# Partially Stateful Server Selection for Minimal Age of Information Scheduling Over a Finite Horizon

Leonardo Badia
Dept. of Information Engineering
University of Padova, Italy
email: leonardo.badia@unipd.it

Andrea Munari
Institute of Communications and Navigation
German Aerospace Center (DLR), Weßling, Germany
email: andrea.munari@dlr.de

***Abstract*—We consider a source reporting real-time information content over a finite horizon so as to obtain minimal age of information (AoI). We assume that the information content requires a computationally heavy handling, as typical of tasks involving AI-augmented interpretation. As such, scheduling an information update requires a processing can be either performed locally, or offloaded to a remote mobile edge computing (MEC) server shared by other sources. The former option is subject to a certain failure rate, whereas the latter is always successful, but taking a longer time subject to how many other similar sources use the remote server. Inspired by the literature for MEC offloading, we consider a partially stateful approach, where the scheduling decision is made according to the system state (comprising the current AoI, the number of updates available, and the current congestion at the remote server), whereas the server selection follows a randomized-alpha policy. Through a dynamic programming approach, we find that the optimal choice of the local processing rate, although dependent on the characteristics of the remote server, is relatively robust to its variations. Not only does this justify our approach, but it also highlight a practical low-complexity approach to draw meaningful considerations on server sharing in multi-source updating systems.**

***Index Terms*—Age of Information; Internet of Things; Data acquisition; Mobile-edge computing; Resource sharing.**

## I. Introduction

Thanks to its seamless integration of multiple smart devices, the Internet of Things (IoT) is expected to revolutionize our interaction with technology at many levels. One of IoT's standout promises is its ability to deliver real-time applications, which reflects into efficient vehicular transportation, smart healthcare, industrial automation, and more [1]–[3]. This relies on rapid decision-making and system responsiveness, which translates in a requirement for data content freshness, usually represented through age of information (AoI) [4].

However, unlike simpler point-to-point remote sensing scenarios, whenever multiple interconnected devices share the same network infrastructure, and data generation at the network border becomes more and more intense, optimized resource allocation becomes essential at the heart of the requirement of low AoI, so as to allow split-second decision making and proactive interventions by the network control [5]–[7]. Especially, traditional computing architectures may be insufficient to provide the required responsiveness when the system status is extrapolated not from direct measurements but from a complex fusion of diverse data streams, possibly processed through AI algorithms [8]–[10].

In this case, an often invoked approach is to resort to offloading data processing to remote servers possessing higher computational capabilities [11]. However, neither a fully remote processing nor a fully local one can achieve the optimal performance (i.e., minimal AoI). Local processing, albeit often insufficient and worse than remote offloading if taken alone, has the advantage of physical proximity to where the data are generated and avoids long transmission delays. At the same time, offloading too many heavy tasks to a shared remote server may result into congestion that causes the performance to degrade, so that it becomes analogous to, if not worse than, that of a local processing – thus resulting in what is known as the Pigou-Knight-Downs paradox [12]. Therefore, it appears that the solution is to be sought in a careful and intelligent balance between the approaches.

Pushed by this motivation, in the present paper we delineate an analysis of a partially stateful resource allocation for AoI minimization over a finite time horizon in the context of computationally intensive status updates. The management of computation resources comprises two aspects, namely, the scheduling within the time horizon of the status updates, which can be further either processed locally or offloaded to a remote MEC server. The last decision follows a randomized-alpha stateless policy [13], which simplifies the analysis, yet considers the role of multple sources possibly congesting the server. Even though investigations of multi-source AoI exist, they belong to the track of queueing theory applications or infinite horizon stationary schedulers [14], [15], so they allow an optimization of the average AoI at steady-state only. Conversely, we remark that our analysis is the first to consider the role of multiple sources under non-stationary conditions.

This paper is organized as follows. Section II discusses the related literature. Section III introduces the system model, analyzed through dynamic programming in Section IV. Section V discusses numerical results. We conclude in Section VI.

## II. Related Work

The idea of quantifying freshness of information through AoI was popularized by seminal references [5], [16]. Over

This work was supported by the Italian PRIN2022PNRR project "DIGIT4CIRCLE," project code P2022788KK, and by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt." joint project 6G-RIC, project identification number: 16KISK022.

the last decade or so, AoI has become the *de facto* reference metric for real time applications, thereby leading to revisiting traditional analytical frameworks such as queueing theory [17].

Beyond the mere performance evaluation, another common approach is to set the minimization of AoI as the objective of real-time scheduling or medium access protocols [6], [14], [18]–[20]. This line of research is traditionally explored from the standpoint of a single central decision point that performs a centralized scheduling, often seen as an optimization problem and solved accordingly. However, while this straightforward methodology fits well point-to-point remote sensing scenarios, we argue that it is insufficient to characterize more modern applications in the IoT, such as autonomous driving, smart healthcare, and industrial IoT [1], [21]. Compared to the original scenario of AoI over a single communication link, two new elements arise in real-time IoT applications.

First of all, the expected widespread adoption of next generation technologies implies that the scenario requires an extension to a multi-agent paradigm [15]. Moreover, the evaluation of AoI is not expected to only relate to individual simple measurements, but rather may stem from computationally-intensive tasks, possibly AI-driven, for which more modern computing architectures based on mobile edge computing (MEC) are strongly pushed forward [11]. While MEC is a key enabler of AI applications to provide low latency and real-time data processing [8], for the overall framework of AoI minimization it inserts a new decision point in the pipeline, i.e., server selection. Similar to the aforementioned revival of queueing theory, we argue that this can lead to revisit classic studies about routing in Internet-like environment [22]. In agreement to this point, some recent contributions, such as [7], [23], explore MEC and server selection as related to AoI. In particular, [24] adapts queueing formulas for AoI from [4] and [15] to MEC processor sharing, and [10] performs a matching game for the resulting knapsack allocation of computationally intensive tasks to multi-server computing slices.

Our proposal, instead of being grounded in a queueing theory approach, explores the joint scheduling/server selection problem, as in [13], where, however, the performance metric was task completion probability within a deadline, not AoI. The authors of that paper argue that the network load must be carefully distributed among the available options of local and MEC servers, and this is simple to implement with a stateless online server selection policy that can be configured with near-optimal performance. Coordinating multiple sources in a fully stateful approach does not only bring limited improvement, but can be dangerously prone to errors. Overall, this idea is not peregrine, since in the end local processing has the main role of avoiding clogging the more powerful but congestion-prone MEC server, and we believe that this approach is conceptually very similar to random early dropping (RED) [25].

We remark that [9] also considers a dichotomy between local and remote processing for minimal AoI. However, there are numerous differences with the present paper, such as the analysis adopting a different methodology, but especially the focus is on a stationary policy over an infinite horizon, and a single source (thereby making either of remote and local processing always preferable depending on the context). Instead, the decision point of the present paper is in the timing of the updates within the horizon, which makes sense only when the latter is finite, the choice between local or remote processing being left to a stateless randomized-alpha policy as per [13]. Up to our knowledge we are the first to include the impact of multiple sources without resorting to queueing theory, thereby including, albeit in a simplified and tractable way, the role of server congestion in a non-stationary context.

## III. System Model

We consider a discrete (slotted) time, where a source of interest is sending status updates within a finite time horizon of $N$ slots. The value of AoI is also measured in slots and follows the standard definition [5], [6] that AoI in slot $n$, denoted as $\delta_n$, is the difference between the current slot index and that of the last successful update, denoted as $\sigma_n$. Formally,

$$\delta_n = n - \sigma_n\,, \qquad \text{where } \sigma_n = \max_{t_k \le n}(0, \{t_k\}_k) \tag{1}$$

with the $t_k$s being the slots where an update of the information content is performed.[1]

In principle, it is immediate to relax the assumption of a finite $N$ and allow for a stationary analysis over an infinite horizon [14], replacing the dynamic programming approach that we adopt in the following, e.g., with value iteration (VI) [20]. Still, we believe that for actual real-time applications it makes more sense to consider a finite horizon that represents their task-oriented nature, as well as better justifies the constraint on the activity rate in terms of duty cycle [26].

Indeed, we assume that within the $N$ slots, the source is allowed to perform only $M$ updates (or status reporting). This can be related to several reasons, such as energy consumption, or even regulation constraints depending on the technology (e.g., LoRa systems are required to keep their duty cycle below $1\%$) [27]. We also remark that the finiteness of the horizon enforces such a constraint in a stronger way, since there cannot be more than $M$ updates in any case, whereas a stationary policy over an infinite time-horizon actually may violate this constraint locally, as long as the long-term average of the duty cycle is below a certain limit.

Moreover, we expand the approach usually taken in the AoI literature, where status reporting is an atomic activity, by considering it as the result of a complex, possibly AI-aided, processing. As such, it can be (i) subject to failures and (ii) offloaded to a more powerful remote server. The latter element is typical of MEC offloading, possibly involving a slice of a network computing facility [24]. However, this means that the actual processing rate of the remote MEC server depends on how many sources use it. In the following, we will thereby consider that $U$ competing sources are possibly using the same remote server. Moreover, these competing sources have the same activity rate as the source of interest, i.e., they perform $M$ updates within $N$ slots, albeit they are fully asynchronous

[1] Conventionally, we initialize AoI as $\delta_0 = 0$.

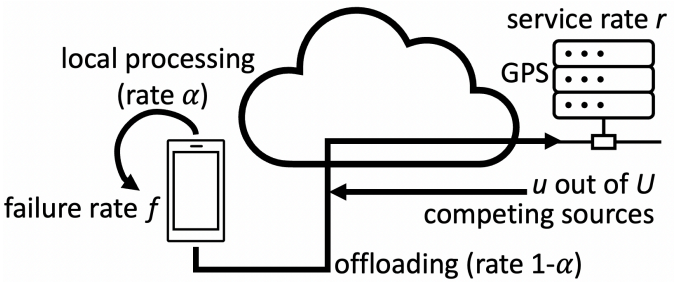


Fig. 1. Model of task offloading (dual local/remote processing).

with each other. This implies that we treat the status updates that other sources possibly offload on the remote server arrive there following independent stateless arrival processes.

When the source of interest schedules a status update, it follows server selection as per Fig. 1, i.e., the update can be performed as either a local task, which implies that it is completed immediately within the same slot, yet it is subject to failures with probability $f > 0$, or a remotely offloaded task, which may require a longer time but is guaranteed to succeed, since we assume that the offloaded request keeps being forwarded/processed until success. Thus, we take the termination of an offloaded task in the current slot as Bernoulli distributed with probability $\rho$, implying that a status update can be performed immediately in the slot it is requested with probability $\rho$, whereas with probability $1-\rho$ it takes one more slot, and the check is repeated again in the following slot. This causes the time elapsed on the remote server until success to be quasi-geometrically distributed, as the value of $\rho$ can change between slots. Indeed, we take $\rho$ as dependent on the load experienced by the MEC server, such that, if $\omega$ sources are currently offloading tasks to it, then $\rho = r/\omega$, with a constant $r$, representing the aggregate completion rate of the updates currently offloaded, which is a characteristic parameter of the remote server. When multiple updates are offloaded, they share the processing capability of the MEC server fairly and independently [2].

We remark that this quasi-geometric distribution is just chosen for the sake of elegance in the analysis, but nothing prevents a replacement with a more complex distribution, such as considering a fixed delay to reach the remote server. Moreover, to isolate the scheduling policy related to the source of interest from the other competing sources, whose number is denoted as $u$, $0 \leq u \leq U$, we consider two separate cases: (i) the source of interest is one of those currently offloaded on the remote server, in which case $\omega = u+1$; or (ii) the source of interest is not on the remote server, thus $\omega = u$.

The choice between local or remote processing for every status update is according to the randomized-alpha policy defined in [13], which implies it to be local or remote with independent identically distributed (iid) probability $\alpha$ and $1-\alpha$, respectively, with $0 \leq \alpha \leq 1$. We refer the interested reader to that reference as supporting our choice, as it is shown to achieve near-optimal results, often analogous to more complex policies, as long as $\alpha$ is properly set. We actually consider all sources, not just the one of interest, as following this approach. Since they are asynchronous, this results in a fraction $(1-\alpha)$ of their updates being offloaded. Finally, we also remark that the randomized-alpha policy is introduced in [13] as a fully stateless policy, whereas in our case the status updates are processed based on a stateful scheduling, this is why we denote our approach as being *partially stateful*.

## IV. Analysis

We aim at deriving a scheduling strategy of $M$ status updates for the source of interest over a finite time horizon of $N$ slots so as to minimize the average AoI. The problem is inherently non-stationary, first of all because of the finiteness of $N$ makes the choice on whether to perform a status update dependent on the current AoI, the number of remaining opportunities, as well as the current time index $n$ [19]. In addition, a further aspect concerns the service capability of the remote server, which is assumed to depend on its current congestion level as per a generalized processor sharing [28].

Thus, the first element to derive is the evolution of the number $u$ of competing sources currently on the remote MEC server. This follows either of two Markov chains, depending on whether the source of interest is offloaded or not. According to the assumptions made, the remote MEC server sees an arrival rate of updating tasks equal to $(1-\alpha)(M+1)/N$ for each of the $U$ competing sources, and provides a service rate equal to $r/\omega$ for each of the sources currently being served [2]. We remark that, despite the Markovian characteristics of arrivals and services, multiple arrivals and/or services are possible in the same time slot as its duration is not infinitesimal.

The transition between $u$ competing sources in the current time slot and $v$ sources in the next one has probability $p_{uv}$ as reported in (2) on top of the next page. The equation is promptly explained by considering, at first, the case of $v-u$ arrivals when $v \geq u$, and $u-v$ departures when $u > v$. But, in addition to those, there may be $k$ extra simultaneous arrivals and departures (e.g., $v-u+k$ arrivals and $k$ departures when $v \geq u$), with $k$ being obviously lower bounded by 0 but upper bounded by the number of other available sources not already offloaded, which justifies the upper limit of the summations.

We remark that (2) contains the variable $\omega$ that, depending on whether the source of interest is currently offloaded or not, is equal to either $u+1$ or $u$, respectively. Thus, we get a slightly different expression for the transition probability $p_{uv}$, depending on this. These values can be collected into matrices $\mathbf{P}_0$ or $\mathbf{P}_1$, both being equal to $\{p_{uv}\}_{uv}$ but considering the source of interest as not offloaded or offloaded to the remote server, respectively. These transition matrices correspond with two different discrete time Markov chains, which alternate in describing the evolution of the number of competing sources depending on the current state of the source of interest, i.e., according to $\mathbf{P}_0$ or $\mathbf{P}_1$.

Now, the optimal allocation of updating tasks over a finite horizon can be performed with our online "partially stateful" scheduling, where we remark that the policy itself is actually

$$p_{uv} = \begin{cases} \sum_{k=0}^{\min(v,U-u)} \binom{u}{u-v+k} \left(\frac{r}{\omega}\right)^{u-v+k} \left(1-\frac{r}{\omega}\right)^{v-k} \times \binom{U-u}{k} \left((1-\alpha)\frac{M}{N}\right)^{k} \left(1-(1-\alpha)\frac{M}{N}\right)^{U-u-k} & \text{if } v < u \\ \sum_{k=0}^{\min(u,U-v)} \binom{u}{k} \left(\frac{r}{\omega}\right)^{k} \left(1-\frac{r}{\omega}\right)^{u-k} \times \binom{U-u}{v-u+k} \left((1-\alpha)\frac{M}{N}\right)^{v-u+k} \left(1-(1-\alpha)\frac{M}{N}\right)^{U-v-k} & \text{if } v \geq u \end{cases} \quad (2)$$

state-aware, despite the underlying server selection being stateless (i.e., following an independently drawn probability). This can be derived through a dynamic programming approach [29] where we consider a state of the system $\mathbf{x}_n = (\delta_n, m_n, b_n, u_n)$, with the following components: current value $\delta_n$ of AoI in slot $n$; number of remaining updates in slot $n$; offloading state of the source of interest, denoted as $b_n$, taking values as 0 if the source of interest is not currently offloaded, and $b_n = i > 0$ if the source of interest has been offloaded since $i$ slots; and finally, the number $u_n$ of competing sources that are also presently offloaded on the remote MEC server. The initialization values are $\delta_0 = 0$,[2] $m_0 = M$, $b_0 = 0$, whereas $u_0$ follows the steady-state probabilities of matrix $\mathbf{P}_0$.

Moreover, one can formalize a dynamic program where $\mathbf{x}_n$ evolves according to a binary control action $\mu(\mathbf{x}_n)$, where $\mu = 1$ and $\mu = 0$ correspond to scheduling an update or not, respectively, and also following noise effects consisting of local failures, the random length of the service time on the remote MEC servers, as well as the transitions of the number of competing sources $u$.

We have the following immediate condition for the evolution of $m_n$, in line with existing theoretical models [19], [30]:

$$m_{n+1} = \max(0, m_n - \mu_n) \quad (3)$$

thereby implying that whenever a status update is scheduled, it consumes one of the remaining update opportunities. Additionally, we impose that when $b > 0$ it is not possible to initiate another status update beyond the one that is presently offloaded on the remote server; the only available action is to wait for its processing to finish.[3]

Conversely, when $b_n = 0$ and $\mu_n = 0$, $\delta_{n+1}$ increases by 1 with respect to $\delta_n$, $b_{n+1}$ also remains 0, and $u_{n+1}$ evolves according to $\mathbf{P}_0$, as in $u_{n+1} = v$ with probability $p_{u_n v}$. If $b_n = 0$ but $\mu_n = 1$, then an update is attempted ($m_{n+1}$ decreases by 1) and with probability $\alpha$ a local processing of the update is performed, otherwise the task is offloaded to the remote server. In the former case, the update is successful with probability $1-f$, in which case $\delta_{n+1}$ is reset to 0, otherwise it is increased by 1 from $\delta_n$. Again, the values of $b_{n+1}$ and $u_{n+1}$ are unchanged and evolving through $\mathbf{P}_0$, respectively. Instead, if the update is offloaded, it can be immediately successful with probability $r/(u+1)$, in which case the value of $b_{n+1}$ remains 0 and $\delta_{n+1}$ resets to 0, or it needs to wait for at least one more slot, hence $\delta_{n+1}$ is increased by 1 and $b_{n+1}$ is set to 1. In this case, $u_{n+1}$ evolves according to $\mathbf{P}_1$.

The latter condition, i.e., $u_{n+1}$ evolving according to $\mathbf{P}_1$, also happens when $b_n > 0$, in which case no update is possible, i.e., $\mu_n = 0$. In this case, either the processing is still unresolved in the next slot, which happens with probability $1 - r/(u_n + 1)$, and in this case both $\delta_{n+1}$ and $b_{n+1}$ grow by 1 over their values in slot $n$, or it is finished, which causes $b_{n+1}$ to reset to 0 and $\delta_{n+1}$ to be assigned the value of $b_n$, as per the following equations:

$$\text{if } b_n{>}0 \quad b_{n+1} = \begin{cases} 0 & \text{with prob. } \frac{r}{u_n+1} \text{ (success)} \\ b_n{+}1 & \text{otherwise} \end{cases} \quad (4)$$

$$\delta_{n+1} = \begin{cases} b_n & \text{with prob. } \frac{r}{u_n+1} \text{ (success)} \\ \delta_n{+}1 & \text{otherwise} \end{cases} \quad (5)$$

The condition in (5) accounts for the status update sent to the remote server growing staler if, due to congestion, the update is not immediately processed, thus the value of AoI does not reset to 0 but rather to $b_n$.

All the aforementioned evolution rules prove that the system state $\mathbf{x}_n$ has the Markov property, i.e., $\mathbf{x}_n$ only depends on $\mathbf{x}_{n-1}$ and the control action undertaken, hence it is possible to formalize a Markov decision process to minimize the expected value of cost $g_n(\mathbf{x}_n) = \delta_n$ taken equal to AoI, whose optimal control is $\mu_n(\mathbf{x}_n)$. We can exploit Bellman's optimality condition [29] since, if $\mu_0(\mathbf{x}_0), \mu_1(\mathbf{x}_1), \ldots \mu_{N-1}(\mathbf{x}_{N-1})$ describes the optimal control over the whole horizon, then for any intermediate $n$, $0 < n < N{-}1$ and states $\mathbf{x}_n$ occurring with positive probability, the control minimizing the residual cost from $n$ onwards is $\mu_n, \ldots, \mu_{N-1}$. This implies that one can start from the control action $\mu_{N-1}$ taken at the end of the horizon and assign it as $\mu_{N-1}(\delta, m, 0, u) = 1$ if $m > 0$ and for every $0 \leq \delta \leq N{-}1$ and $0 \leq u \leq U$, whereas $\mu_{N-1} = 0$ in every other case (i.e., either no updating opportunities are left or the last one is currently offloaded, which requires to wait for its termination). Applying backward induction to minimize the expected cost $g_n$ yields the optimal control policy.

TABLE I
NOTATION AND DEFAULT VALUES OF THE PARAMETERS

| Parameter | Symbol | default value |
|---|---|---|
| Time horizon (in slot) | $N$ | 300 |
| Number of update opportunities | $M$ | 3 |
| Remote server completion rate | $r$ | 0.25 |
| Local failure probability | $f$ | 0.3 |
| Local processing rate | $\alpha$ | $\{0.1, 0.3, 0.5, 0.6\}$ |
| Total number of competing sources | $U$ | 40 |

[2]Since we focus on a discrete time, it makes sense to adopt the convention that AoI starts counting from 0. However, this is a purely conventional choice.

[3]Preemption by newer updates may be possible; we choose not to implement it, as its effect for the considered low duty cycles would be negligible.

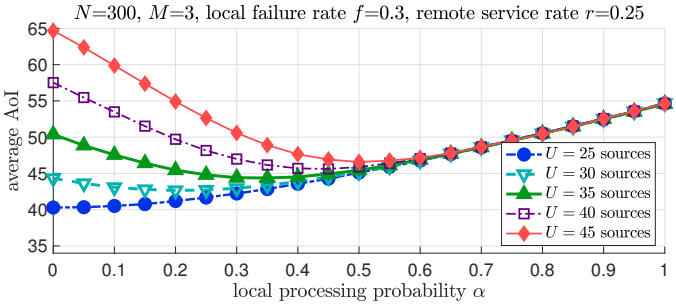


Fig. 2. Optimal average AoI vs. local processing rate $\alpha$ for different numbers of competing sources $U$, for $M = 3$ updates over $N = 300$ slots, remote server success rate $r = 0.25$, local failure probability $f = 0.3$.

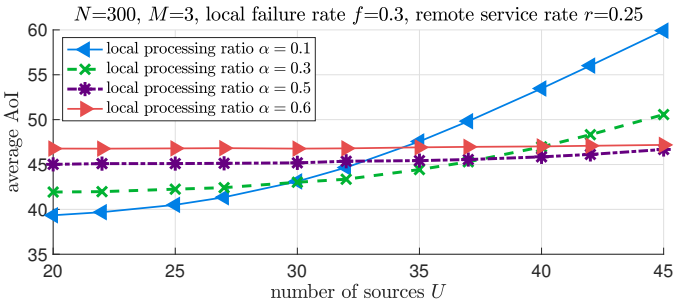


Fig. 3. Optimal average AoI vs. number of competing sources $U$, for different values of the local processing rate $\alpha$, $M = 3$ updates over $N = 300$ slots, remote server success rate $r = 0.25$, local failure probability $f = 0.3$.

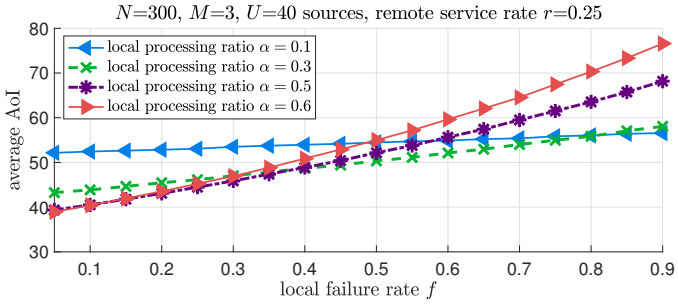


Fig. 4. Optimal average AoI vs. local failure probability $f$, for different values of local processing rate $\alpha$, with $M = 3$ updates over $N = 300$ slots, remote server success rate $r = 0.25$, $U$=40 competing sources.

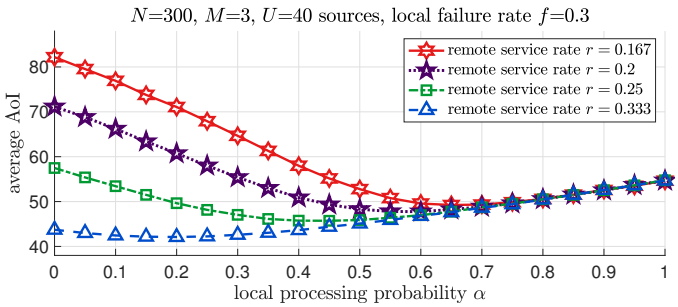


Fig. 5. Optimal average AoI vs. local processing rate $\alpha$ for different values of remote server success rate $r$ for $M = 3$ updates over $N = 300$ slots, $U = 40$ competing sources, local failure probability $f = 0.3$.

## V. Numerical Results

We show numerical evaluations of the optimal scheduling policy derived in the previous section. Table I reports a list of the system parameters, together with their default values that are taken unless specified otherwise. We highlight that the values chosen for $M$ and $N$ imply a duty cycle of $(M+1)/N$ (i.e., an update every $C = N/(M+1) = 75$ slots). This is also a useful reference to quantify the resulting average AoI: In an ideal scenario where status updates are both instantaneous and always successful, unlike ours where those offloaded on the remote MEC server and the local ones only possess either characteristic, the average AoI would be $(C-1)/2 = 37$ slots.

Fig. 2 shows the average AoI as a function of the local processing rate $\alpha$, for different values of the total number of competing sources $U$. The figure highlights how the AoI in the left-side part, corresponding to most of the updates being offloaded, changes with $U$. Conversely, in the right part the system performance is basically the same, so that whenever $\alpha \geq 0.6$ (only a minority of the updates are offloaded) the curves are indistinguishable. This implies that the optimal choice of $\alpha$ is never to leave full local processing of updates, and the remote MEC server ought to be exploited, yet without causing congestions. The optimal choice of $\alpha$ (i.e., the bottom point of the curves) strongly depends on $U$, but we remark that the curves are relatively shallow around the minimum. This is actually a more general and intrinsic characteristic of AoI, with important consequences on distributed systems from the perspective of, e.g., game theory [31].

Our partially stateful approach actually implies that we regard the $U$ competing sources as non-strategic, or at least without any coordination or synchronization with the source of interest. Although a full game theoretic analysis is out of scope for the present paper, this result seems to suggest that the anarchy of the resulting allocation has a limited impact. Namely, it is reasonable to expect selfish sources only interested in their own AoI minimization to overuse offloading to the remote MEC server beyond the optimal point, a property known as the *tragedy of the commons* [32]. Here, its impact is expected not to be dramatic thanks to the robustness of AoI to small local perturbations.

Fig. 3 reverses the plot of Fig. 2 by considering $U$ as the independent variable, and different choices of $\alpha$. This shows that the average AoI is always non-decreasing in the number of competing sources, but when $\alpha \geq 0.5$ the impact is very limited, which makes sense as most of the updates are processed locally, and congestion matters little. This confirms that the optimal choice of $\alpha$ (i.e., the lowest curve) depends on the competition on the remote server. However, high values of $\alpha$ are never optimal as they do not leverage the MEC server.

Fig. 4 shows the average AoI versus the local failure probability $f$, for different values of $\alpha$. This serves to highlight that the AoI increases with $f$, but the slope of the increase depends on how large is the fraction $\alpha$ of updates that are processed locally. As a result, the best value of $\alpha$ is once again shown to depend on the server's characteristics, with a decreasing trend as $f$ grows.

To explore the impact of the remote server, we can look at Fig. 5, where the average AoI is plotted versus $\alpha$, but considering different values of $r$. As a side remark, even though the model does not require this to be an integer number, the choices for $r$ imply that the MEC server completes processing in an average of $\{3, 4, 5, 6\}$ slots. In a sense, this figure is analogous to Fig. 2, with the difference that here we are changing the service rate, whereas in that figure we explored the potentially offered traffic. The consequence is that this figure shows a higher performance variability (see that the AoI spans over a $90\%$ range in the worst cases), and we interpret this as a consequence of the optimality of our scheduler. Indeed, the optimal dynamic control policy allows the source of interest to avoid scheduling a status update when the remote server is congested. Thus, it is essentially better to have more competing sources, as congestion can be contrasted by scheduling updates in the instants of lighter load, than a slower MEC server, which is unavoidable.

## VI. Conclusions

We presented an analysis of the AoI-minimizing partially stateful scheduling in a scenario of MEC computing [21], exploring the role of computational offloading in the presence of other sources, which can cause congestion at the remote server's side [13], [15]. We showed how the optimal scheduling policy can be obtained through a dynamic programming approach [20], [30], and important results can be found, which paves the road for further extensions of the analysis.

One follow-up may involve a joint optimization of the update instants as well as the value of $\alpha$, with a layered iterative approach. Advanced investigations can include multiple *strategic* sources, studied under the lens of game theory. As is known, the resulting Nash equilibrium does not necessarily reflect into an efficient allocation [4], [10], [32], and it may be challenging to introduce a form of distributed cooperation. This appears as an interesting direction for future research.

## References


[1] B. Picano, R. Fantacci, and Z. Han, "Nonlinear dynamic chaos theory framework for passenger demand forecasting in smart city," *IEEE Trans. Veh. Tech.*, vol. 68, no. 9, pp. 8533–8545, 2019.

[2] F. Chiariotti, "Age of information analysis for a shared edge computing server," *IEEE Trans. Commun.*, vol. 72, no. 12, pp. 7826–7841, 2024.

[3] G. Cisotto and S. Pupolin, "Evolution of ICT for the improvement of quality of life," *IEEE Aerosp. Elec. Syst. Mag.*, vol. 33, no. 5-6, pp. 6–12, 2018.

[4] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Th.*, vol. 65, no. 3, pp. 1807–1827, 2019.

[5] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. IEEE SECON*, 2011, pp. 350–358.

[6] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Trans. Inf. Th.*, vol. 62, no. 4, pp. 1897–1910, 2016.

[7] H. Li, J. Zhang, H. Zhao, Y. Ni, J. Xiong, and J. Wei, "Joint optimization on trajectory, computation and communication resources in information freshness sensitive MEC system," *IEEE Trans. Veh. Tech.*, vol. 73, no. 3, pp. 4162–4177, 2024.

[8] Y. Wang, C. Yang, S. Lan, L. Zhu, and Y. Zhang, "End-edge-cloud collaborative computing for deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 4, pp. 2647–2683, 2024.

[9] A. Munari, T. De Cola, and L. Badia, "Local or edge/cloud processing for data freshness," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2023, pp. 1–6.

[10] B. Picano and E. Mingozzi, "Age-oriented resource allocation for IoT computational intensive tasks in edge computing systems," *IEEE Internet Things J.*, 2025, to appear.

[11] F. Bahramisirat, M. A. Gregory, and S. Li, "Multi-access edge computing resource slice allocation: A review," *IEEE Access*, vol. 12, pp. 188 572–188 589, 2024.

[12] J. Morgan, H. Orzen, and M. Sefton, "Network architecture and traffic flows: Experiments on the Pigou–Knight–Downs and Braess paradoxes," *Games Econ. Behav.*, vol. 66, no. 1, pp. 348–372, 2009.

[13] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Stateful versus stateless selection of edge or cloud servers under latency constraints," in *Proc. IEEE WoWMoM*, 2022, pp. 110–119.

[14] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2637–2650, 2018.

[15] M. Moltafet, M. Leinonen, and M. Codreanu, "On the age of information in multi-source queueing models," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5003–5017, 2020.

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

[17] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "The age of information in a discrete time queue: Stationary distribution and non-linear age mean analysis," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1352–1364, 2021.

[18] O. T. Yavascan and E. Uysal, "Analysis of slotted ALOHA with an age threshold," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1456–1470, 2021.

[19] A. Munari and L. Badia, "The role of feedback in AoI optimization under limited transmission opportunities," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2022, pp. 1972–1977.

[20] E. Fountoulakis, T. Charalambous, A. Ephremides, and N. Pappas, "Scheduling policies for AoI minimization with timely throughput constraints," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 3905–3917, 2023.

[21] P. Han, B. Liu, Y. Liu, and L. Guo, "Cell-less offloading of distributed learning tasks in multi-access edge computing," *IEEE Trans. Mob. Comp.*, vol. 23, no. 12, pp. 14 377–14 395, 2024.

[22] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," in *Proc. ACM SIGCOMM*, 2003, pp. 151–162.

[23] Y. Dong, H. Xiao, H. Hu, J. Zhang, Q. Chen, and J. Zhang, "Mean age of information in partial offloading mobile edge computing networks," *arXiv preprint arXiv:2409.16115*, 2024.

[24] Z. Tang, Z. Sun, N. Yang, and X. Zhou, "Age of information of multi-user mobile edge computing systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1600–1614, 2023.

[25] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[26] L. Badia and A. Munari, "Exogenous update scheduling in the industrial Internet of things for minimal age of information," *IEEE Trans. Ind. Informat.*, vol. 21, no. 2, pp. 1210–1219, 2024.

[27] D. Kim, G. Hwang, O. Jo, and K. Shin, "Q-learning based medium access technology for minimizing AoI in LoRa wireless relay networks," *IEEE Access*, vol. 12, pp. 183 024–183 037, 2024.

[28] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *IEEE/ACM Trans. Netw.*, vol. 2, no. 2, pp. 137–150, 1994.

[29] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012.

[30] A. Javani, M. Zorgui, and Z. Wang, "On the age of information in erasure channels with feedback," in *Proc. IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[31] V. Mancuso, P. Castagno, L. Badia, M. Sereno, and M. Ajmone Marsan, "Optimal allocation of tasks to networked computing facilities," in *Proc. Int. Conf. An. Stoch. Model. Techn. Appl. (ASMTA)*, 2024, pp. 33–50.

[32] L. Badia and A. Munari, "A game theoretic approach to age of information in modern random access systems," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.