Local or Edge/Cloud Processing for Data Freshness

Andrea Munari, Tomaso De Cola
Institute of Communications and Navigation
German Aerospace Center (DLR), Weßling, Germany
email: {andrea.munari, tomaso.decola} @dlr.de

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

***Abstract*—Several actuation choices in the Internet of things (IoT) require state information to be up-to-date. In this context, the choice between local processing and offloading to a more powerful remote server can have a significant impact on data freshness. Local processing is indeed typically less reliable due to limited computing power and may require multiple attempts, which can result in data becoming stale. Conversely, remote (i.e., edge/cloud) processing can be more reliable, yet entail longer response times due to data transmission to an external server, possibly performing processor sharing with other tasks. The optimal balance depends on the specific system conditions, especially on the congestion at the remote server, and is in general non-trivial. We explore this tradeoff through a mathematical model, and argue about the derivation of a freshness-efficient local vs. remote split of requests, providing insights on the role played by reliability and latency in selecting the optimal strategy.**

***Index Terms*—Age of information; Edge computing; Signal-flow graphs; Mathematical modeling.**

I. Introduction

A pervasive integration of multiple devices in the Internet of Things (IoT) is expected for a number of applications, including autonomous vehicles, industry 4.0, augmented reality, smart healthcare, and the Metaverse [1], [2]. This leads to the generation of vast amounts of data at the network border, which poses difficult challenges for traditional computing architectures to deal with the requirements in terms of computational efficiency and real-time promptness [3].

To address these issues, two approaches are generally considered: offloading to more resource-abundant remote servers, or local processing within less powerful peripheral units with the advantage of a closer proximity to the end user [4]. In this paper, we refer to this dichotomy as "remote" versus "local" processing, which may be declined in different instances.

Local processing is typically operated inside a single IoT end device, but in a broader sense can also be meant to include scenarios of coordination mechanisms in a cluster of neighbor nodes [5]. The emphasis here is on the relevant computational power being available in the proximity of where data are generated. Local computation and processing incur low latency and bandwidth usage, at the price of a worse accuracy [6].

In contrast, remote offloading involves transferring information bundles from IoT end devices to further data centers, possibly in the cloud, for processing and analysis. It is also common to envision edge computing architectures, where computational power and storage capabilities are brought at the network's edge [7]. In this context, processing tasks can be offloaded to closer, but still external, edge servers. We consider both cases as instances of remote processing, since the IoT end nodes just act as relays for data to some external agents, regardless of their location in the cloud or at the network's edge. The advantage of remote offloading is to leverage a more powerful infrastructure, providing abundant resources and advanced analytics, but it can suffer from communication round-trip delays, as well as network congestion, whenever multiple end nodes offload their tasks [8].

The choice between local computing and offloading to remote edge/cloud servers involves tradeoffs, e.g., between latency and reliability [9], [10], which are crucial for designing efficient IoT systems. These factors must also be seen through the lens of a proper key performance indicator. In the literature, the objective is often seen as the maximization of the amount of raw data processed, the minimization of the latency, or the limitation of the outage probability for service requests.

Conversely, in this paper we focus on data freshness, a performance indicator that is gaining research momentum, especially for IoT applications requiring a timely interaction with the surrounding environment, e.g. to enable real-time decision-making for dynamic contexts where circumstances can change rapidly [11], [12]. In this context, age of information (AoI) is often used, capturing the time elapsed since the generation of the last successfully processed data, and describing how "fresh" is the result of the computational task [13], [14]. Alternatively, various generalizations can be used, considering a broader "penalty" based on the difference between the data generation instant and its ultimately availability for usage after being processed [15], [16].

Starting from these remarks, we analyze a layered computing architecture where data can be either processed *locally* or offloaded to a *remote* server [17]. As argued above, neither choice is inherently better, as the balance depends on the system conditions and the network congestion. Remote processing is accurate, yet incurs a round-trip communication delay, which can be worsened by the network workload of devices querying the server. Conversely, local processing is faster and not affected by congestion, but may have to be repeated to achieve satisfactory results, worsening data freshness. We also explore how the staleness of the available knowledge can be

A. Munari acknowledges the financial support 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.

represented through different penalty metrics related to AoI.

Through an analysis based on the signal-flow graph (SFG) methodology [18], we derive closed form expressions that quantify this tradeoff, and discuss the most appropriate choice between local and remote processing of data. We show how the best network setup lies in a careful balance of the two options, to properly exploit the remote server without congesting it. Thus, the development of tailored offloading strategies is key for timely information processing [19].

II. Related Work

The investigation of choices surrounding computation offloading is a well-studied problem in the literature, often seen as the routing of requests through different paths [8], [20]. The objective of this choice may be related to minimizing the task execution times [7], latency [21], limiting the energy consumption [22] or addressing the quality of experience in multimedia flows [11]. Additionally, information protection can be taken into account, through security techniques such as distributed ledgers [3], [23].

Regardless of the objective, a common issue is that centralized offloading can be derived only after collecting information from every agent, which not only is computationally prohibitive, but also increases the network congestion. Thus, distributed policies are often considered [9] that, while suboptimal, can be made efficient through proper incentives.

Insofar, the issue of local processing versus remote computational offloading has been addressed for latency optimization or task completion maximization. Information freshness is rarely considered, despite being a critical factor in many IoT environments, where the timeliness of data used for decision-making is essential. One of such use cases is the control of autonomous unmanned aerial vehicles, as seen in [24], which explores AoI minimization within an air-ground integrated multi-access edge computing system. This is solved as a stochastic game, further transformed into a single-agent Markov decision process. Another scenario, the Internet of medical things, is considered in [2], with a similar focus on remote vs. local processing, declined as offloading to a central server or processing data inside a wireless body area network to avoid congestion. A decentralized solution is found through a non-cooperative game played by rational agents.

A more general perspective is adopted in [16], where an edge computing architecture is mathematically framed as a queueing system, keeping into account the peak AoI as the performance indicator, so that violation of an age threshold implies staleness of data. Conversely, [25] still considers a queueing approach and focuses on the tradeoff between transmission and processing, but with a slightly different approach than ours. In that paper, edge nodes can choose how much computation to perform, and leave the residual unprocessed data to the remote server. Through a stationary distribution analysis, closed form expressions for AoI and peak AoI are obtained, showing how they can be optimized.

Finally, [17] shares many similarities with our approach, considering local and remote processing, as well as a mixture thereof. However, the approach is again based on queueing theory and a zero-waiting policy is adopted, without considering different reliabilities of the local processing as opposed to the remote offloading.

III. System Model and Preliminaries

A. Notation

We denote a discrete random variable (r.v.) and its realization by upper- and lower-case letters, respectively, e.g. X and x. The probability mass function (PMF) of a r.v. X is $p(x)$, whereas $G_X(u)$ is its *probability generating function*

$$G_X(u) = \mathbb{E}\left[u^X\right] = \sum\nolimits_x u^x p(x).$$

In turn, the probability generating function allows to derive the k-th order moment of the r.v. X as [26]

$$\mathbb{E}\left[X^k\right] = \sum_{j=1}^{k} \left\{ {k \atop j} \right\} \frac{d^j G_X}{du^j}\bigg|_{u=1} \tag{1}$$

where $\left\{ {k \atop j} \right\}$ is the Stirling number of the second kind, i.e.

$$\left\{ {k \atop j} \right\} = \frac{1}{j!}\sum_{\ell=0}^{j}(-1)^{j-\ell}\binom{j}{\ell}\ell^k.$$

Finally, for discrete time Markov chains, we write the transition probability between states i and j as $q_{i,j}$, and the stationary probability of being in state i as π_i.

B. System model

We consider a set of n independent agents (or nodes). Each agent is interested in performing a series of computational tasks, aiming to minimize a cost or penalty function. Tasks can either be performed locally or offloaded, leaning on the support of a remote server. Moreover, each task may or may not be successful, which reflects on the computed penalty.

To model this setup, we consider time to be divided in slots of equal duration, and all agents to be slot-synchronized. In each slot, a node, when idle, chooses among three options: i) it performs a task locally (with probability p_ℓ); ii) it requests remote completion of a task (with probability p_r); or, iii) it does not perform any action (with probability $1 - \mathsf{p}_\ell - \mathsf{p}_r$).

A local task is completed immediately, i.e., within the slot it is initiated by the agent, yet is only successful with probability $\alpha < 1$. Conversely, a remote task is always successful, but may require longer to be processed. This does not necessarily mean that edge/cloud servers do not fail (e.g., because of congestion), but whenever they do, the request is forwarded to another available server until it is eventually successful, so individual failures are just represented with the tail of the delay distribution.

Thus, when a remote task is initiated, the agent enters a waiting state until the outcome is delivered from the server. The time-to-response of a remote request is modeled as a geometric random variable W of parameter δ, with average

$$\mathbb{E}\left[W\right] = 1/\delta. \tag{2}$$

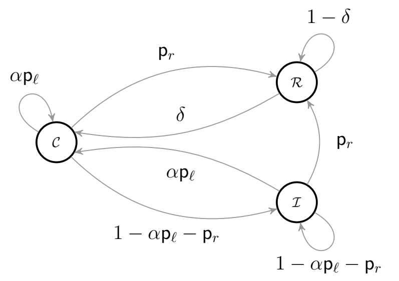

Fig. 1. Markov chain describing the behavior of an agent.

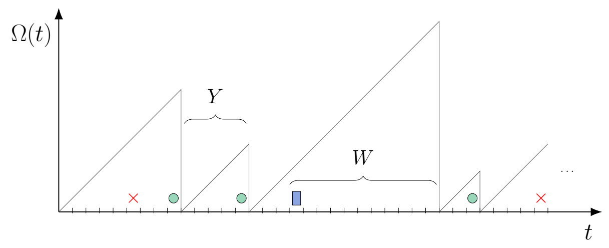

Fig. 2. Sample evolution of penalty $\Omega(t)$, for $m = 1$. Red crosses (×) and green circles (●) denote slots of a local task failing or succeeding, respectively. Blue rectangles (▮) are slots in which a remote task is requested, and will be completed after a delay W, resetting the penalty . The inter-reset time is Y.

As soon as the response from the server is obtained, the agent reverts to idle mode, potentially initiating other tasks.

Accordingly, the behavior of an agent can be described by means of the Markov chain reported in Fig. 1. Here, $\mathcal{I}$ denotes the idle state, whereas the node remains in $\mathcal{R}$ while waiting for the completion of a remote request. Finally, state $\mathcal{C}$ is entered whenever a task is successfully completed. Following this notation, a transition to $\mathcal{R}$ can take place from both $\mathcal{C}$ and $\mathcal{I}$ with probability p_r. On the other hand, once a remote task has been initiated, the agent remains in state $\mathcal{R}$ with probability $1-\delta$ or transitions to $\mathcal{C}$ with probability δ. When idle, the node may also successfully perform a task (probability $\alpha\mathsf{p}_\ell$) and transition to $\mathcal{C}$. Similarly, a transition from this state to $\mathcal{I}$ is possible in case no remote request is triggered and no local task is completed (probability $1-\alpha\mathsf{p}_\ell-\mathsf{p}_r$). The long term distribution for the irreducible, aperiodic chain can readily be derived. In particular, the stationary probability for a node to be in state $\mathcal{R}$ takes the form

$$\pi_{\mathcal{R}} = \frac{\mathsf{p}_r}{\delta+\mathsf{p}_r}.$$

Leaning on this, the frequency of local tasks can be expressed as $(1-\pi_{\mathcal{R}})\,\mathsf{p}_\ell$. To capture the behavior of most practical systems, we consider this quantity to be constrained, due to, e.g., energy, duty cycle, or processor sharing arguments at the node [2], imposing $(1-\pi_{\mathcal{R}})\,\mathsf{p}_\ell \le \varepsilon$. After simple manipulations, the limitation can be expressed as

$$\mathsf{p}_\ell \le \frac{\varepsilon(\delta+\mathsf{p}_r)}{\delta}. \quad (3)$$

Finally, consider the remote processing of tasks. If all agents contend for the same remote server, which implements a processor sharing policy and equally splits the available computational resources [9], the average time needed to complete a task is proportional to the overall intensity of amount of requests. We model this by having

$$\mathbb{E}[W] = \mathsf{T} + (\mathsf{n}-1)\lambda(\mathsf{n},\mathsf{p}_r) \quad (4)$$

where T is the average time for an agent to get a response from the server without any contention, whereas $\lambda(\mathsf{n},\mathsf{p}_r)$ is the average number of remote requests per slot generated by each of the remaining $\mathsf{n}-1$ nodes. Recalling that a poll to the server is generated with probability p_r only when the agent is not waiting for a response, we get $\lambda(\mathsf{n},\mathsf{p}_r) = \pi_{\mathcal{R}}\,\mathsf{p}_r$. From this standpoint, we remark the non-trivial relation between the frequency of requests and the remote service time. Indeed, the longer a node has to wait for a server response (i.e., the larger $\pi_{\mathcal{R}}$), the lower the rate with which it generates further polls to the server. This is characterized by plugging (2) into (4) and solving for δ, to obtain

$$\delta(\mathsf{n},\mathsf{p}_r) = \frac{1-\mathsf{p}_r\mathsf{T}+\sqrt{1+2\mathsf{p}_r\mathsf{T}+\mathsf{p}_r^2(\mathsf{T}^2+4\mathsf{n}-4)}}{2(\mathsf{T}+(\mathsf{n}-1)\mathsf{p}_r)} \quad (5)$$

where the the dependency of the service time on the network cardinality and the remote request probability is highlighted.

The performance of an agent is gauged over time through the penalty function

$$\Omega_m(t) := (t-\tau(t))^m$$

where $\tau(t)$ is the instant of successful completion of the last task (either locally or remotely) for the node as of time t. The metric captures the *timeliness* of the processing, being reset to zero as soon as the successful output of a computation task is available to the agent, and growing as time elapses without further updates. A sample of realization of the stochastic process $\Omega(t)$ is reported in Fig. 2 for $m = 1$ (i.e., a linear penalty).[1] Incidentally, the metric is akin to AoI [15]. In the following, we will focus on the steady-state of the process, which is shown to be ergodic, and consider the *average penalty*

$$\bar{\Omega}_m = \mathbb{E}[\Omega_m(t)]. \quad (6)$$

C. Signal Flow Graphs

To characterize $\bar{\Omega}_m$, we resort to tools borrowed from the study of SFGs. In the remainder, we provide a minimal introduction to the concepts that will be used, referring the interested reader to, e.g., [27], for further details.

An SFG is a directed, weighted graph, in which each vertex is associated to a variable. Within the graph, the value of the variable linked to vertex $\mathcal{A}$ is the weighted sum of all variables associated to vertices with an outgoing edge towards $\mathcal{A}$, each multiplied by the weight of the corresponding edge. Accordingly, an SFG provides a graphical representation of a system of linear equations. One may be interested in expressing one variable as a function of a single other one,

[1]We take slots as the basic time units. Depending on the application, the time to complete a task may vary [21], [22]. Yet, what matters is the ratio of the times to perform an action locally or remotely. Resorting to slots allows us to abstract from specific values and extract general tradeoffs [2], [6].

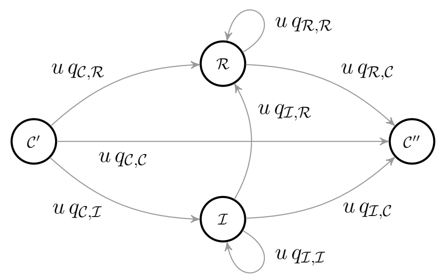
$u\,q_{\mathcal{R},\mathcal{R}}$
$u\,q_{\mathcal{C},\mathcal{R}}$
$\mathcal{R}$
$u\,q_{\mathcal{R},\mathcal{C}}$
$u\,q_{\mathcal{I},\mathcal{R}}$
$\mathcal{C}'$
$\mathcal{C}''$
$u\,q_{\mathcal{C},\mathcal{C}}$
$u\,q_{\mathcal{C},\mathcal{I}}$
$\mathcal{I}$
$u\,q_{\mathcal{I},\mathcal{C}}$
$u\,q_{\mathcal{I},\mathcal{I}}$

Fig. 3. Signal flow graph corresponding to the Markov chain of Fig. 1.

obtaining the so-called *transfer function* between the two. Although the problem can be tackled with several techniques, e.g., substitution within the corresponding linear system, a method known as Mason's gain formula allows to derive the solution by graphical inspection of the SFG. Specifically, the transfer function can be obtained by identifying direct paths between the two variables as well as loops in the graph, and combining the weights of the involved edges [27].

When dealing with finite state Markov chains, SFGs turn out to be useful for deriving statistics of absorption times. More precisely, given a Markov chain, a corresponding SFG can be obtained by preserving the same set of vertices and edges, and considering as weights the transition probabilities of the original chain, multiplied by a dummy variable u. Following this notation, the transfer function between states $\mathcal{A}$ and $\mathcal{B}$ corresponds to the probability generating function of the time of first visit to state $\mathcal{B}$ when starting from $\mathcal{A}$ [27].

IV. Average Penalty

To capture the performance of different task allocation policies, we start by providing the following result.

Lemma 1: For our system, the average penalty $\bar{\Omega}_m$ is

$$\bar{\Omega}_m = \frac{\mathbb{E}\left[Y^{m+1}\right]}{(m+1)\mathbb{E}[Y]} \qquad (7)$$

where Y is the r.v. denoting *inter-refresh* time, i.e., the time elapsed between two successive resets of the penalty.

Proof: We exploit geometric arguments, akin to, e.g., [14]. Leaning on the ergodicity of the stochastic process $\Omega_m(t)$, proved at the end of the section, (6) is expressed as the long term time average of a realization. Accordingly,

$$\bar{\Omega}_m = \lim_{t\to\infty}\frac{1}{T}\int_0^T \Omega_m(t)\,dt \overset{(a)}{=} \lim_{\ell\to\infty}\frac{1}{\sum_{\ell=1}^{\infty} Y_\ell}\sum_{\ell=1}^{\infty} A_\ell$$
$$\overset{(b)}{=} \lim_{\ell\to\infty}\frac{\ell}{\sum_{\ell=1}^{\infty} Y_\ell}\cdot\frac{1}{\ell}\sum_{\ell=1}^{\infty}\frac{Y^{m+1}}{m+1} \overset{(c)}{=} \frac{\mathbb{E}\left[Y^{m+1}\right]}{(m+1)\mathbb{E}[Y]}.$$

Here, (a) follows by expressing the integral of the penalty function as the sum of the areas A_ℓ below the curve within the ℓ-th inter-refresh period (see Fig. 2), and by computing the normalization factor $1/T$ as the sum of the durations of the corresponding periods Y_ℓ. In turn, (b) expresses the area $A_\ell = \int_0^{Y_\ell} t\; dt = Y^{m+1}/(m+1)$, whereas (c) follows from the ergodicity of the process Y_ℓ. ■

The result in (7) clarifies how the statistical moments of the inter-refresh time suffice to compute the average penalty. To this aim, one can conveniently consider the Markov chain in Fig. 1, and observe how Y corresponds to the time of first visit to state $\mathcal{C}$ (metric reset) when starting from the same. Recalling the discussion of Sec. III-C, the probability generating function of the sought quantity, $G_Y(u)$, can then be promptly computed. Specifically, we introduce in Fig. 3 the SFG corresponding to the Markov chain in Fig. 1. For convenience, the state $\mathcal{C}$ has been split into $\mathcal{C}'$ (with only outgoing edges) and $\mathcal{C}''$ (with only incoming edges), and we recall that u is a dummy variable. In this setting, the transfer function between $\mathcal{C}'$ and $\mathcal{C}''$ can be derived by applying Mason's gain formula [27] to obtain

$$G_Y(u) = \frac{u\alpha\,\mathsf{p}_r + u^2(\delta\mathsf{p}_r - \alpha\mathsf{p}_\ell(1-\delta))}{[1-(1-\delta)u]\,[1-(1-\mathsf{p}_r-\alpha\mathsf{p}_\ell)u]}. \qquad (8)$$

By properly combining the derivatives of $G_Y(u)$, as per (1), the statistical moments of any order for the inter-refresh time can be computed, obtaining $\bar{\Omega}_m$ for any value of the penalty order m. As an example, focus on a linear penalty ($m=1$). In this case, $\bar{\Omega}_1 = \mathbb{E}[Y^2]/(2\mathbb{E}[Y])$, where the first and second order moments follow from (1) as $\mathbb{E}[Y] = G'_Y(1)$, $\mathbb{E}[Y^2] = G''_Y(1) + G'_Y(1)$. Computing the derivatives of (8) leads, after simple manipulations, to the closed form

$$\bar{\Omega}_1 = -\frac{1}{2} + \frac{1}{\delta} - \frac{1}{\delta+\mathsf{p}_r} + \frac{1}{\mathsf{p}_r+\alpha\mathsf{p}_\ell}. \qquad (9)$$

The derivation neatly pinpoints the role played by the system parameters α and δ, as well as of the degrees of freedom of the agent in the choice of p_r and p_ℓ.

We also note that (8) proves the ergodicity of $\Omega_m(t)$. To this aim, consider the embedded Markov chain X_ℓ, tracking the penalty metric at the end of a slot. The (infinite-state) chain can immediately be seen to be aperiodic and irreducible. Moreover, the mean recurrence time of state 0 is, by definition, the mean inter-refresh time of the penalty metric, i.e., $\mathbb{E}[Y]$. Observing from (8) that $G'_Y(1) > 0$, the chain is positive recurrent, admits stationary distribution, and is thus ergodic.

V. Numerical Results

A problem of particular interest consists in finding the optimal probabilities for an agent to perform a task locally or request for remote support, aiming to minimize the average penalty. To get some preliminary insights on these aspects, we start by considering a setup with a single node, i.e. $\mathsf{n}=1$, and focus on a linear penalty ($m=1$). In this case, the average response time of the remote server does not depend on the frequency with which the agent issues requests, so that, according to (4), $\delta = 1/T$ irrespectively of p_r. Furthermore, as confirmed by (9), the agent shall set p_ℓ to the highest possible value that does not violate the activity constraint in (3). We remark that, as the frequency of remote requests increases, the time spent waiting for a response from the server increases. Accordingly, higher values of p_ℓ can be employed whenever

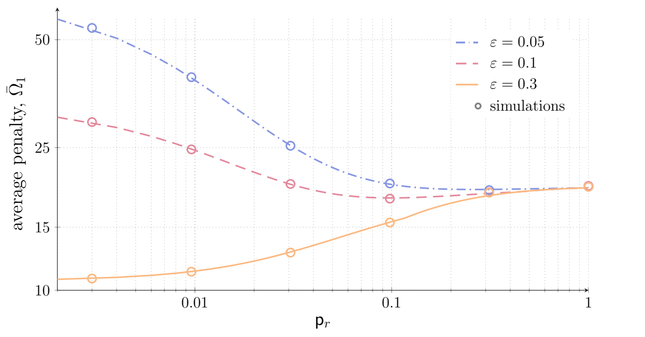

Fig. 4. Average penalty $\bar{\Omega}_1$ vs probability of remote processing, p_r, for a single agent ($\mathsf{n} = 1$), with T=20 and α=0.3, and different activity rates ε. Lines and markers report analytical and simulation results, respectively.

the agent enters (less frequently) the idle state, as captured by the proportionality between p_r and p_ℓ for a fixed δ in (3).[2]

The average penalty attained by changing p_r is reported in Fig. 4 for different values of the activity constraint ε. Markers report the outcome of Montecarlo simulations, verifying the correctness of the reported analysis. In all cases, the average remote service time is set to $\mathsf{T} = 20$, and the success probability of a local task is $\alpha = 0.3$. Consider first the case $\varepsilon = 0.3$ (orange line), in which the agent is allowed to attempt a local task quite frequently. In this setting, the average penalty undergone without resorting to the help of the remote server is lower than what can be achieved by constantly asking for it, discouraging the use of external resources. This can readily be verified by computing $\bar{\Omega}_1$ in (9) for $\mathsf{p}_r = 0$ and $\mathsf{p}_r = 1$, to obtain $-1/2+1/(\alpha\varepsilon)$ and $1/2+T^2/(T+1)$, respectively. The situation is reversed when the agent can perform a local task only sporadically (blue curve, $\varepsilon = 0.05$), as a constant poll to the remote processor ($\mathsf{p}_r = 1$) becomes the most beneficial approach. On the other hand, more interesting insights are offered by the intermediate case under analysis ($\varepsilon = 0.1$, red line). For this configuration, performing only local tasks is still less convenient than resorting to remote support. However, a balance point exists, and the minimum penalty is obtained by selecting an intermediate value p_r^*. Indeed, recalling what discussed earlier, a reduction in p_r allows the agent to operate with higher values of p_ℓ. In the considered configuration, trading off some time spent awaiting a remote answer to attempt a (potentially successful) local task eventually pays off. In the single-agent case, the optimal probability can be explicitly derived in closed form by simply plugging (3) into (9) and nulling the derivative with respect to p_r, to obtain

$$\mathsf{p}_r^* = \frac{1-(\alpha\varepsilon\mathsf{T})^2+\sqrt{1+\alpha\varepsilon\mathsf{T}}}{\alpha\varepsilon\mathsf{T}^2\,(1+\alpha\varepsilon\mathsf{T})}.$$

From this standpoint, the presented closed form analysis provides a useful and practical tool for proper tuning of the agent behavior based on the system parameters.

This is further explored in Fig. 5, reporting the optimal probability of requesting a remote task as a function of

[2]In particular, for any pair $(\varepsilon, \mathsf{T})$ such that $\varepsilon(1-\mathsf{T}) < 1$, we have $\mathsf{p}_\ell = 1$ without violating (3). This only holds for $\mathsf{n} = 1$, as δ does not depend on p_r.

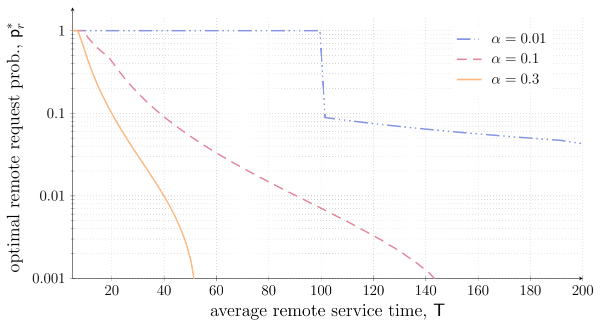

Fig. 5. Optimal remote request probability p_r^* vs mean service time T. Single agent ($\mathsf{n} = 1$), with $\varepsilon = 0.1$, for different local success probabilities α.

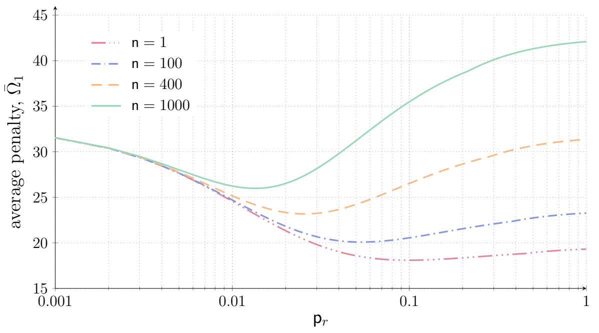

Fig. 6. Average penalty $\bar{\Omega}_1$ vs. probability of requesting a remote task, p_r when n agents contend. In all cases, $\mathsf{T} = 20$, $\alpha = 0.3$ and $\varepsilon = 0.1$.

the average service time T. Different curves denote results for distinct success probabilities of a local task, α. In all cases, the constraint on the local activity is $\varepsilon = 0.1$. The plot clarifies how a quicker response renders the use of the remote server more convenient, eventually leading to all tasks being offloaded remotely. As the probability for a local task to successfully reset the penalty reduces, this configuration dominates for larger values of T. On the other hand, as the average waiting time increases, an optimal balance between local and remote operations shall be aimed for.

Consider now the setting where multiple agents may poll the same remote server. In this case, the average response time increases proportionally to the average number of performed requests per slot, as captured by (4). This triggers some non-trivial effects. On the one hand, the presence of more nodes may lead to more requests, resulting in longer waiting times. On the other hand, it reduces the fraction of idle time, and thus the possibility for an agent to trigger both a local and a remote task. To shed light on the interplay of such factors, we report in Fig. 6 the behavior of the average penalty as a function of p_r when the number of agents n varies. In all cases, $\alpha = 0.3$, $\varepsilon = 0.1$ and $\mathsf{T} = 20$, so that the red curve corresponds to the one already discussed in Fig. 4 for comparison. The plot highlights several interesting take-aways. First, while for low values of n solely resorting to the remote server ($\mathsf{p}_r = 1$) is more convenient than performing tasks fully locally ($\mathsf{p}_r = 0$), the opposite trend emerges for larger network populations, due to resource sharing. In all reported configurations, the optimal solution then lies in a proper balance between the two options.

Secondly, the value of p_r that minimizes the penalty func-

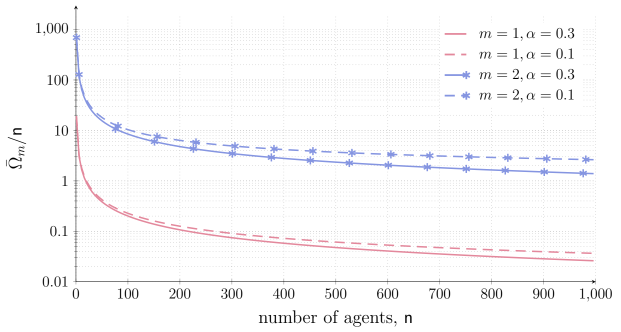

Fig. 7. Average penalty $\bar{\Omega}_m/\mathsf{n}$ vs agent population n, for T=20, ε=0.1. Lines with markers denote penalty order m=2, whereas lines without markers refer to m=2. Dashes differentiate local success probabilities α=$\{0.3, 0.1\}$.

tion reduces as n increases, pinpointing how local operation becomes more important as the contention level grows. From this standpoint, we remark how a closed form for p_r^* becomes cumbersome for $\mathsf{n} > 1$, since δ in (9) is driven by p_r as per (5). Nonetheless, our analysis allows to exactly capture the optimal configurations via straightforward numerical optimizations. Finally, as expected, the minimum attainable penalty increases as the number of considered agents grows.

To conclude, we investigate the impact of the penalty order m. To this aim, Fig. 7 reports the minimum achievable value of $\bar{\Omega}_m$, obtained by optimizing over p_r, vs the number of contending agents n. Lines without and with markers denote the linear penalty considered so far, and a quadratic penalty ($m = 2$), respectively. Similar trends arise, with an expected increase in the undergone penalty for $m = 2$, as longer times spent without refreshing the metric induce a higher cost. Along the same line, lower success rates for local tasks (dashed line) lead to a more severe loss for a quadratic penalty.

VI. Conclusions and Future Work

We presented an analysis of information freshness for data that can be processed either locally (e.g., in the generating devices itself, or an edge server) or globally, i.e., offloaded to the cloud. The outcome is that the optimal choice is non-trivial, and depends on multiple system parameters. Even a single source optimization has different working regimes, according to which the preferable choice results in local processing, remote processing, or even a mixture [17]. Future extensions may include optimization or game theoretic approaches [2], [10], [24], to compare the optimal choice and generalize it to a selfish initiative of distributed nodes.

References

[1] Y. Han, D. Niyato, C. Leung, D. I. Kim, K. Zhu, S. Feng, X. Shen, and C. Miao, "A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 268–284, Jan. 2023.

[2] Z. Ning, P. Dong, X. Wang, X. Hu, L. Guo, B. Hu, Y. Guo, T. Qiu, and R. Y. Kwok, "Mobile edge computing enabled 5G health monitoring for internet of medical things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, Feb. 2020.

[3] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, Apr–Jun 2019.

[4] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comp. Surv.*, vol. 52, no. 1, pp. 1–23, Feb. 2019.

[5] A. Shahraki, A. Taherkordi, Ø. Haugen, and F. Eliassen, "A survey and future directions on clustering: From WSNs to IoT and modern networking paradigms," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2242–2274, 2020.

[6] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proc. IEEE*, vol. 107, no. 8, pp. 1717–1737, Aug. 2019.

[7] H. Gedawy, K. Habak, K. A. Harras, and M. Hamdi, "RAMOS: A resource-aware multi-objective system for edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 8, pp. 2654–2670, Aug. 2021.

[8] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[9] 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.

[10] A. V. Guglielmi, M. Levorato, and L. Badia, "A Bayesian game theoretic approach to task offloading in edge and cloud computing," in *Proc. IEEE ICC Wkshps*, 2018.

[11] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, Mar. 2017.

[12] L. Badia and A. Munari, "A game theoretic approach to age of information in modern random access systems," in *Proc. IEEE Globecom Wkshps*, 2021.

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

[14] A. Munari and L. Badia, "The role of feedback in AoI optimization under limited transmission opportunities," in *Proc. IEEE Globecom*, 2022.

[15] 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, May 2021.

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

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

[18] A. Munari and G. Liva, "Information freshness analysis of slotted ALOHA in Gilbert-Elliot channels," *IEEE Commun. Lett.*, vol. 25, no. 9, pp. 2869–2873, Sep. 2021.

[19] C. Li, "Information processing in Internet of things using big data analytics," *Comp. Commun.*, vol. 160, pp. 718–729, Jul. 2020.

[20] L. Badia, M. Miozzo, M. Rossi, and M. Zorzi, "Routing schemes in heterogeneous wireless networks based on access advertisement and backward utilities for QoS support," *IEEE Commun. Mag.*, vol. 45, no. 2, pp. 67–73, Feb. 2007.

[21] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[22] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2020.

[23] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Survey on multi-access edge computing security and privacy," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1078–1124, Apr–Jun 2021.

[24] X. Chen, C. Wu, T. Chen, Z. Liu, H. Zhang, M. Bennis, H. Liu, and Y. Ji, "Information freshness-aware task offloading in air-ground integrated edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 243–258, Jan. 2021.

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

[26] W. Feller, *An Introduction to Probability Theory and its Applications*. New York: John Wiley & Sons, 1957, vol. 1.

[27] S. J. Mason, "Feedback theory–some properties of signal flow graphs," *Proc. IRE*, vol. 41, no. 9, pp. 1144–1156, 1953.