Freshness on Demand: Optimizing Age of Information for the Query Process

Josefine Holm*, Anders E. Kalør*, Federico Chiariotti*, Beatriz Soret*, Søren K. Jensen+, Torben B. Pedersen+, and Petar Popovski*
*Department of Electronic Systems, Aalborg University
Fredrik Bajers Vej 7C, 9220 Aalborg, Denmark, email: {jho,aek,fchi,bsa,petarp}@es.aau.dk
+Department of Computer Science, Aalborg University
Selma Lagerløfs Vej 300, 9220 Aalborg, Denmark, email: {skj,tbp}@cs.aau.dk

***Abstract*—Age of Information (AoI) has become an important concept in communications, as it allows system designers to measure the freshness of the information available to remote monitoring or control processes. However, its definition tacitly assumes that new information is used at any time, which is not always the case. Instead instants at which information is collected and used are dependent on a certain query process. We propose a model that accounts for the discrete time nature of many monitoring processes, by considering a *pull-based* communication model in which the freshness of information is only important when the receiver generates a query. We then define the Age of Information at Query (QAoI), a more general metric that fits the pull-based scenario, and show how its optimization can lead to very different choices from traditional push-based AoI optimization when using a Packet Erasure Channel (PEC).**

***Index Terms*—Age of Information, networked control systems**

I. Introduction

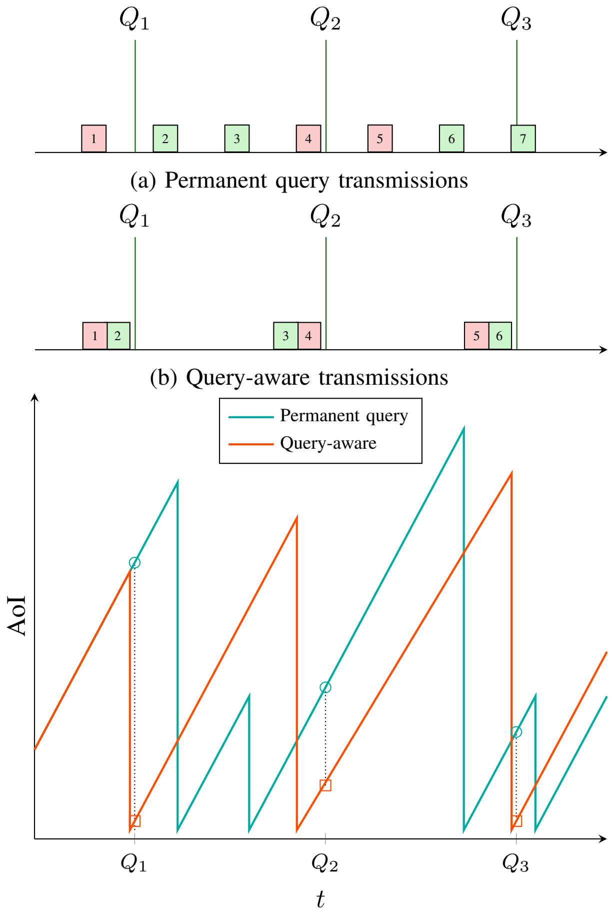
Over the past few years, the concept of information freshness has received a significant attention in relation to cyber-physical systems that rely on communication of various updates in real time. This has led to the introduction of *Age of Information (AoI)* [1] as a metric that reflects the freshness at the receiver with respect to the sender, and denotes the difference between the current time and the time when the most recently received update was generated at the sender.

The first works to deal with AoI considered simple queuing systems, deriving analytical formulas for information freshness [2]. Follow-up works addressed AoI in specific wireless scenarios with errors [3] and retransmissions [4], or basing their analysis on live experiments [5]. The addition of more sources in the queuing system leads to an interesting scheduling problem, which aims at finding the packet generation rate that minimizes the age for the whole system [6]. Optimizing the access method and senders' updating policies to minimize AoI in complex wireless communication systems has been proven to be an NP-hard problem, but heuristics can achieve near-optimal solutions [7] by having sources decide whether an update is valuable enough to be sent, i.e., whether it would significantly reduce the AoI [8]. The average AoI has been derived in slotted [9] and unslotted ALOHA [10], as well as in scheduled access [11], and the performance of scheduling policies has been combined with these access methods in [12].

However, the tacit assumption behind AoI, regardless of the system for which it is computed, has been that the receiver is interested in having fresh information *at any time*. In other words, this assumption works with *push-based* communication, in which a hypothetical application residing at the receiver has a *permanent query* to the updates that arrive at the receiver. The motivation for this paper starts by questioning this underlying assumption and generalizes the idea of AoI by considering the timing of the query process. This makes the communication between the sensor and receiver *pull-based*, where the query can guide the communication strategy for the sensor updates.

The impact of the query-driven, pull-based communication model becomes immediately obvious with the (over)simplified example in Fig. 1. The time is slotted and each packet, labeled $1, 2, \dots 7$, takes one slot. Each update is generated immediately prior to the transmission. The queries $Q_1, Q_2, Q_3, \dots$ arrive periodically, every 7-th slot. Furthermore, as an energy constraint, it is assumed that the sender can transmit on average one packet every 3 slots. Fig. 1a shows the case in which the sender is oblivious to the query arrival process and distributes the transmissions evenly in time. Another strategy could be, in each slot, to decide to transmit with probability $1/3$ or stay silent otherwise; the important point is that this decision is made independently from the query process. Fig. 1b shows the case in which communication is query-drive so the sender knows the query instants and optimizes the transmissions with respect to the timing of the query process, i.e., sends just before the query instants. In both cases the (red) packets $1, 4, 5$ are lost due to transmission errors. Fig. 1c shows that the query-driven strategy is more likely to provide updates that are fresh when a query arrives, although its average AoI is worse at the instants in which there is no query.

Despite the deceptively simple insight offered by the example from Fig. 1, the introduction of query-driven communication strategies does have a practical significance and introduces novel and interesting problems, as this paper shows. In fact, the assumption of a permanent query is relatively uncommon in the network control literature [13], which often uses periodic discrete time systems that poll the state of the monitored process at predefined intervals. Most network



(a) Permanent query transmissions

(b) Query-aware transmissions

(c) Age for the two systems

Fig. 1: Example of the difference between a system assuming a permanent query and one that is aware of the query arrival process. The same packets are lost (depicted in red) in both systems, and the markers indicate the age at the query arrival instants.

control systems are asynchronous, and use different sampling strategies that depend on the reliability of the connection and on the monitored process [14]. We define a *query arrival process* and consider the optimization of the communication process with respect to that arrival process. Furthermore, we define an Age of Information at Query (QAoI) metric which reflects the freshness in the instants when the receiver actually needs the data: having fresh data when the monitoring process is not asking for it does not provide any benefits to the system, as the information will not be used. Our model is also relevant for duty cycle-based applications, in which the sleeping pattern of the sensors are synchronized with the monitoring process.

This paper introduces models to analyze the difference in the communication strategies that should be used when the query arrival process is taken into account compared to the treatment of AoI in the context of a permanent query. In this initial work, we derive a Markov Decision Process (MDP) model for the problem with periodic queries and an erasure channel, and show that an optimization aimed at QAoI can significantly improve the perceived freshness with respect to classical models.

The remainder of the paper is organized as follows. We define the system model and the concept of QAoI in Sec. II, and we formalize it as an MDP in Sec. III. The setting and results of our simulations are described in Sec. IV, and Sec. V concludes the paper and presents some possible avenues of future work.

II. System model

We consider a scenario in which a wireless sensor generates updates at will and transmits them to an edge node over a wireless channel. The edge node receives queries from a server about the state of the sensor, e.g. as part of a monitoring or control process. The objective of this work is to maximize the freshness of the information used in the query responses while considering that the sensor is energy-constrained and needs to limit the number of transmissions to the edge node to prolong its lifetime.

A. Age of Information at Query

We consider a time-slotted system indexed by $t = 1, 2, \ldots$, and denote the time instants at which updates are successfully delivered to the edge node by $t_{u,1}, t_{u,2}, \ldots$. Following the common definition of AoI considered in the literature, e.g. [2], [6] we denote the AoI in time slot t by $\Delta(t)$, and define it as the difference between t and the time at which the last successfully received packet was generated:

$$\Delta(t) = t - \max_{i:t_{u,i} \leq t} t_{u,i}. \tag{1}$$

We will assume that $t_{u,1} = 0$ so that $\Delta(t)$ is well defined. An alternative, but equivalent definition can be obtained by introducing an indicator function $\psi(t)$, which is equal to 1 if a packet is successfully received in slot t and 0 otherwise:

$$\Delta(t) = \begin{cases} \Delta(t-1)+1 & \text{if } \psi(t) = 0; \\ 1 & \text{if } \psi(t) = 1, \end{cases} \tag{2}$$

where $\Delta(0) = 0$.

Most work considers the problem of minimizing the long-term average of $\Delta(t)$. However, this is only one possibility in real monitoring and control systems: discrete-time systems involve queries in which the monitoring process samples the available information. To capture such applications, we introduce the QAoI metric, which generalizes AoI by sampling $\Delta(t)$ according to an arbitrary querying process, thereby considering only the instants at which a query arrives. We denote the query arrival times at the edge node by $t_{q,1}, t_{q,2}, \ldots$, and define the overall objective as minimizing the long-term expected QAoI defined as

$$\tau_\infty = \lim_{t\to\infty} \frac{1}{t} \mathbb{E}\left[\sum_{i:t_{q,i} \leq t} \Delta(t_{q,i}) \right]. \tag{3}$$

Although the query process may in general follow any random process, in this initial paper we limit the focus to the case in which the exact query instants are known in advance to the edge node and the sensor. This is for instance the case when the queries are periodic, or if the server repeatedly announces its next query instant.

B. Models for Communication and Query Arrivals

We assume that each update has a fixed size and is transmitted over a Packet Erasure Channel (PEC) with erasure probability ϵ. For simplicity's sake, in the following we refer to the success probability $p_s = 1-\epsilon$. Packets are instantaneously acknowledged by the receiver, so the sensor knows if a packet was erased or correctly received.

To model the energy-constrained nature of the node, we use a *leaky bucket* model, as commonly done in the literature [15]: we consider a bucket of tokens, which is replenished by a process which can generate tokens independently at each step with probability μ_b. The node can only transmit a packet if there are tokens in the bucket, and each transmission consumes one token. This model can fit an energy gathering node, as well as a general power consumption constraint on a battery-powered node, which should limit its number of transmissions in order to prolong its lifetime.

In this work, we assume the simplest possible query arrival process, with periodic queries every T_q steps. We assume that the sensor and receiver are synchronized, i.e., the sensor knows when the next query will come. While simple, this assumption is often realistic, as discrete time monitoring processes are often designed with a constant time step.

The model can be easily extended to more complex query arrival processes, and the process statistics can even be learned implicitly as part of the optimal strategy, as long as it is consistent. If we follow the definitions from Sec. II-A, the strategies to minimize AoI and QAoI coincide in the memoryless case in which the query arrival process is Poisson or when the query arrival process is much faster than the sensor, i.e., when there is a query in each time slot.

III. MDP FORMULATION AND PROBLEM SOLUTION

In the following, we will model the two communication scenarios described in the next paragraph as MDPs, which we will then proceed to solve. An MDP is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a set of transition probabilities $p_a(s,s') = P(s_{t+1}=s'|a_t=a, s_t=s)$, and an instantaneous reward function $r(s,a,s')$, which represents the immediate reward when taking action a and transitioning from state s to state s'. The model can be used to represent two different systems: a Permanent Query (PQ) system, which minimizes the traditional AoI, and a Query Arrival Process Aware (QAPA) system, which minimizes the QAoI, only caring about the instants when a query arrives. These two systems can use the same state and action spaces, and only differ in the reward function that they use.

Decisions are made at every slot, as the sensor can either keep silent or send a packet. Consequently, the action space is $\mathcal{A} = \{0,1\}$. As the aim of the QAPA agent is to minimize the QAoI, the state should include the current age $\Delta(t)$, as well as the number of slots $\sigma(t)$ until the next query. Additionally, the agent should know the number of available tokens, $b(t)$, as it will influence its decision whether to transmit. If the number of tokens is 0, the sensor is blocked from transmitting until a token is generated. The state space can then be defined as $\mathcal{S} = \mathbb{N} \times \{0,\dots,T_q-1\} \times \mathbb{N}$, where $\mathbb{N}$ indicates the set of strictly positive integers. A state s_t is given by the tuple $(\Delta(t), \sigma(t), b(t))$. Each element in the state tuple evolves independently between time steps, so in the following we describe the state dynamics one by one.

The AoI increases by one between each slot unless the node decides to transmit and the packet is successfully received, with probability p_s, in which case the AoI is reduced to one in the subsequent slot. The non-zero transition probabilities are thus described by

$$\Pr(\Delta(t+1)=\delta|\Delta(t),a_t) = \begin{cases} a_t p_s & \text{if } \delta = 1; \\ 1-a_t p_s & \text{if } \delta = \Delta(t)+1; \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where a_t is the action at time t, which equals zero if the sensor is silent and one if it transmits. The time until the next query, denoted by $\sigma(t)$, is deterministic and independent of the action, and decreases by one until it reaches zero, at which point it is reset to T_q. Assuming that the first query happens at time $t = T_q$, the value of $\sigma(t)$ can be written

$$\sigma(t) = T_q - t \pmod{T_q}. \tag{5}$$

Finally, the number of tokens in the next slot depends on whether a new token is generated and whether the sensor transmits, in which case, it uses one token. The transition probability from $b(t)$ to $b(t+1)$ is:

$$p(b(t+1)=b+i|b(t),a_t) = \begin{cases} \mu_b & \text{if } i = 1-a_t; \\ 1-\mu_b & \text{if } i = -a_t; \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

We define two cost functions; one for the PQ system, which does not depend on the query instant and will be used as baseline, and one for the QAPA system, in which the cost is only considered when a query arrives. In the baseline PQ model, the cost is given by the AoI in any slot:

$$c_{\text{PQ}}(s_t,a_t,s_{t+1}) = \Delta(t+1). \tag{7}$$

However, in the QAPA system, the cost is the AoI when a query arrives:

$$c_{\text{QAPA}}(s_t,a_t,s_{t+1}) = \begin{cases} \Delta(t+1) & \text{if } \sigma(t+1) = 0; \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

In both cases, the objective is to find a policy π^* that minimizes the long-term cost. In this initial work, we limit ourselves to consider the discounted case, which benefits from strong convergence guarantees, and defer the case with undiscounted costs to future work. Specifically, we solve:

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(t)|\pi\right], \tag{9}$$

where $\lambda < 1$ is the discount factor.

We can now proceed to solve the MDP for the two systems we have defined using policy iteration, as described in [16, Ch. 4]. In order to apply the algorithm, we need to truncate the problem to a finite MDP. We do so by defining a maximum age $\Delta_{\max}$ and a token bucket size B: once the age or the number of tokens in the bucket reach the maximum, they cannot increase further. As long as the maximum values are sufficiently large, they are not reached during normal operation and this simplification does not affect the optimal policies or their performance.

The policy iteration algorithm has two steps: 1) policy evaluation and 2) policy improvement which are repeated until convergence. To solve the proposed problem we initialize the policy with zeros i.e. the policy where we never send any updates, and the value to be larger than we expect from a reasonable policy.

1) The policy is evaluated using

$$v_\pi(s) = \sum_{s'} p(s', c|s, a) \left(c + \lambda v_\pi(s')\right). \tag{10}$$

for all s, where s is the current state, s' is the new state, a is the action, and c is the cost from either (7) or (8).

2) The policy is improved. First, we compute

$$q_\pi(s, a) = \sum_{s'} p(s', c|s, a) \left(c + \lambda v_\pi(s')\right) \tag{11}$$

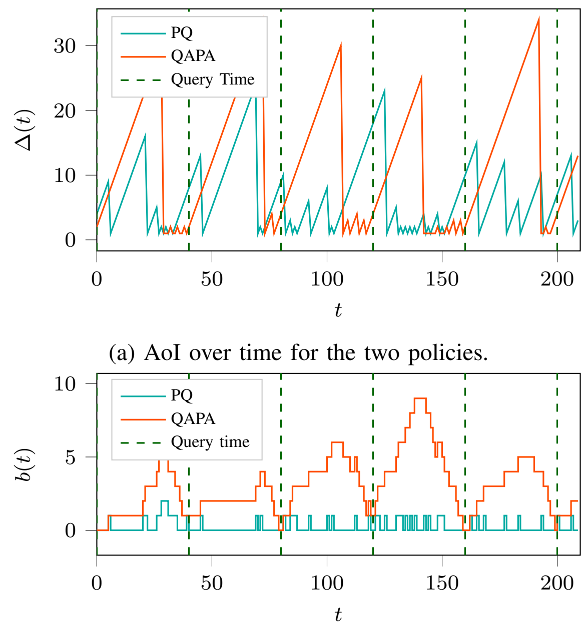
for all a. If $q_\pi(s, a) > v_\pi(s)$, we substitute a into the policy. This is repeated for all s.

Policy iteration is guaranteed to converge to the optimal policy [17] in finite-state MDPs with finite reward. As mentioned above, we truncated the age and token bucket size to make the MDP finite, so the conditions to use the algorithm apply.

IV. Numerical results

This section presents Monte Carlo evaluations of the policies obtained using the MDP described in Section III. Although, the methods in Section III can be applied to any query process, throughout the evaluation we will consider queries that occur periodically, at a fixed time interval T_q. Furthermore, we truncate the MDP at a maximum age of $\Delta_{\max} = 100T_q$ and a maximum token bucket size of $B = 10$, and we use a discount factor $\lambda = 0.75$. We use the term AoI to refer to the age at any time and QAoI for the age sampled at the query instants.

We start by exploring the temporal dynamics of the AoI process obtained using the PQ and the QAPA policies. Recall that PQ is optimized to achieve a low AoI independent of the query process, while QAPA minimizes the AoI at the query times, using cost functions (7) and (8), respectively. Fig. 2a shows the AoI for queries occurring periodically every $T_q = 40$-time slots as indicated by the vertical lines, a packet error probability of $\epsilon = 0.2$, and a token rate $\mu_b = 0.2$. It is seen that the PQ policy reduces the AoI approximately uniformly across time, while the QAPA policy consistently tries to reduce the AoI in the slots immediately prior to a query, so that the AoI is minimized when the query arrives. This is reflected in Fig. 2b, which shows that the QAPA policy accumulates energy when the next query is far in the future, unlike PQ. A consequence of this is that the QAPA policy generally has a slightly higher average AoI than the PQ policy, but the QAoI of the QAPA is significantly lower than that of the PQ policy.

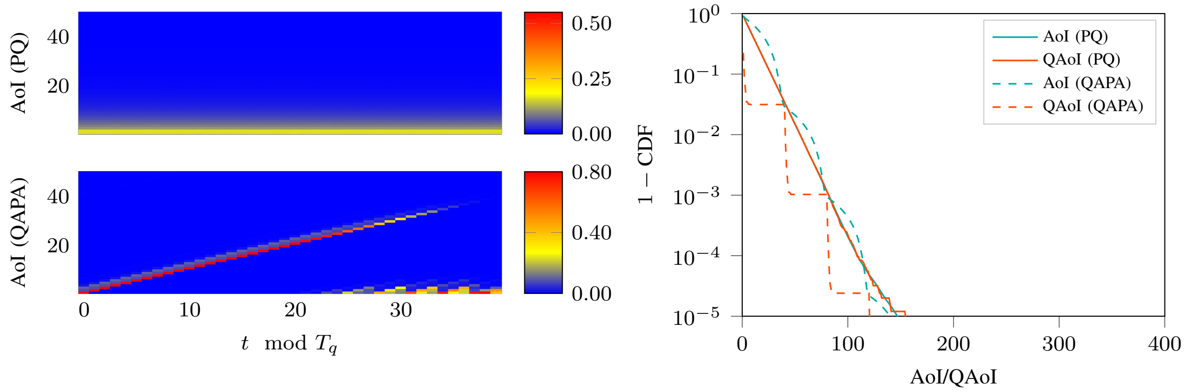


(a) AoI over time for the two policies.

(b) Available tokens over time for the two policies.

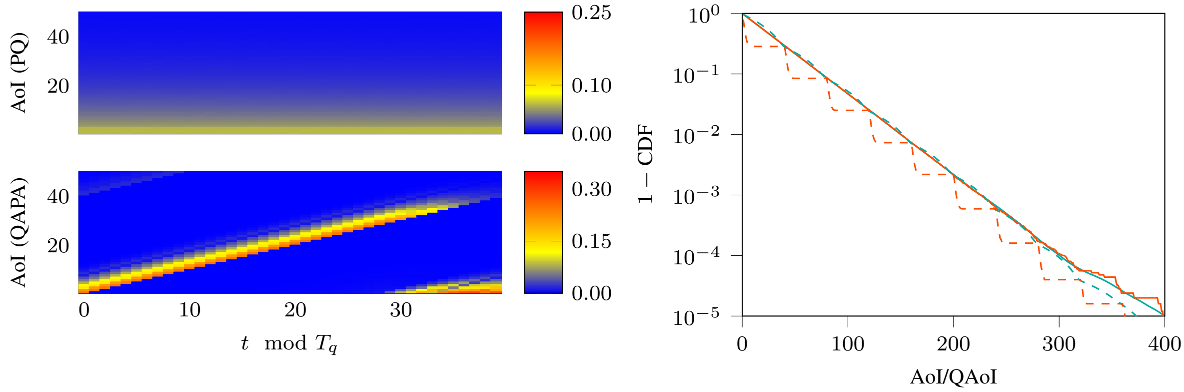
Fig. 2: AoI dynamics of the PQ and QAPA policies for $T_q = 40, \mu_b = 0.2, \epsilon = 0.2$. The PQ policy generally has a lower AoI, but the QAPA policy minimizes the AoI at the query instants.

The initial observations from Fig. 2 can be confirmed by the distribution of the AoI as a function of the time since the last query, as illustrated in Fig. 3. Figs. 3a and 3c show the probability mass function of the AoI conditioned $t \mod T_q$, while Figs. 3b and 3d show the Cumulative Distribution Function (CDF) of the overall AoI and QAoI. In the scenario with low error probability, $\epsilon = 0.2$, the AoI distribution of the PQ policy is uniform across time (upper plot in Fig. 3a), while the QAPA policy has an increasing age as time since the query passes, but a far lower age right before and at the query instant, $t \mod T_q = 0$ (lower plot in Fig. 3a). The resulting CDF in Fig. 3b reveals, as expected, that the AoI and the QAoI are equivalent for the PQ policy, as the distribution is the same at any time instant. However, for the QAPA policy, the QAoI is significantly lower than the AoI, while the AoI is often larger than the PQ policies. This is because the QAoI is only measured at the query instants, at which the age of the QAPA policy is minimized. Due to the energy constraint, this comes at the cost of a generally higher age, causing a higher AoI measured at each time instant. Finally, the staircase appearance in the CDF is because the queries happen periodically. If the queries came at variable (but known in advance) intervals, the



(a) PQ (upper) and QAPA (lower) density, $\epsilon = 0.2$.

(b) Complementary CDF, $\epsilon = 0.2$.



(c) PQ (upper) and QAPA (lower) density, $\epsilon = 0.7$.

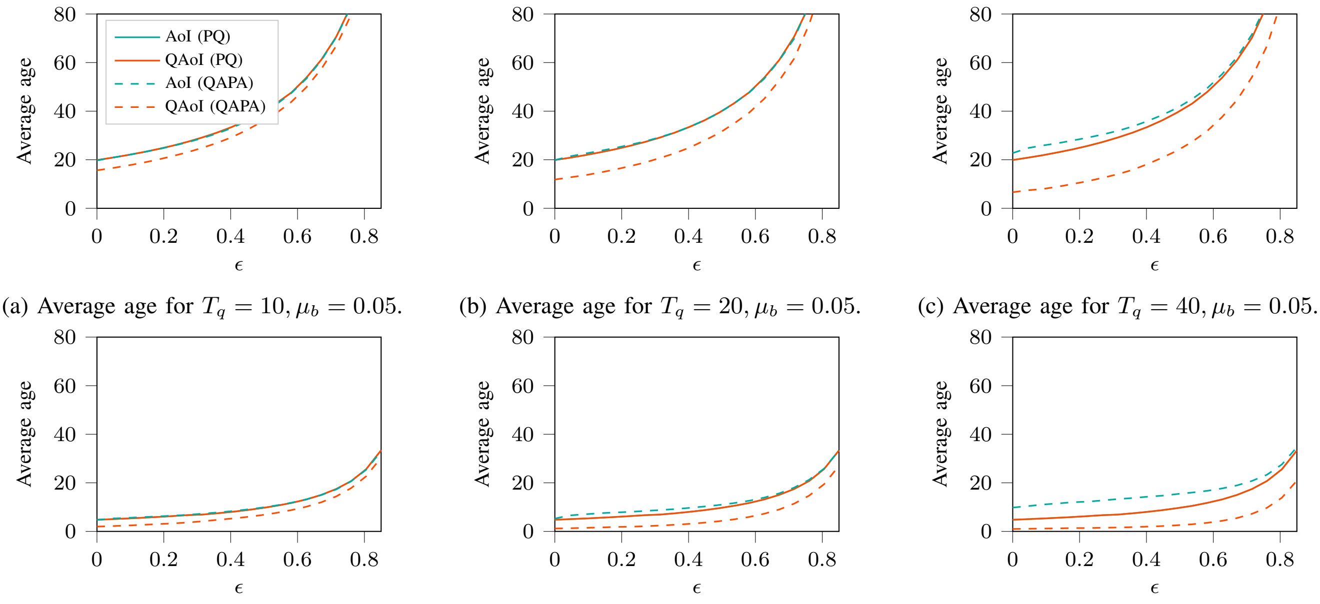
(d) Complementary CDF, $\epsilon = 0.7$.

Fig. 3: AoI distributions and CDFs for PQ and QAPA for $T_q = 40, \mu_b = 0.1$ and $\epsilon = \{0.2, 0.7\}$. (a), (c): AoI distribution for the PQ and QAPA conditioned on the time since the last query $t \mod T_q$, which corresponds to $T_q - \sigma(t)$, for $\epsilon = 0.2$ and $\epsilon = 0.7$, respectively. PQ achieves low AoI at all times, QAPA ensures that the AoI is low at the query instants, i.e. $t \mod T_q = 0$. (b), (d): Complementary CDF of the AoI and QAoI achieved by the two policies for $\epsilon = 0.2$ and $\epsilon = 0.7$. Generally, the QAPA policy has lower QAoI but higher AoI than the PQ policy.

CDF would be smoother, maintaining QAPA's performance advantage.

The same observations apply for the scenario with high error probability, $\epsilon = 0.7$, shown in Figs. 3c and 3d. Although the AoI and QAoI are higher due to the high packet error rate, the applied policies are similar. The gain that the QAPA policy achieves by clustering its transmissions close to the query instant is clearly reflected in Fig. 3c where, although there is a significant probability that the packet immediately prior to the query is lost, the AoI distribution at $t \mod T_q = 0$ is still concentrated close to one.

We close the section by studying how the average AoI and QAoI changes with the packet error probability ϵ for various choices of the parameters, shown in Fig. 4. For all cases, the QAPA policy achieves the lowest QAoI, while the PQ policy achieves the lowest AoI. When the query period, T_q, is low, the difference between AoI and QAoI is relatively small, as is the difference between the two policies. Intuitively, this is because the query instants, which are prioritized by the QAPA policy, are more frequent, making the two problems more similar. If we set $T_q = 1$, the two policies would coincide. As a result, awareness of the query arrival process becomes more important when queries are rare, i.e., when T_q is large: this is clear from the large gap between the average QAoI achieved by QAPA and by PQ in Fig. 4c and Fig. 4f. The upper row, Fig. 4a-4c, shows the results for $\mu_b = 0.05$, i.e., when a new token is generated on average every 20 time slots. When $T_q = 10$ (4a), the token period becomes a limiting factor, and both the AoI and QAoI are relatively high even for low values of ϵ. In particular, in the error-free case when $\epsilon = 0$, the average QAoI cannot be lower than $(1 + 11)/2 = 6$, which is achieved by transmitting an update prior to every second query. Interestingly, the impact of the energy limit becomes less significant for the QAPA policy as the time between queries increases: by saving up tokens until right before the query, this policy can significantly reduce the QAoI, at the cost of a higher AoI. On the other hand, the PQ policy does not benefit from this increase, as it is oblivious of the query arrival frequency. When tokens are generated faster, at rate $\mu_b = 0.2$, as shown in Fig. 4d-4f, the AoI and the QAoI are generally



(a) Average age for $T_q = 10, \mu_b = 0.05$.
(b) Average age for $T_q = 20, \mu_b = 0.05$.
(c) Average age for $T_q = 40, \mu_b = 0.05$.
(d) Average age for $T_q = 10, \mu_b = 0.2$.
(e) Average age for $T_q = 20, \mu_b = 0.2$.
(f) Average age for $T_q = 40, \mu_b = 0.2$.

Fig. 4: Average AoI and QAoI for the two systems for different values of T_q and μ_b.

lower, since more frequent transmissions are allowed.

V. Conclusions and Future Work

In this paper, we proposed a new metric for information freshness, which we dubbed QAoI: unlike standard AoI for push-based communication, this metric can be used for pull-based communication in which the monitoring process is not always listening, but sends queries when it is interested in the information. With the proposed model and subsequent MDP solution, we show the benefit of optimizing the transmission policy using the available knowledge on the query arrival process. Our results show that the standard PQ optimization, which minimizes AoI at any instant, can be very different from a QAPA policy that optimizes QAoI by concentrating its transmissions right before it expects a new query.

We are considering several avenues of future work, such as a formal derivation of the QAoI in simple queuing systems and the modeling of more complex query processes with stochastic timing, which would require the sensors to learn the nature of the query arrival process online.

Acknowledgment

This work has, in part, been supported the Danish Council for Independent Research (Grant No. 8022-00284B SEMIOTIC).

References

[1] A. Kosta, N. Pappas, V. Angelakis *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.

[2] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *INFOCOM*. IEEE, Mar. 2012, pp. 2731–2735.

[3] K. Chen and L. Huang, "Age-of-information in the presence of error," in *International Symposium on Information Theory (ISIT)*. IEEE, Jul. 2016, pp. 2579–2583.

[4] R. Devassy, G. Durisi, G. C. Ferrante, O. Simeone, and E. Uysal, "Reliable transmission of short packets through queues and noisy channels under latency and peak-age violation guarantees," *IEEE JSAC*, vol. 37, no. 4, pp. 721–734, Feb. 2019.

[5] H. B. Beytur, S. Baghaee, and E. Uysal, "Measuring age of information on real-life connections," in *27th Signal Processing and Communications Applications Conference (SIU)*. IEEE, Apr. 2019.

[6] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," *IEEE Transactions on Mobile Computing*, Dec. 2019.

[7] 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.

[8] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," in *International Symposium on Information Theory (ISIT)*. IEEE, Jun. 2015, pp. 3008–3012.

[9] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *International Symposium on Information Theory (ISIT)*. IEEE, Jun. 2017, pp. 331–335.

[10] R. D. Yates and S. K. Kaul, "Age of information in uncoordinated unslotted updating," *arXiv preprint arXiv:2002.02026*, Feb. 2020.

[11] R. Talak, S. Karaman, and E. Modiano, "Distributed scheduling algorithms for optimizing information freshness in wireless networks," in *19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, Jun. 2018.

[12] X. Chen, K. Gatsis, H. Hassani, and S. S. Bidokhti, "Age of information in random access channels," *arXiv preprint arXiv:1912.01473*, Dec. 2019.

[13] L. Zhang and D. Hristu-Varsakelis, "Communication and control codesign for networked control systems," *Automatica*, vol. 42, no. 6, pp. 953–958, Jun. 2006.

[14] D. Zhang, P. Shi, Q.-G. Wang, and L. Yu, "Analysis and synthesis of networked control systems: A survey of recent advances and challenges," *ISA transactions*, vol. 66, pp. 376–392, Jan. 2017.

[15] V. Raghunathan, S. Ganeriwal, M. Srivastava, and C. Schurgers, "Energy efficient wireless packet scheduling and fair queuing," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 1, pp. 3–23, Feb. 2004.

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

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