Age of Information for Machine Learning Tasks With Mobile Edge Computing Offloading

Leonardo Badia*, Paolo Castagno†, Vincenzo Mancuso‡§, Matteo Sereno† and Marco Ajmone Marsan§
*Dept. Information Engineering, University of Padova, 35131 Padua, Italy
†Computer Science Department, University of Turin, 10125 Turin, Italy
‡Department of Engineering, University of Palermo, 90133 Palermo, Italy
§IMDEA Networks Institute, 28918 Leganes (Madrid), Spain
Email: leonardo.badia@unipd.it, {paolo.castagno,matteo.sereno}@unito.it,
vincenzo.mancuso@ieee.org, marco.ajmone@imdea.org

***Abstract*—We investigate the minimization of the age of information (AoI) of an AI-powered application that requires timely processing of data generated by a multitude of users. We consider that sequences of inference tasks generated at individual terminals can either be processed locally with a tiny machine learning (ML) model or be offloaded to a more powerful ML model residing on an edge computing facility shared by all users. Since the local ML model is less powerful, its inferences may have low confidence. When this happens, the user is forced to repeat the inference with the more powerful edge ML model. The choice between local processing or offloading follows a randomized-alpha policy, where the local ML model, while less powerful, offers the advantage to alleviate congestion of the edge server. The AoI model follows the frameworks presented in the literature for multiple sources sharing the same queue. Local processing instead works as a single-server dedicated queue, but we account for the imperfections of the tiny ML model by including a failure probability in the local server. Tasks that are processed locally but eventually fail to achieve a minimum confidence level are offloaded to the edge server, resulting in a longer overall processing time. We derive a queueing model of the entire system based on some bounds from the literature. Our results show the trade-offs between processing latency, inference accuracy, and system congestion, highlighting the importance of optimizing task allocation strategies.**

***Index Terms*—Age of Information; Machine learning; Mobile-edge computing; Resource sharing.**

I. Introduction

Machine learning (ML) aided applications, especially involving large language models (LLMs), often require significant computational resources to interpret data due to their complexity and multidimensionality [1]. Failure to obtain a processor that is both sufficiently powerful and free from congestion can lead to long inference times, particularly when accurate predictions require evaluation of many parameters [2].

For example, convolutional neural networks used in image recognition or transformer-based models in natural language processing may involve billions of parameters [3]. Even simpler models, such as support vector machines or decision trees, can experience considerable delays when handling large-scale data or complex decision boundaries [4]. As a result, applications requiring immediate responses, such as autonomous driving or medical monitoring, may suffer latency issues that hinder their effectiveness [5].

Delayed data interpretation is particularly problematic in scenarios where rapid decision-making is critical. In real-time applications, such as industrial automation or emergency response systems, any delay in processing can lead to substantial inefficiencies and a possible safety hazard [6], [7]. The usual solution invoked to deal with computationally intensive tasks is to resort to more powerful remote servers [8]. In our scenario, this can actually further exacerbate the problem by introducing additional latency due to network congestion [9], also due to the convergence of requests from multiple users [10].

This prompts a joint analysis of resource allocation in the edge-cloud continuum to optimize the offloading of tasks and include the freshness of the resulting outputs. The latter can be connected to the characterization available in the literature through age of information (AoI) [11]. For our purposes, we need to extend the standard AoI models, which usually focus on atomic sensing and/or measurement tasks, to include traffic splitting, merging, and processing times [12]–[14].

We consider a scenario in which multiple terminals perform sequences of ML tasks, as would be the case for persistent monitoring of patients, motion tracking, or ambient surveillance. We investigate the freshness of the ML output, captured through AoI as the main performance metric [15], [16]. Moreover, we study the impact of network decisions on AoI, assuming that devices can either process tasks locally (e.g., with their onboard capabilities and/or at a close-by dedicated server) with a tiny ML model, or offload them to a more powerful remote computing facility at the network edge where a better ML model is available [17]. Neither of these solutions is optimal if taken alone, as far as timeliness is concerned [18], since the remote server can become congested if too many tasks are offloaded there by all the terminals that share it, while local processing alone is possibly not accurate enough

This work has been supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"), S2 SUPER – Programmable Networks, Cascade project PRISM - CUP: C79J24000190004. Marco Ajmone Marsan's activity was supported by the TUCAN6-CM project (TEC-2024/COM-460), funded by CM, the Regional Government of Madrid, Spain (ORDEN 5696/2024).

and may require further evaluation on the remote edge server, which makes offloading unavoidable [19].

As a result, careful balance between task offloading and local processing must be sought. To characterize it, we focus on a randomized-alpha policy as introduced in [20]. In our analysis, we leverage the (queuing-based) derivations available in the literature to characterize average AoI in queues with superimposed flows, for two different cases. In both, we use an M/M/1 model [21]. However, the local processing is assumed to behave like a queue with concurrent flows corresponding to the correct and failed classification, respectively; the latter delays the queue but does not decrease the AoI. Conversely, the remote edge server never fails its classification, yet is represented as a queue where flows from multiple sources converge [22].

Our analysis allows us to highlight how the joint distributed optimization of relevant parameters, specifically the offloading rate and the intensity of task generation, is neither viable nor robust to errors. A small deviation of the values from the optimum can lead to performance degradation. Conversely, reaching convergence on the amount of local processing and optimizing the generation rate afterwards is much more robust.

The remainder of this paper is organized as follows. In Section II, we review the related literature. Section III expands analytical formulas taken from seminal papers to the specific case at hand, giving an estimate of the average AoI under different offloading choices. We present numerical results in Section IV, and we conclude in Section V.

II. Related Work

Considering the timeliness of ML tasks is necessary due to the recent surge in the adoption of AI for many real-time applications. However, the idea of evaluating data freshness of ML tasks and how networking choices influence it is relatively unexplored in the recent literature. Only a handful of papers consider these aspects, and for the most they just claim that long execution times of AI-driven applications may lead to frequently obtaining accurate but obsolete classification.

This is relevant for medical monitoring leveraging multimodal LLM; however, producing stale output from these models may be entirely useless or, worse, harmful. This point is advanced by [23] when proposing an AoI-aware semantic scheduler for sensed data in a body area network. In the same spirit, [24] discusses this issue as a byproduct of data sharing and security aspects. The authors investigate data freshness, but only indirectly to evaluate its influence on the usefulness of the output, whereas the modeling aspects concern the application layer and not the resource allocation.

Another field of real-time applications exploiting ML tasks is autonomous driving (for both terrestrial and unmanned aerial vehicles), where computationally intense computer vision jobs can be performed either by some units available on-board or offloaded to some external edge server. As argued in [25], safe and seamless vehicle synchronization would require consistent use of fresh sensing, to which end the authors propose an AoI-driven scheduling.

Instead, [10] considers an edge computing server that performs processor sharing among the offloading sources, possibly focusing on ML-related time critical applications. The analysis expands on the way of sharing the queue among multiple sources and leverages results similar to ours in the derivation, but it does not argue about server selection or the failure rate of the tiny ML processing.

Another recent paper [14] focuses on how offloading to mobile edge computing servers affects AoI; however, it uses stochastic geometry to study physical placement of computation resources and does not give a performance evaluation through queueing theory. The conclusions are that a careful balance of the available resources should be used, so the authors propose a *partial offloading* approach. This is analogous to the randomized-alpha policy proposed in [20], which we adopt in this paper, with the difference that the latter randomly assigns individual tasks to different servers, whereas the former paper considers all tasks partially processed in multiple units. Also, [14] focuses on a wireless channel and possibly successful or erroneous transmissions depending on signal quality, whereas we assume an independent and identically distributed failure probability of the tasks that follows from the underlying ML model. In addition, they consider instantaneous output of their computationally intense tasks, whereas we keep into account the propagating effect of congestion on the remote edge computing facility, which may actually choke data freshness.

The authors of [26] propose a joint optimization of server selection in MEC scenarios, also involving the allocation of bandwidth within the shared wireless channel and computing resources within the shared computing facilities. Similar to our investigation here, the paper considers a choice between local and edge processing, invokes ML tasks in the evaluation, and considers AoI in the evaluation metric. However, the main goal of that paper lies in the optimal server selection, as opposed to finding closed form expressions for AoI.

In general, it is evidenced by the recency of these references that the issue of evaluating the average AoI of ML-driven real-time applications remains largely unexplored in the literature, not only in terms of raising awareness about it, but even more so in developing analytical frameworks of the network choices impacting on it, so as to optimize performance.

For what concerns the modeling of AoI within queueing systems, a kind of analysis that enjoyed popularity in the last decade already from the seminal papers introducing the idea of AoI [11], there are indeed analytical results such as the derivation of the average AoI for an M/GI/1 queue [27] as well as the numerous results in [21] (and references therein), especially considering queues with multiple sources.

While no previous paper combines queues of multiple kinds, we explicitly consider the dichotomy between local and edge processing, seen as an exclusive server with limited rate (and possibly failures) versus another queue with higher service rate but shared with other sources. This is meant to leverage analytical instruments and may be seen as a first step towards deriving a fully closed-form characterization of such systems.

III. AoI of Tasks with Offload Options

Consider a time-critical application run by N terminals. We imply that each terminal generates a sequence of ML tasks to be processed, according to a Poisson point process of intensity λ, which means that task generation is memoryless. These tasks can be processed on a local or a remote server, according to independently drawn probabilities. Due to the properties of Markov processes, each server sees a sequence of tasks with exponentially distributed inter-arrival times.

The decision on where to process it is made independently for each task, according to a randomized-alpha policy [20], which means that a task can be processed locally with probability α or offloaded to a remote (mobile edge) server facility with probability $1-\alpha$. Local processing with the tiny ML model translates into an M/M/1 FCFS queue with service rate μ_1, i.e., the service time is exponentially distributed and arrivals are memoryless, as per the previous discussion.

We consider that the processing in the remote facility uses a single server according to an M/M/1 FCFS queue whose service rate is $\mu_2 > \mu_1$. We remark that this assumption is just preferred for the sake of a simpler exposition and the better availability of some analytical components. We tested different queueing systems, also including multiple servers, and under reasonable approximations the results are qualitatively similar. To work efficiently, the shared remote facility must approach instability, without reaching it, which means that the main aspect to consider is the overall service rate μ_2. More than the queueing notation characterizing the remote facility, the difference from local processing is that while the M/M/1 queue describing the latter is reserved for that individual user, all the N terminals use the same remote facility, where the tasks offloaded by all of them converge.

Finally, we also consider that local processing is not always successful. We define a local success probability, ς, that captures how often the accuracy of the tiny ML model is adequate. In contrast, remote processing at the edge computing facility is assumed to always be successful. This assumption can be justified by either the higher reliability of the remote classifier or the fact that tasks offloaded there are repeatedly processed until success. Instead, tasks whose inference in the tiny ML model is not successful are subsequently transferred to the remote facility after the first attempt.

This system model, shown in Fig. 1, implies that a task can follow one of three possible routes: (i) With probability $\alpha\varsigma$, it is processed locally successfully. (ii) With probability $1-\alpha$, it is immediately offloaded to the remote facility. (iii) With probability $\alpha(1-\varsigma)$, it is processed locally but fails, and is eventually sent to the remote facility, traversing both queues.

This would imply that the entire system can be characterized as an M/G/1 queue, where we classify the service discipline as "general" since we account for the linear combination of the three cases described above. For this scenario, some analytical results on AoI are well established [27]. However, to further complicate things, processing in some of the paths involves other sources sharing the queue. For this reason, we resort

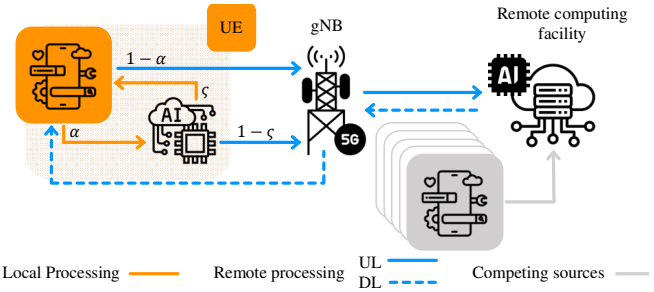

Fig. 1. System model. Local processing is chosen with probability α but is successful only with probability ς. In the case of offloading (chosen with probability $1-\alpha$) or failure of the local processing (probability $\alpha(1-\varsigma)$) the task is processed in the remote computing facility, shared by N terminals.

to leveraging the known result [10], [21] that states that the average AoI, Δ, can be computed as:

$$\Delta = \frac{\mathbb{E}[T^2]/2 + \mathbb{E}[ST]}{\mathbb{E}[T]}\,, \tag{1}$$

where T is the interarrival time in the queue, i.e., the time between generation of tasks by one source, and S is the system time (i.e., the sum of the waiting time in the queue and the service time). The principle of (1) is well known and follows from geometric arguments [22].

Since (1) requires memoryless arrivals but works for a generic service process, we can include a fixed delay to reach the edge server, which in many cases can be relevant [9]. For the sake of simplicity in the exposition, we do not consider this extension in the analysis, although it would be immediate to include the fixed delay to reach the edge server within the term S. At the end of Section IV, we will show its impact.

In the system at hand, T is exponentially distributed, so clearly $E[T] = 1/\lambda$ and $E[T^2] = 2/\lambda^2$, thus (1) simplifies to

$$\Delta = \frac{1}{\lambda} + \lambda\mathbb{E}[ST]\,. \tag{2}$$

Computing $\mathbb{E}[ST]$ would be easy for an M/M/1 queue. In the case under exam, it becomes complex due to the parallel service routes that a packet can experience. For an individual packet, one can write $\mathbb{E}[S] = \alpha(\mathbb{E}[S])_{\mathrm{L}} + (1-\alpha\varsigma)(\mathbb{E}[S])_{\mathrm{R}}$, where subscripts L and R stands for "local" and "remote," respectively. This holds since, according to the adopted system model, the time spent by a task is the weighted sum of the probability of being served through the local processing and the edge server, where with probability $\alpha(1-\varsigma)$ an initial local processing fails and is then followed by the remote processing; therefore, the task experiences both system times.

To compute both $(\mathbb{E}[S])_{\mathrm{L}}$ and $(\mathbb{E}[S])_{\mathrm{R}}$ we can use (35) in [21]. For $(\mathbb{E}[S])_{\mathrm{L}}$, we consider the AoI minus the term $1/\lambda$ by taking a useful traffic of $\alpha\varsigma\lambda$ and a competing traffic of $\alpha(1-\varsigma)\lambda$, representing unsuccessful classifications, insisting on the same M/M/1 server. For $(\mathbb{E}[S])_{\mathrm{R}}$, we adopt the same approach, but we consider the useful and competing traffic rates as $(1-\alpha\varsigma)\lambda$ and $(N-1)(1-\alpha\varsigma)\lambda$, respectively, since in this case there are no failures but extra traffic is contributed by other $N-1$ sources.

However, the above approach results in an expression, denoted as Δ_1, which is not correct unless the offered load on both servers is low. In this case, there is no queue on either server, and the term $\mathbb{E}[ST]$ in (2) can be split into $\mathbb{E}[S]\cdot\mathbb{E}[T]$ and subsequently $\mathbb{E}[S]$ can be written as a weighted average. In general, Δ' can be seen as an *upper bound* of AoI [12], where we neglect the option that tasks are processed along multiple routes in different order of arrival.

As argued in [21], based on the considerations also advanced by [28] and [12], a scenario with parallel service is generally complex. However, a solution is proposed in [29] by approximating the age through multiple servers with the weighted harmonic mean.

Thus, we take a further estimate Δ_2 of the average AoI as the weighted harmonic average of the paths, i.e.,

$$\Delta_2 = \frac{1}{\lambda} + \left[\frac{\alpha\varsigma}{(\mathbb{E}[S])_{\mathrm{L}}} + \frac{1-\alpha}{(\mathbb{E}[S])_{\mathrm{R}}} + \frac{\alpha(1-\varsigma)}{(\mathbb{E}[S])_{\mathrm{L}} + (\mathbb{E}[S])_{\mathrm{R}}}\right]^{-1}.$$

Finally, as an empirical estimate, which is further verified through simulation in the following section, we estimate the average AoI Δ as $(\Delta_1+\Delta_2)/2$.

IV. Results

We evaluate the formulas derived in the previous section in a scenario where N terminals behave as sources of ML tasks associated with a time critical application. These ML tasks become computation jobs that are processed by FCFS queues either locally or in a remote computing facility (i.e., a mobile edge computing server). The choice about this split is via a randomized-alpha policy, i.e., each task is assigned a probability α of local processing, whereas with probability $1-\alpha$ it is offloaded to the remote server. The individual local processing queue is exclusive to that source; the remote server is instead shared by all offloaded traffic. Moreover, while the offloaded tasks are always completed with success, local processing is only successful with probability ς (i.e., local inference does not achieve a sufficient confidence level with probability $1-\varsigma$). If local processing fails, the task is also sent (after local processing) to the remote server. All time and rate values are reported as normalized to the local service rate μ_1, taken equal to 1 (inverse time units). The same plots work by rescaling the numerical values if the same proportion of λ and μ_1, μ_2 is kept.

In Fig. 2, we validate our estimate of the average AoI. The parameters are μ_1=1, μ_2=16, N=25, ς=0.8, while we consider λ as the independent variable and different values for α. We plot the results of the analytical estimate, the upper and lower bounds, and the simulation results. To understand the figure, remember that this scenario involves the joint management of two parameters with different meanings, that is, the probability of local processing α and the intensity of traffic generation λ [17]. Most of the literature investigates the optimization of AoI via λ alone [22]. We show that the optimization strongly depends on α too, controlling how much traffic is sent to either server. Choosing a different α does not mean that the AoI cannot be close to optimal, but λ becomes

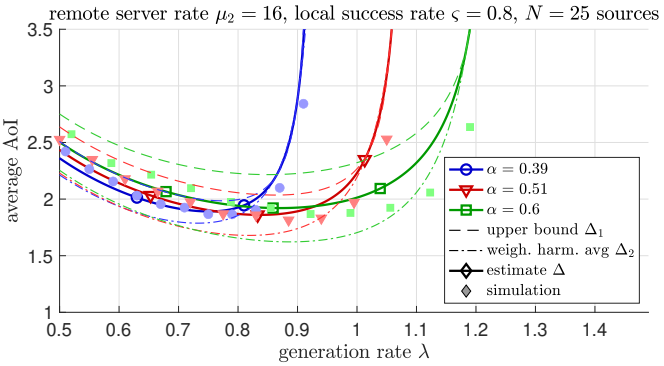

Fig. 2. Average AoI vs. data generation rate λ, for different values of local processing probability α, local success probability $\varsigma = 0.8$, $N = 25$ offloading sources, remote server rate $\mu_2 = 16$. All values are normalized to μ_1 taken equal to 1 per unit time.

completely different. In other words, a joint optimization of both parameters at once is fragile, which makes a distributed optimization challenging.

In particular, the curves rise relatively sharply as the edge server instability approaches. In this case, the simulation results show a lower value since they are computed over successfully processed tasks only, but there are also many incomplete or unprocessed tasks when instability is approached. As a side note, it is difficult to exactly predict the optimum, since it happens when the gap between the upper and lower bounds is wider. This is because optimal server usage requires a careful balance of both options for local or remote processing, which is exactly when the analytical estimates are less accurate. However, the lowest point of each curve occurs just before the more powerful remote edge server reaches instability. In this sense, local processing can be seen as a way to alleviate the load on the remote server, in a way similar to classic approaches to active queue management, that is, random removal of tasks to avoid congestion [30]. Compared to these approaches, we are not dropping tasks, but processing them elsewhere, which represents a further improvement if properly done.

However, this implies that identifying the precise pair of optimal α and λ is quite a challenge. Increasing α further pushes the limit of an unstable edge server, which causes the lowest point of the curve to occur for a higher λ. In general, a joint optimization of these two parameters is difficult and possibly subject to high cross-variability of the parameters, i.e., a small variation in α implies a high variability of the optimal associated λ. In contrast, it is notable that the lowest point of each curve is substantially equivalent, which will be further discussed later on.

If we focus on the minimum achievable AoI with each choice of α, we get Figs. 3 and 4. These are basically the envelopes of the minima in Fig. 2, plotting the y-axis value (minimum) and the x-axis value (minimizing point), respectively, for different values of N. The local success probability is kept at $\varsigma = 0.8$ as before.

Differently from the individual curves of Fig. 2, these are

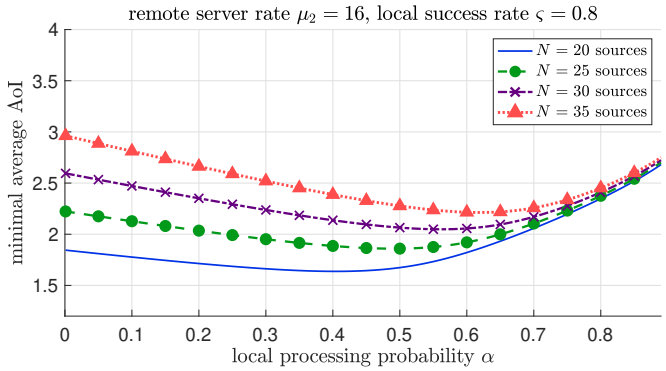

Fig. 3. Minimum AoI vs. local processing probability α, under optimization of the data generation rate λ, for different numbers of offloading sources N, local success probability $\varsigma = 0.8$, MEC service rate $\mu_2 = 12$. All values are normalized to μ_1 taken equal to 1 per unit time.

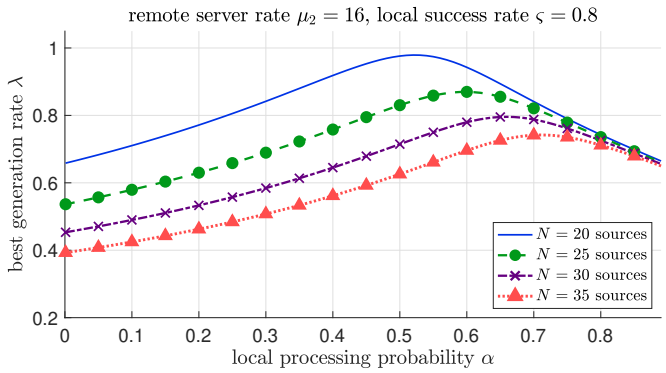

Fig. 4. AoI-minimizing data generation rate λ vs. local processing probability α, for different numbers of offloading sources N, local success probability $\varsigma = 0.8$, MEC service rate $\mu_2 = 12$. All values are normalized to μ_1 taken equal to 1 per unit time.

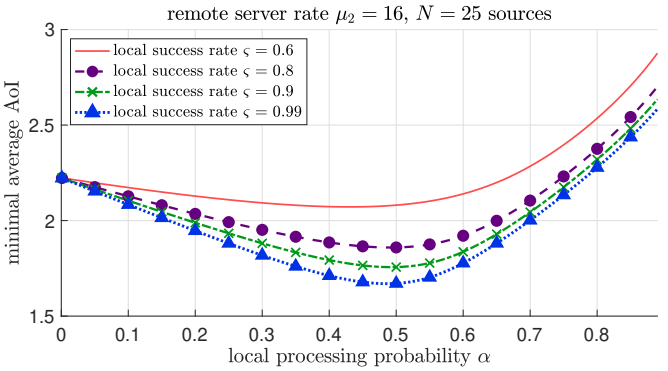

Fig. 5. Minimum AoI vs. local processing probability α, under optimization of the data generation rate λ, for different values of local success probability ς, N=30 offloading sources, MEC service rate $\mu_2 = 12$. All values are normalized to μ_1 taken equal to 1 per unit time.

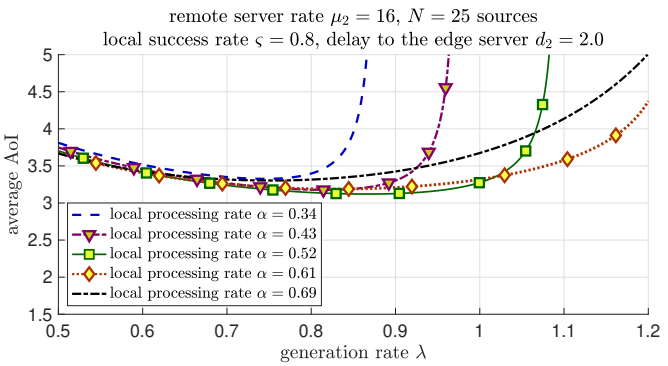

Fig. 6. Average AoI vs. data generation rate λ, for different values of local processing probability α, local success probability $\varsigma = 0.8$, $N = 20$ offloading sources, remote server rate $\mu_2 = 12$. All values are normalized to μ_1 taken equal to 1 per unit time.

relatively flatter. Although there is a minimum for the average AoI and, as such, a best choice of the task generation rate λ, a suboptimal choice of α alone does not cause a significant change, provided that λ is still optimized. Thus, unlike the fragility shown by the previous figure, setting the value of α alone is relatively robust. This suggests an approach in which a target value of α, even if not optimal but not far from it, is communicated to the entire network, which would still obtain near-optimal performance at the price of a small communication exchange. In other words, while full-scale optimization of the parameters, despite obviously achieving the best results, is neither practical nor robust enough, limited coordination and message exchange can be applied to improve system management. If the terminals cooperate in the choice of λ, they can converge towards an efficient choice.

Analogous to the previous figures, in Fig. 5 we plot again the minimal average AoI but considering N fixed to 30 and varying the success probability ς of local processing in each curve. The trends are similar to Fig. 3, but in reverse, and we remark that a similar plot can be drawn to show the minimizing value of λ instead. In fact, the number of sources only affects the shared edge server, while the probability of success only influences the local server, so the curves do not differentiate when α=1 (only local processing) in Fig. 3 and when α=0 (all traffic is offloaded) in Fig. 5.

Notice that lowering ς makes the AoI optimization more difficult than increasing the number of users, as it makes it more likely that a task is processed twice, thereby increasing traffic on both servers. For a low ς (especially below 0.5), the margin for optimization of the average AoI is limited, as most tasks processed by the tiny ML model do not reach an adequate level of confidence, and further processing on the edge server is still required, which nullifies the role of local processing in alleviating congestion for the shared part.

Finally, we also consider the impact of a constant delay to reach the edge server [9]. To this end, we consider the same queueing systems as before, but reaching the edge server now implies an extra delay d_2 constantly equal to 2.0 time units. Analysis-wise, this only implies to replace the previous computation of S_{remote} with $S_{\text{remote}} + d_2$. Although this extra delay worsens performance in terms of average AoI, it also makes the edge server, in a sense, less susceptible to congestion, since the constant delay d_2 is always present and does not depend on the offered load in the edge computing

facility. For space reasons, we do not replot all the curves previously shown, but we just limit the analysis to Fig. 6, which is analogous to Fig. 2 but with the extra delay d_2 now kept into account. In this figure, the curves for different values of α become more similar (and a similar trend can be shown for all other plots), therefore strengthening our conclusion that, instead of looking for a fine-tuned optimization of α simultaneous to the data injection, a layered approach where α is pre-determined and λ is individually optimized would be basically equivalent but easier to implement.

V. Conclusions and Future Work

We analyzed the average AoI of a multiserver system with alternative paths for task processing, which represents a scenario where N terminals execute a time-critical application that requires the processing of a sequence of computationally intensive ML tasks. The options available to the terminals are to process the task locally, with a tiny ML model, possibly leading to low confidence inferences, or to offload the task to a remote computing facility at the network edge, more powerful and accurate, but prone to congestion [14], [17].

In our analysis, the choice of the offload probability is parametric and shared by the entire network, yet this kind of stateless policy is still efficient and often more robust than fine-tuned optimization [20]. Our results confirm this claim by demonstrating that the joint optimization of the offload probability and the intensity of traffic generation can be fragile, potentially leading to congestion or suboptimal choices of parameters. Conversely, a layered approach with limited signaling, such as binding all terminals to use the same offload probability but leaving them free to choose their own data injection rate, is more robust and efficient.

There are many possible developments of our analysis, e.g., considering alternative queue models, such as multiserver systems or deterministic service, as well as including preemption or buffer limitations. Another interesting extension involves game theory, with a comparison between optimal control and Nash equilibrium along the lines of [9]. Finally, future work can evaluate ML approaches in practical settings, identifying vulnerabilities that arise under heavy usage.

References

[1] M. Shao, A. Basit, R. Karri, and M. Shafique, "Survey of different large language model architectures: Trends, benchmarks, and challenges," *IEEE Access*, vol. 12, pp. 188 664–188 706, 2024.

[2] K. Zen, S. Mohanan, S. Tarmizi, N. Annuar, and N. U. Sama, "Latency analysis of cloud infrastructure for time-critical IoT use cases," in *Proc. Appl. Inform. Int. Conf. (AiIC)*, 2022, pp. 111–116.

[3] H. Luo, Y. Yang, B. Tong, F. Wu, and B. Fan, "Traffic sign recognition using a multi-task convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1100–1111, 2017.

[4] F. S. Abkenar, L. Badia, and M. Levorato, "Online domain adaptive classification for mobile-to-edge computing," in *Proc. IEEE Symp. World Wirel. Mob. Multimedia Netw. (WoWMoM)*, 2023, pp. 21–29.

[5] A. Telikani, A. Sarkar, B. Du, and J. Shen, "Machine learning for UAV-aided ITS: A review with comparative study," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 15 388–15 406, 2024.

[6] 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, 2025.

[7] Y. Inoue and T. Kimura, "Age-effective information updating over intermittently connected MANETs," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1293–1308, 2021.

[8] 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.

[9] 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.

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

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

[12] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. IEEE Int. Symp. Inf. Th. (ISIT)*, 2015, pp. 1681–1685.

[13] 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.

[14] 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.

[15] 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. Technol.*, vol. 73, no. 3, pp. 4162–4177, 2024.

[16] H. Sedghani, F. Filippini, and D. Ardagna, "SPACE4AI-D: A design-time tool for AI applications resource selection in computing continua," *IEEE Trans. Service Comput.*, vol. 17, no. 6, pp. 4324–4339, 2024.

[17] L. Badia and A. Munari, "Partially stateful server selection for minimal age of information scheduling over a finite horizon," in *Proc. IEEE Infocom ASoI Workshop*, 2025.

[18] 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.

[19] A. Buratto, B. Yivli, and L. Badia, "Machine learning misclassification within status update optimization," in *Proc. IEEE Int. Conf. Commun. Netw. Satellite (COMNETSAT)*, 2023, pp. 640–645.

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

[21] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, 2021.

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

[23] B.-S. Kim, "Semantic-aware scheduling for minimizing age of informative data in WBAN-based health monitoring systems," *IEEE Internet Things J.*, 2025, early access.

[24] C. Su, J. Wen, J. Kang, Y. Wang, Y. Su, H. Pan, Z. Zhong, and M. S. Hossain, "Hybrid RAG-empowered multi-modal LLM for secure data management in Internet of medical things: A diffusion-based contract approach," *IEEE Internet Things J.*, 2025, early access.

[25] T. Shi, Q. Xu, J. Wang, C. Xu, K. Wu, K. Lu, and C. Qiao, "Enhancing the safety of autonomous driving systems via AoI-optimized task scheduling," *IEEE Trans. Veh. Technol.*, vol. 74, no. 3, pp. 3804–3819, 2025.

[26] M. Kim, J. Jang, Y. Choi, and H. J. Yang, "Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 12, pp. 15 149–15 166, 2024.

[27] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka, "A general formula for the stationary distribution of the age of information and its application to single-server queues," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 8305–8324, 2019.

[28] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Trans. Inf. Theory*, vol. 62, no. 3, pp. 1360–1374, 2015.

[29] R. Talak and E. H. Modiano, "Age-delay tradeoffs in queueing systems," *IEEE Trans. Inf. Theory*, vol. 67, no. 3, pp. 1743–1758, 2020.

[30] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 362–373, 1998.