEAD-OF-LINE PACKET DEPARTS.

a_{k+1}^q	Conditions for transition	Description
$a_k^{q+1} + 1$	$a_k^{q+1} > 0$ and $q = 1, \ldots, N_k^m - 1$	All packets in the queue will be shifted towards the head-of-line, and, accordingly, the values of a_k^q must be shifted to the right, i.e., $a_k^{q+1} \to a_k^q$. Especially for status updates, $a_k^{q+1} > 0$, the corresponding counters a_k^q will be increased by one to indicate that the packets will spend another time-slot in the system.
a_k^{q+1}	$a_k^{q+1} = -1$ and $q = 1, \ldots, N_k^m - 1$	Application packets will also be shifted to the right although the values of a_k^q will not be incremented by one.
-1	$u_k^s = 0$ and $w_k^a = 1$ and $q = N_k^m$	Addition of a newly arrived application packet at the first empty queue position.
1	$u_k^s = 1$ and $N_k^m < Q$ and $q = N_k^m$	Addition of a new status update at the first empty queue position. Status updates are generated only when the queue is not full ($N_k^m < Q$).
-1	$u_k^s = 1$ and $w_k^a = 1$ and $N_k^m \leq Q - 1$, and $q = N_k^m + 1$	Addition of both a new status update and a new application packet. We assume that status updates enter the queue first. There will always be enough queue slots for both packets given that status updates are generated only if there already exists an empty queue position and, in this case, we also have the departure of the head-of-line packet.
0	$u_k^s = 0$ and $w_k^a = 0$ and $q = N_k^m$	a_k^q counters will be set to 0 for all empty queue positions.

TABLE IV
EXPRESSIONS TO UPDATE DELAY COUNTERS WHEN THE HEAD-OF-LINE PACKET DOES NOT DEPART AND $x_k \notin X_{\Delta_{\max}}$.

a_{k+1}^q	Conditions for transition	Description
$a_k^q + 1$	$a_k^q > 0$ and $q = 1, \ldots, N_k^m$	Since no packet departed from the source node all packets in the queue will remain in the same queue position. Counters a_k^q of status updates will be increased by one to account for the additional time-slot they will spend in the source node.
a_k^q	$a_k^q = -1$ and $q = 1, \ldots, N_k^m$	Counters for application packets will not be incremented.
-1	$u_k^s = 0$ and $w_k^a = 1$ and $N_k^m \leq Q - 1$ and $q = N_k^m + 1$	An application packet arrival will be accommodated if there was at least one empty queue position during the k-th time-slot.
1	$u_k^s = 1$ and $N_k^m \leq Q - 1$ and $q = N_k^m + 1$	A fresh status update will enter the queue before a new application packet. Given that fresh status updates are generated only when there exists at least one empty queue position there will always be place for the fresh status update. Application packets that find the queue full will be dropped.
$-1,$	$u_k^s = 1$ and $w_k^a = 1$ and $N_k^m \leq Q - 2$ and $q = N_k^m + 2$	There will be enough queue positions to accommodate both a fresh status update and an application packet only if there were two empty queue slots during the k-th time-slot.
$0,$	$q = N_k^m + 2, \ldots, Q$	Counters a_{k+1}^q will be set to zero for all empty queue positions.

thermore, transitions between states depend on x_k, u_k, and w_k but not on their past values. Additionally, the probability distribution of the random variables is invariant over time. Finally, the cost associated with a state transition is bounded and the cost function $J(\cdot)$ is additive over time. Due to its structural properties the dynamic system at hand constitutes a Markov Decision Process (MDP) [10] which is described by its state transition probabilities,

$$p_{ij}(u) = P\{x_{k+1} = j | x_k = i, \ u_k = u\} = \sum_{(w_k^s, w_k^a) \in W_j} P\{W_k^s = w_k^s\} P\{W_k^a = w_k^a\} \quad (9)$$

where, $i, j \in X$, $u \in U(i)$, $(w_k^s, w_k^a) \in \{0,1\}^2$ and $W_j = \{(w_k^s, w_k^a) \in \{0,1\}^2 : j = f(i, u, [w_k^s, w_k^a]^T)\}$. From this point on we will utilize the MDP notation $p_{ij}(u)$ that presents the probability for the system to make a transition to state j given that the system is in state i and decision u was made.

For the MDP under consideration, given that $0 < \gamma < 1$, there exists an optimal stationary policy $\pi = \{\mu, \mu, \ldots\}$, i.e., a policy that applies the same control function μ at all stages [10, Sec. 2.3]. What is more, the control function μ will be independent of the initial state of the system and deterministic [10], i.e., each time the system is in state i, $\mu(i)$ applies the same control u. We will refer to a stationary policy $\pi = \{\mu, \mu, \ldots\}$ as stationary policy μ. Our objective is to find a stationary policy μ^*, from the set of all admissible stationary policies $\mathcal{M} \subseteq \Pi$, that minimizes the total cost in (8), i.e.,

$$\mu^* = \arg \min_{\mu \in \mathcal{M}} J_\mu(i), \qquad \text{for all } i \in S. \quad (10)$$

Let J^* be the total cost attained when the optimal policy μ^* is used, then, for the MDP at hand, J^* satisfies the Bellman equation,

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left[g(i, u, j) + \gamma J^*(j) \right], \text{ for all } i \in S, \quad (11)$$

where n is the cardinality of the state space. Equation (11) describes a system of n non-linear equations, the right hand side of which is a contraction, due to $\gamma < 1$, with a unique

fixed point located at $J^*(i)$. Due to the contraction property, one can derive both J^* and μ^* via iterative methods.

In this work we utilize the Optimistic Policy Iteration (OPI) algorithm [10]–[12] to approximate the optimal policy μ^* and the optimal infinite horizon cost J^* for the problem under consideration. Part of the OPI algorithm is the Approximate Policy Evaluation (APE) [10]–[12] algorithm, used to evaluate the infinite horizon cost for the sequence of policies produced by the OPI in the process of approximating μ^*. APE is presented in Algorithm 1. APE requires as input a stationary policy μ that maps each state $i \in X$ to a single control $u \in U(i)$ and returns an approximation of the infinite horizon cost J_μ for that policy. Optionally, if prior estimates for the values of J_μ exist, one may provide a J_μ in tabular form with preset cost values for each state $i \in X$, otherwise, APE will initialize arbitrarily the J_μ. APE will apply the transformation presented in the 5-th line of Algorithm 1 to each state and will produce J'_μ whose values are a closer estimate to the true values to the infinite horizon cost of policy μ. Formally, the values of J_μ will converge to the infinite horizon cost of policy μ only after an infinite number of repetitions. In practice, however, a finite number of repetitions is required for the algorithm to terminate and heuristically chosen values lead to an accurate calculation of J_μ as indicated by analysis and computational experience [10]. In Algorithm 1 repetitions stop when $\max_{i \in X}|J'_\mu(i) - J_\mu(i)|$ becomes smaller than a predefined threshold ϵ [11].

Algorithm 1 Approximate Policy Evaluation

Require: $\mu \in \mathcal{M}$
1: Initialize $J_\mu(i) \in \mathbb{R}$, $\forall i \in X$ arbitrarily if not given as input
2: Initialize ϵ to a small value
3: **repeat**
4:   **for all** $i \in X$ **do**
5:     $J'_\mu(i) \leftarrow \sum_{j=0}^n p_{ij}(\mu(i))[g(i, \mu(i), j) + \gamma J_\mu(j)]$
6:   **end for**
7:   $D \leftarrow \max_{i \in X}|J'_\mu(i) - J_\mu(i)|$
8:   $J_\mu \leftarrow J'_\mu$
9: **until** $D < \epsilon$
10: Return J_μ

The OPI procedure is presented in Algorithm 2. OPI begins with arbitrarily initialized values for the policy μ and its infinite horizon cost J. The values stored in tabular form will be updated iteratively and eventually will converge to μ^* and J^*. The major operation of the OPI algorithm, besides calling APE, is presented in Line 5 and is called the *policy improvement* step because its execution results in an improved policy μ', i.e., a policy that has a smaller infinite horizon cost compared to the previous policy μ. Subsequently, APE is called with the improved policy μ' and J as input. In this case J is provided as a better initial guess for the infinite horizon cost for policy μ' compared to an arbitrarily set table of values and as a result the call to APE will terminate faster. Upon termination APE will return an approximation for the infinite horizon cost of the improved policy μ' which

will be subsequently used to derive an improved policy by the policy improvement step. According to the Bellman's optimality principle [7], [12], unless policy μ_m is the optimal policy, the policy improvement step will always result in an improved policy, thus, Algorithm 2 will terminate in case a policy improvement step does not result in an improved policy, i.e., $\mu' = \mu$. Detailed analysis of the OPI and APE algorithms and their convergence properties can be found in [7], [10]–[12]. Finally, we note that the APE algorithm is also used to evaluate the infinite horizon cost for three heuristic policies that we will present in the next section.

Algorithm 2 Optimistic Policy Iteration

1: Initialize arbitrarily $J(i) \in \mathbb{R}$ and $\mu(i) \in U(i)$, $\forall i \in X$.
2: **repeat**
3:   policy_is_stable $\leftarrow$ true
4:   **for all** $i \in X$ **do**
5:     $\mu'(i) \leftarrow \arg\min_{u \in U(i)} [\sum_{j=0}^n p_{ij}(\mu(i))(g(i, \mu(i), j) + \gamma J(j))]$
6:     **if** $\mu'(i) \neq \mu(i)$ **then**
7:       policy_is_stable $\leftarrow$ false
8:     **end if**
9:   **end for**
10:   $J \leftarrow$ APE(μ', J)
11:   $\mu \leftarrow \mu'$
12: **until** policy_is_stable
13: Return $\mu \approx \mu^*$ and $J \approx J^*$

V. HEURISTIC POLICIES

To the best of our knowledge, the derived optimal policy could not be characterized by a simple threshold-based structure; thus, to provide insight into the inner workings of the optimal policy we introduce three heuristic policies whose intuitive operation allows us to reach conclusions about the way the optimal policy operates via comparative performance results.

The first heuristic policy is the *zero-wait* policy, denoted with μ_z, whereby the sensor will generate a status update either when the queue is empty or, mandatorily, when $x \in X_{\Delta_{\max}}$. In both of these cases the status update will spend zero time waiting in queue. The zero-wait policy presented here is a slight variation of the well-known zero-wait policy [5]. Recently, the authors in [13] verified that the zero-wait policy is suboptimal through experimental evaluation in two wireline scenarios. The results presented in the next section serve a similar purpose for the IoT scenario we consider. The second heuristic policy is the *max-sampling* rate policy, denoted with μ_m, whereby the sensor will generate a status update in all states that this is permitted, i.e., in all states x where $U(x)$ includes a control u with $u^s = 1$ the max-sampling policy will select that specific control. The third heuristic policy is the *never-sample* policy, denoted with μ_n, whereby the source node will never generate a status update unless this is mandatory, i.e., when $x \in X_{\Delta_{\max}}$. The main characteristic of the never-sample policy is the periodicity of Δ and transition cost values. More specifically, Δ will start with a value of one and

Description	Parameter	Value
Queue Size	Q	4
AoI Threshold	$\Delta_{\max}$	10
Max. Retransmission Number	$r_{\max}$	4
Expensive channel cost	$G_{\Delta_{\max}}$	100
Discount Factor	γ	0.99

will be incremented by one at each time-slot until, eventually, it becomes equal to $\Delta_{\max}$. The cost for these state transitions, $g(x_k, u_k, w_k)$, is imposed *at the end* of each time slot and its value is determined by the second branch of (7). Once the threshold $\Delta_{\max}$ is reached, a status update will be transmitted through the expensive channel, resulting in a transition cost of $G_{\Delta_{\max}}$, and Δ will become equal to one again. Fig. 3 presents Δ_k and $g(x_k, u_k, w_k)$ for the never-sample policy when $\Delta_{\max} = 10$ and $G_{\Delta_{\max}} = 20$. The total cost over each
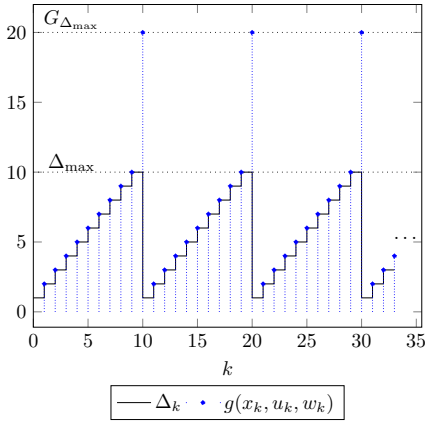

Fig. 3. AoI and transition cost for the never-sample policy when $\Delta_{\max} = 10$ and $G_{\Delta_{\max}} = 20$.

period is given by,

$$C_p = \sum_{c=2}^{\Delta_{\max}} c + G_{\Delta_{\max}} = \frac{\Delta_{\max}(\Delta_{\max} + 1)}{2} - 1 + G_{\Delta_{\max}}. \quad (12)$$

Never-sample policy exhibits the worst expected cost among all possible policies due to its complete lack of control over the status update process. In this work we also utilize its cost value as an indicator of how often the other three policies make use of the expensive channel.

VI. RESULTS

In this section we evaluate numerically the cost efficiency of the optimal policy μ^* for the system under consideration.

We consider the system of Fig. 1 configured with the set of parameter values presented in Table V. Let x_0 denote the initial state of the system, whereby the system is empty of packets and $\Delta_0 = 0$, then Fig. 4a presents the infinite horizon cost of all policies, i.e., J_{μ^*}, J_z, J_m, J_n for increasing values of the arrival probability P_a and a successful transmission probability of $P_s = 0.8$. In Fig. 4a and all subsequent figures we use $J(x_0)$ to refer to the cost associated with any policy. We note from Fig. 4a that when $P_a = 0$ or $P_a = 0.2$ the

zero-wait policy is nearly optimal, as has been already shown in the literature [8]. This indicates that for a low value of P_a the queue will often be empty of packets and a new status update will be generated frequently enough to avoid using the expensive channel. On the other hand, the max-sampling policy performs poorly because it constantly fills the queue with status updates that consequently suffer long waiting times. However, both the zero-wait and the max-sampling policies, as well as the optimal policy, achieve a much lower cost compared to the never-sample policy. This result indicates that, unlike the never-sample policy, these policies successfully avoid high cost state transitions and especially the frequent use of the expensive channel. This indication will become more concrete subsequently when we present results related to the frequency of usage of the expensive channel. When $P_a = 0.4$, both zero-wait and max-sampling policies perform much worse than the optimal policy, a result that exhibits the inability of these policies to capture the trade-off between the arrival rates for status and application packets. When P_a is equal to 0.6 or 0.8 the max-sampling policy is a better approach to the optimal policy than the zero-wait policy. This is due to the fact that application packets arrive at the queue with a high probability in each time-slot thus reducing the probability of an empty queue. As a result the zero-wait policy will generate status updates less frequently and, consequently, will resort to the use of the expensive channel more often. Finally, for $P_a = 1$, the optimal policy as well as all heuristic policies achieve similar costs. This indicates that the queue is always full with application packets and this causes the frequent use of the expensive channel by all policies in a way that resembles the operation of the never-sample policy. For this latter policy, we see from Fig. 4a that its performance does not change with P_a since it exclusively utilizes the expensive channel.

Figures 4b to 4d present $J(x_0)$ for decreasing values of the probability to successfully transmit, P_s. With the exception of the never-sample policy, Figures 4a to 4d depict that, for a specific value of P_a, a decrement in P_s results in an increased cost $J(x_0)$ for all policies. As expected, unsuccessful packet transmissions increase the waiting time of all packets in the queue and often result in packet drops, which cause even larger values of Δ_k, i.e., larger transition costs, and eventually lead to a more frequent use of the expensive channel. The frequent use of the expensive channel is also indicated by Fig. 4d where all policies achieve a cost close to that of the never-sample policy even for relatively small values of P_a. Summing it up, we have that the zero-wait policy approximates the optimal policy for relatively low values of P_a and P_s, while the max-sampling polixy is a better approximation to the optimal policy for larger values of P_a and P_s. Finally, there exists a range of values of P_a and P_s, as depicted in Figures 4a to 4d, for which both the zero-wait and max-sampling policies are poor approximations of the optimal policy. Unfortunately, the optimal policy does not accept a simple threshold-based formulation that could facilitate understanding of its inner working for this range of values of P_a and P_s.

To verify the assumption that the significant increase in $J(x_0)$ is due to the more frequent use of the expensive channel
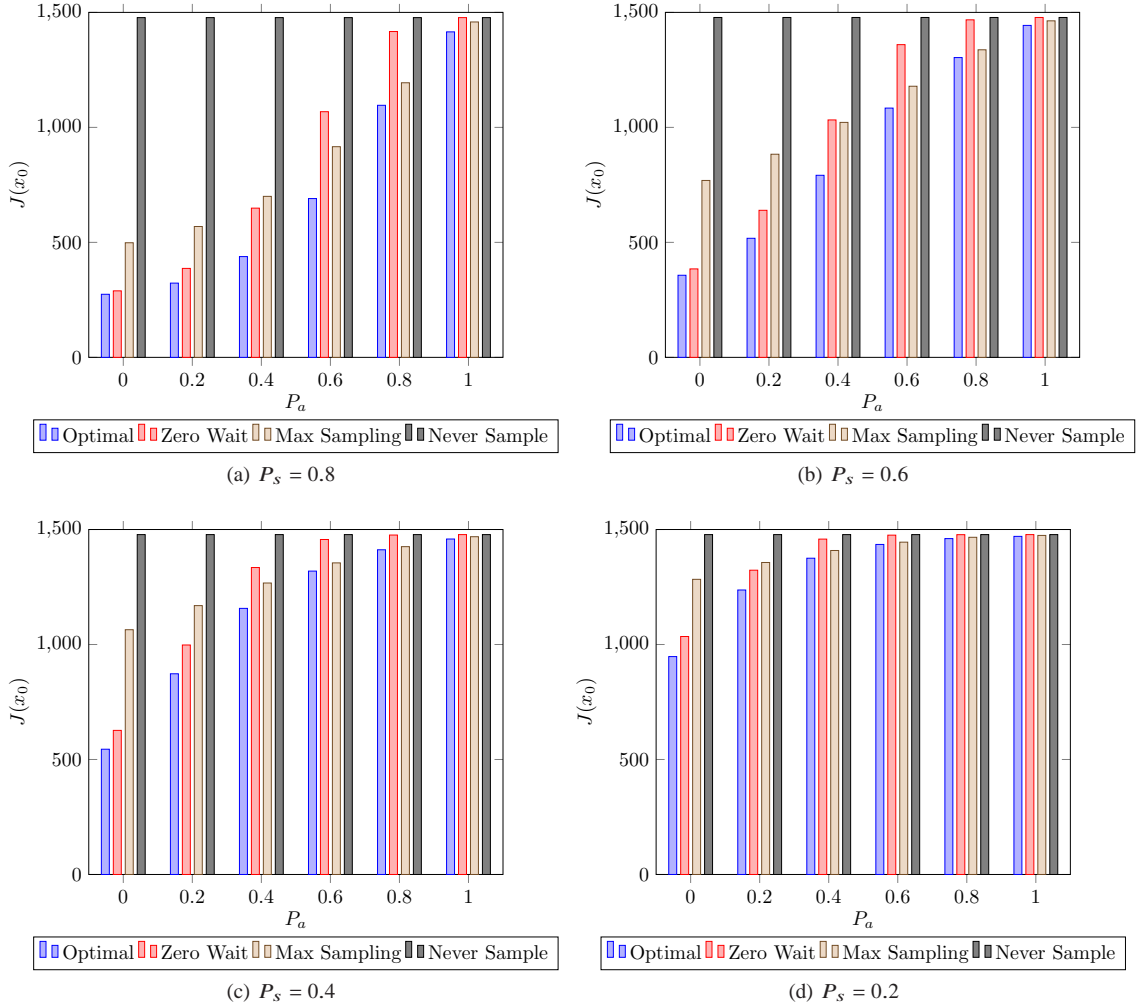
Fig. 4. Comparative plots for the optimal, zero-wait, max-sample and never-sample policies in terms of $J(x_0)$, i.e., the infinite horizon cost starting from an empty system with $\Delta_0 = 0$, for different values of the arrival probability for application packets P_a and the probability for a successful transmission P_s.

when P_a increases or when P_s decreases, we present in Fig. 5 the aggregate steady state probability of the system being in a state that will result in using the expensive channel, i.e., the aggregate steady state probability to be in a state $x \in X_{\Delta_{\max}}$. We note that given $p_{ij}(u)$ for the MDP, as defined in (9), and the three stationary policies μ^*, μ_z and μ_m one can derive the transition probability matrix P, for the resulting stochastic system as controlled by the provided policy. For example, the elements of P under the optimal policy are given by $P_{ij} = p_{ij}(\mu^*(i))$, for all $i, j \in X$. To derive a steady state probability vector we focus on the recurrent class of states that includes the initial state x_0. Now let P_r denote the transition probability matrix for this recurrent class of states, then we derive π, the steady state probability vector of P_r, as the normalized eigenvector of P_r that corresponds to P_r's eigenvalue λ which is equal to one [14]. Finally, the aggregate steady state probability of the system to be in a state that will result in using the expensive channel is given by, $\pi_e = \sum_{x \in X_{\Delta_{\max}}} \pi(x)$.

The zero-sample policy is not amenable to the analysis presented above due to the periodic character of the resulting Markov process. More specifically, the states of the resulting

Markov process can be grouped in a finite number of disjoint subsets so that all transitions from one subset lead to the next [15]. This is clearly shown in Fig. 3 where a transition from a state with AoI equal to Δ will always lead to a state with AoI equal to $\Delta + 1$ unless Δ equals $\Delta_{\max}$, in which case a transition will lead to a state with Δ equal to one. Therefore, by grouping states according to their AoI we can deduce the periodic character of the Markov process. However, one can see from Fig. 3 that the system will visit a state with AoI equal to $\Delta_{\max}$ once every $\Delta_{\max}$ transitions. From this observation we can derive that it will spend $1/\Delta_{\max}$ of its time in states where the expensive channel is used. For the scenarios in Fig. 5 π_e would be equal to 0.1. Figs. 5a-d exhibit that for large values of P_a or low values of P_s all policies behave the same way as the never-sample policy, i.e., they depend on the expensive channel. Finally, we note that although all policies have the same steady state probability to use the expensive channel when $P_a = 1$, as depicted in all cases of Fig. 5, they do not attain the same value of $J(x_0)$. This is due to the discount factor γ being strictly less than one, which results in early transition costs having a larger impact on $J(x_0)$ compared to the transition costs for larger k
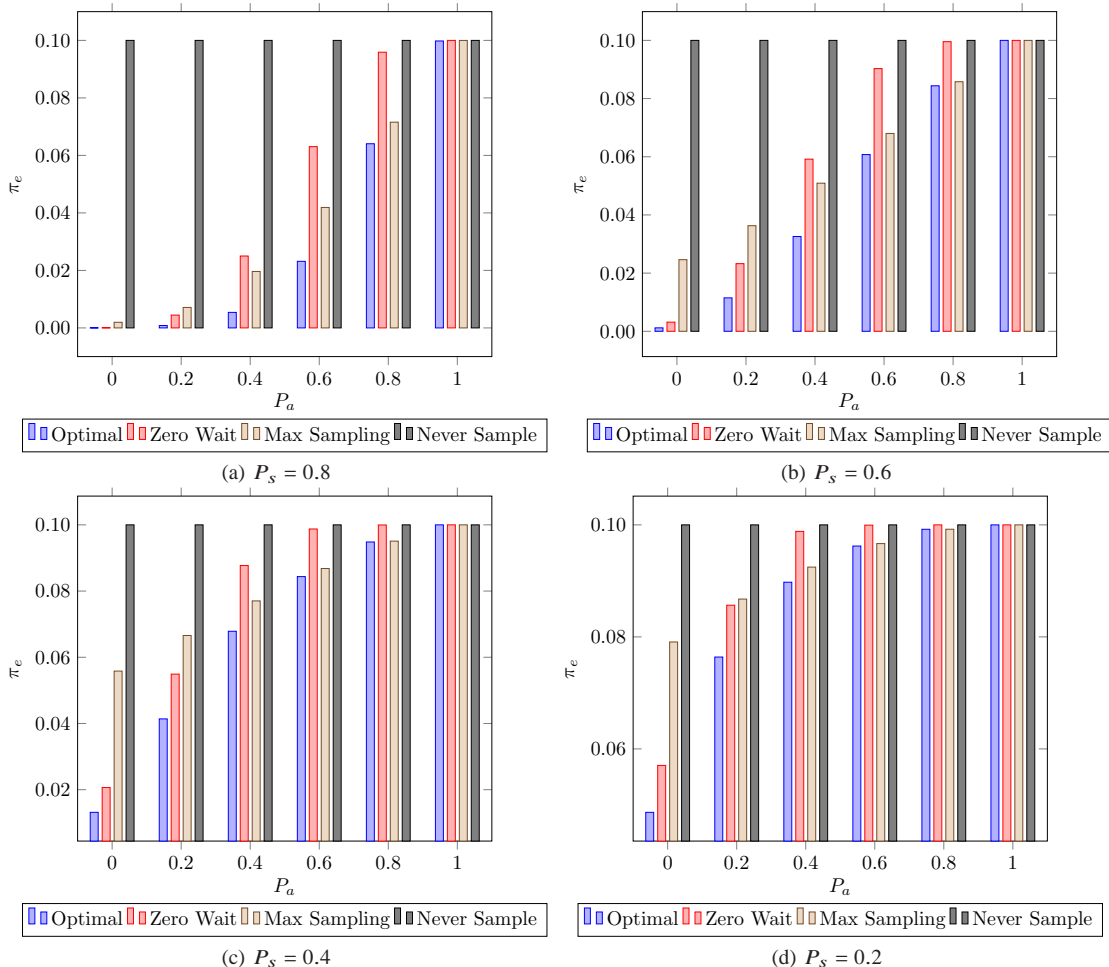
(a) $P_s = 0.8$

(b) $P_s = 0.6$

(c) $P_s = 0.4$

(d) $P_s = 0.2$

Fig. 5. Steady state probability to make use of the expensive channel, i.e., the sum of steady state probabilities for all states x where $\Delta = \Delta_{\max}$. The never-sample policy is not amenable to the same probabilistic analysis as the other three policies due to its periodic character, yet the system will spend 10% of its time using the expensive channel as is clearly shown in Fig. 3.

values. More specifically, during the early stages, whereby the system begins with an empty queue, the optimal, zero-wait and max-sampling policies make better decisions compared to the never-sample policy and thus achieve relatively lower values of $J(x_0)$.

Fig. 6 presents the impact of an increase of $G_{\Delta_{\max}}$ to the cost $J(x_0)$ when $P_s = 0.8$. More specifically, we set $G_{\Delta_{\max}} = 1000$ and note that the cost of the never-sample policy increases by an order of magnitude. Comparing the results in Fig. 6 with those in Fig. 4a one can identify that for low values of P_a cost $G_{\Delta_{\max}}$ has a small effect on the cost of all policies, with the exception of the never-sample policy. This is justified by the fact that these policies resort infrequently to the use of the expensive channel when P_a is low as has already be shown in Fig. 5. On the other hand, for larger values of P_a we observe a steep increment in cost which is due to the extensive use of the expensive channel.

Fig. 7 presents the effect of a progressive increase in the size of the queue on cost $J(x_0)$ for the optimal, zero-wait and max-sampling policies. More specifically, we increase the value of Q progressively from 4 to 8, while having $G_{\Delta_{\max}} = 1000$, $P_s = 0.8$ and $P_a = 0.4$. Comparing the results of Fig. 7 with
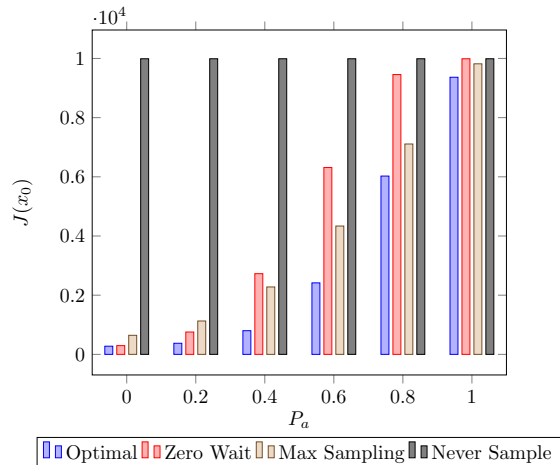

Fig. 6. Expected infinite horizon cost for all policies when the virtual cost $G_{\Delta_{\max}}$ associated with the use of the preemptive transmission mechanism is increased.

the corresponding scenario of Fig. 6 one can see that the cost of the max-sampling policy for $Q = 8$ has more than doubled compared to the scenario with $Q = 4$ due to the increased
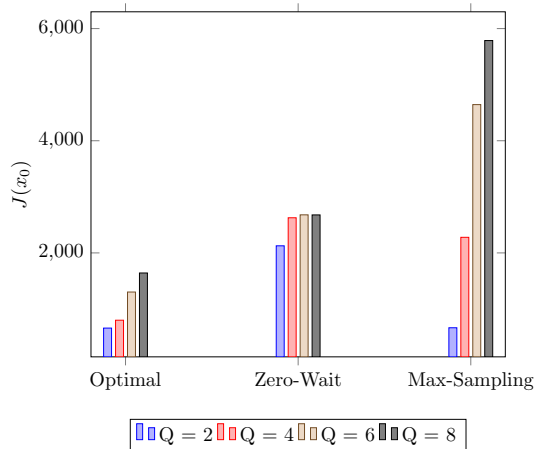
Fig. 7. Expected infinite horizon cost for the three policies when the size of the queue increases.

Fig. 8. Expected infinite horizon cost for three policies as the value of Δ_{max} increases.

waiting times caused by the larger number of status updates that enter the queue. Similarly, the cost for the optimal policy has also increased significantly because the state space for the increased queue size scenario involves many states with a high cost expectancy, i.e., states with a large number of application packets that would incur increased waiting times and more frequent use of the expensive channel. To avoid these states the controller has to make decisions that involve a more frequent generation of status updates so as to avoid using the expensive channel frequently. However, these decisions involve higher values of Δ_k compared to the scenario with the same setup but a smaller queue, i.e., higher transition costs. On the other hand, the cost for the zero-wait policy remains at relatively the same level as that for a smaller queue size since the zero-wait policy takes control actions only when the queue is empty. The rate with which the queue becomes empty depends on the values for P_s and P_a rather than the size of the queue, thus it was expected that the zero-wait policy would not be affected by an increment of the queue size.

Finally, Fig. 8 presents the evolution of cost $J(x_0)$ as Δ_{max} increases. More specifically, we increase the value of Δ_{max} progressively from 6 to 22, while keeping $G_{\Delta_{max}} = 1000$, $P_s = 0.8$, $P_a = 0.4$ and $Q = 4$. Comparing the results in Fig. 8 with the corresponding scenario of Fig. 6 we see that by relaxing the constraint imposed by Δ_{max} the cost for all three policies is significantly reduced. Furthermore, when Δ_{max} is within the range of values of 6 to 12 the cost obtained by using the optimal policy is significantly smaller compared to that achieved by the zero-wait and max-sampling policies. This is mainly because for low values of Δ_{max} cost $J(x_0)$ is dominated by the cost induced by the URLLC mechanism. The results indicate that the optimal policy succeeds in avoiding the use of URLLC mechanism contrary to the baseline policies. For larger values of Δ_{max} the optimal policy still performs better as indicated by the embedded figure within Fig. 8 and, since the cost of the URLLC mechanism is induced less often in this case, the results indicate that the optimal policy performs better in terms of AoI.
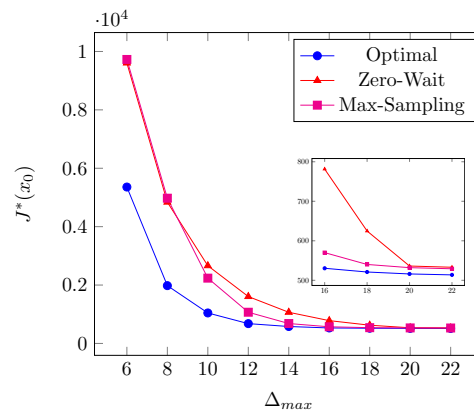
VII. RELATED WORK

In this section we present related work divided in two categories. The first category includes works that follow a queueing theoretic approach to the performance analysis and optimization of communication systems with respect to AoI and AoI related metrics while the second category includes works that focus on scheduling with respect to AoI.

In [16] the AoI in a general multi-class $M/G/1$ queueing system is studied. In addition, the exact peak-age-of-information (PAoI) expressions for both $M/G/1$ and $M/G/1/1$ systems are obtained. The work in [17] studied the status age of update packets transmitted through a network. The authors modeled a network as an $M/M/\infty$ model, and they derived the expression for the average AoI. The PAoI in an $M/M/1$ queueing system with packet delivery errors is considered in [18].

The work in [19] considers multiple independent sources that transmit status updates to a monitor through simple queues. A new simplified technique for evaluating the AoI in finite-state continuous-time queueing systems is derived. The technique is based on stochastic hybrid systems and makes AoI evaluation to be comparable in complexity to finding the stationary distribution of a finite-state Markov chain. In [20] the stationary distributions of AoI and the PAoI are considered. The authors derived explicit formulas for the Laplace-Stieltjes transforms of the stationary distributions of the AoI and PAoI in FCFS $M/GI/1$ and $GI/M/1$ queues. Yates in [21] employed stochastic hybrid systems to enable evaluation of all moments of the age as well as the moment generating function of the age in any network that can be described by a finite-state continuous-time Markov chain.

In [22], the authors introduce the metrics of Cost of Update Delay (CoUD) and Value of Information of Update (VoIU) in order to characterize the cost of having stale information at a remote destination and to capture the reduction of CoUD upon reception of an update respectively. The works in [23]–[25] consider setups with nodes with heterogeneous traffic and the interplay between AoI and throughput/delay is studied.

In [26], the average AoI for an $M/M/1/2$ queueing system with packet deadlines is studied.

The work in [27] studied the optimal control of status updates from a source to a remote monitor. The main differences

with our work is that their transmitter will generate a new status update only when the previously generated status update has reached the destination and its arrival has been acknowledged. No packet losses are considered and the authors of [27] completely disregard the existence of application traffic that shares the same queue with the status update packets. Their controller decides on the optimal time that the transmitter has to wait before transmitting a fresh status update. Finally, the optimization objective of their work takes the form of a constrained semi-Markov decision process where the infinite horizon undiscounted average of an AoI based penalty function is minimized.

Next we present works that focus on scheduling. The work in [28] considers a wireless broadcast network with a base station sending time-sensitive information to a number of nodes. A discrete-time decision problem is formulated to find a scheduling policy that minimizes the expected weighted sum of AoI for all nodes in the network. The authors in [29] consider a stream of status updates where each update is either of high priority or an ordinary one. Then, a transmission policy that treats updates depending on their priority is considered. The arrival processes of the two kinds of updates are modeled as independent Poisson processes while the service times are modeled as two exponentials. In [30] the optimal sampling problem for maximizing the freshness of received samples is formulated as an MDP in a system comprised of sampler, a wireless link and a receiver. The work in [31] considers a problem of sampling a Wiener process, the samples are forwarded to a remote estimator via a channel that consists of a queue with random delay. The estimator reconstructs a real-time estimate of the signal. The optimal sampling strategy that minimizes the mean square estimation error subject to a sampling frequency constraint is studied.

In [32] the problem of AoI minimization for single-hop flows in a wireless network, under interference constraints and a time varying channel is considered. A class of distributed scheduling policies, where a transmission is attempted over each link with a certain attempt probability is studied. AoI minimization for a network under general interference constraints and a time varying channel is studied in [33] and [34] with known and unknown channel statistics respectively. The work in [35] proposed a real-time algorithm for scheduling traffic with hard deadlines that provides guarantees on both throughput and AoI. The work in [36] considered a set of transmitters, where each transmitter contains a given number of status packets and all share a common channel. The problem of scheduling transmissions in order to minimize the overall AoI is considered. The authors in [37] study an AoI minimization problem, where multiple flows of update packets are sent over multiple servers to their destinations. The authors in [38], considered an alternative metric, the effective age, in order to achieve lower estimation error in a remote estimation problem. The problem they considered for developing an effective age is the remote estimation of a Markov source.

The work in [39] considers a sequential estimation and sensor scheduling problem in the presence of multiple communication channels. In [40], scheduling the transmission of status updates over an error-prone communication channel is studied

in order to minimize the average AoI at the destination under a constraint on the average number of transmissions at the source node. The works [41]–[43], introduce a deep reinforcement learning-based approach that can learn to minimize the AoI.

Similar to our work, the authors in [44] consider a single-hop Low-Power Wide-Area network where an IoT device transmits status updates to a base station following a Truncated Automatic Repeat reQuest (TARQ) protocol that supports preemption, i.e., the IoT device may conduct up to a given number of transmission attempts before dropping a status update or it will preemptively drop a packet in the case of a new status update arrival. However, the focus of their work is on the AoI-energy tradeoff and how it affects the average and peak AoI and the corresponding energy metrics, whereas we consider the problem of optimally controlling status update generation when the same queue is shared between two flows only one of which is AoI-sensitive.

In [45] the authors rely on the concept of AoI to offer an alternative approach to the problem of accuracy, decision-making, and lifespan of devices in an IoT network. The authors utilize correlated information in order to increase the accuracy of decisions and the lifespan for battery-powered devices and reduce the contention for the shared channel. More specifically the authors analyzed two correlated information sources and showed how the different spatial-temporal variation of the observed physical phenomenon influences the optimal use of status updates from the two sources.

In [46] the authors study a system whereby a source is equipped with a Last-In First Out queue and transmits status updates to the destination. Both generation and transmission of status updates are associated with power costs and a power constraint is set on the system's expected power consumption. At each decision stage the controller has to decide whether to spend energy in order to generate a fresh status update and whether to spend the necessary energy in order to transmit the currently available status update.

Additional references can be found in the survey [47].

VIII. Conclusion

In this work, we consider the problem of optimally controlling the generation of status updates for an IoT system that serves the data traffic of two applications, one that is AoI sensitive and one that is not through a single queue. We utilize the framework of MDPs to derive optimal status update generation policies for a wide range of configurations and compare them against two baseline policies, the zero-wait policy and the max-sampling policy. The comparative results clearly exhibit that both baseline policies are suboptimal because they disregard the effect on AoI of the non-status update packet arrivals and the unsuccessful transmissions. A limitation of the current work is that the modeling framework of MDPs is plagued with the curse of dimensionality which prohibits the efficient derivation of optimal policies for large scale systems.As part of a future work we will apply approximate dynamic programming techniques on the current problem with the intention to derive near optimal policies in a computationally efficient way.

REFERENCES

[1] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[3] D. Lund, C. MacGillivray, V. Turner, and M. Morales, "Worldwide and regional internet of things (iot) 2014–2020 forecast: A virtuous circle of proven value and demand," *International Data Corporation (IDC), Tech. Rep*, vol. 1, 2014.

[4] J. E. Siegel, S. Kumar, and S. E. Sarma, "The future internet of things: Secure, efficient, and model-based," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2386–2398, Aug. 2018.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *IEEE International Conference on Computer Communications (INFOCOM)*, Mar. 2012.

[6] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the role of age of information in the internet of things," *IEEE Communications Magazine*, vol. 57, no. 12, pp. 72–77, 2019.

[7] D. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I, 4th Edition*. Athena Scientific Belmont, MA, 2017, vol. 1.

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

[9] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, Oct 2018.

[10] D. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific Belmont, MA, 2012, vol. 2.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[12] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[13] B. Barakat, S. Keates, I. Wassell, and K. Arshad, "Is the zero-wait policy always optimum for information freshness (peak age) or throughput," *IEEE Communications Letters*, 2019.

[14] F. Gebali, *Analysis of computer networks*. Springer, 2015.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to probability*. Athena Scientific Belmont, MA, 2008, vol. 1.

[16] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *IEEE International Symposium on Information Theory (ISIT)*, June 2015.

[17] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1360–1374, Mar. 2016.

[18] K. Chen and L. Huang, "Age-of-information in the presence of error," in *IEEE International Symposium on Information Theory (ISIT)*, July 2016.

[19] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, 2018.

[20] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka, "The stationary distribution of the age of information in fcfs single-server queues," in *IEEE International Symposium on Information Theory (ISIT)*, 2017.

[21] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *ArXiv e-prints*, Jun. 2018.

[22] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[23] ——, "Age of information and throughput in a shared access network with heterogeneous traffic," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.

[24] Z. Chen, N. Pappas, E. Björnson, and E. G. Larsson, "Age of information in a multiple access channel with heterogeneous traffic and an energy harvesting node," in *IEEE INFOCOM Workshops*, April 2019.

[25] N. Pappas and M. Kountouris, "Delay violation probability and age of information interplay in the two-user multiple access channel," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, July 2019.

[26] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "On the age of information with packet deadlines," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6419–6428, Sept. 2018.

[27] 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 Transactions on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[28] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *54th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2016.

[29] E. Najm, R. Nasser, and E. Telatar, "Content based status updates," in *IEEE International Symposium on Information Theory (ISIT)*, June 2018.

[30] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *Journal of Communications and Networks*, 2019.

[31] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu, "Remote estimation of the wiener process over a channel with random delay," in *IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[32] R. Talak, S. Karaman, and E. Modiano, "Distributed scheduling algorithms for optimizing information freshness in wireless networks," in *2018 IEEE SPAWC*.

[33] ——, "Optimizing age of information in wireless networks with perfect channel state information," in *16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2018.

[34] R. Talak, I. Kadota, S. Karaman, and E. Modiano, "Scheduling policies for age minimization in wireless networks with unknown channel state," in *IEEE International Symposium on Information Theory (ISIT)*, June 2018.

[35] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, June 2018.

[36] Q. He, D. Yuan, and A. Ephremides, "Optimal link scheduling for age minimization in wireless systems," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5381–5394, July 2018.

[37] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-optimal updates of multiple information flows," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018.

[38] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Towards an effective age of information: Remote estimation of a markov source," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018.

[39] X. Gao, E. Akyol, and T. Basar, "On remote estimation with multiple communication channels," in *American Control Conference (ACC)*, July 2016.

[40] E. T. Ceran, D. Gunduz, and A. Gyorgy, "Average age of information with hybrid arq under a resource constraint," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2018.

[41] E. Sert, C. Sonmez, S. Baghaee, and E. Uysal-Biyikoglu, "Optimizing age of information on real-life tcp/ip connections through reinforcement learning," in *26th Signal Processing and Communications Applications Conference (SIU)*, May 2018.

[42] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks," in *PIMRC*. IEEE, 2018.

[43] Y.-P. Hsu, E. Modiano, and L. Duan, "Scheduling algorithms for minimizing age of information in wireless broadcast networks with random arrivals," *IEEE Transactions on Mobile Computing*, 2019.

[44] Y. Gu, H. Chen, Y. Zhou, Y. Li, and B. Vucetic, "Timely status update in internet of things monitoring systems: An age-energy tradeoff," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[45] J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Using correlated information to extend device lifetime," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[46] B. Zhou and W. Saad, "Optimal sampling and updating for minimizing age of information in the internet of things," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018.

[47] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: A new concept, metric, and tool," *Foundations and Trends in Networking*, vol. 12, no. 3, pp. 162–259, 2017. [Online]. Available: http://dx.doi.org/10.1561/1300000060