

MIND-IT: Server Selection in the Edge-Cloud Continuum Under Non-Negligible Path Latencies

Vincenzo Mancuso^{1,4}, *Senior Member, IEEE*, Paolo Castagno², Leonardo Badia³, *Senior Member, IEEE*,
Matteo Sereno², Marco Ajmone Marsan⁴, *Fellow, IEEE*

¹Department of Engineering, University of Palermo, 90133 Palermo, Italy,

²Computer Science Department, University of Turin, 10125 Turin, Italy,

³Dept. Information Engineering, University of Padova, 35131 Padua, Italy,

⁴IMDEA Networks Institute, 28918 Leganes (Madrid), Spain

Abstract—Today’s networks include decentralized intelligence, scattered over multiple computing facilities, for the execution of computation-intensive tasks that are necessary for both network management and service provision to end users. This raises a number of questions regarding task allocation, in terms of both the task placement algorithm, which can be either centralized or distributed, and the resulting task placement in the edge-cloud continuum. Existing analytical models that tackle these issues were conceived for network scenarios that include only cloud computing facilities. In such settings, performance predictions can be derived assuming that all users are located at practically indistinguishable (hence irrelevant for the allocation decision) distances in terms of delay from the cloud server. With the diffusion of edge servers, the role of path latencies becomes relevant, and just considering the response times (i.e., waiting plus service times) at the servers is no longer sufficient. Thus, we propose a model, called Minimum Delay Independent of Traffic and service (MIND-IT), for the optimization of task allocation. MIND-IT considers both the path latencies to reach the edge/cloud servers and their response times, derives explicit performance metrics and provides optimization algorithms. We show, through numerical analysis and real experiments over the Internet, that differences in path latencies to reach edge/cloud servers cannot be neglected. By means of algorithmic game theory, we also study the inefficiency of distributed and selfish task allocation solutions, unveiling important properties such as the fact that their performance loss with respect to centralized optimization is generally limited, except when the system is overloaded, and that worst-case guarantees can be computed with low complexity.

Index Terms—Network servers; Latency; Next generation networking; Optimization; Game Theory; Price of Anarchy.

I. INTRODUCTION

IN today’s softwarized networks, services are implemented through virtual network functions (VNFs) that execute on in-network computing elements. Meeting the latency requirements of real-time services (e.g., those typical of the so-called Tactile Internet [2], [3], [4]) is possible only thanks to a distribution of computing facilities close to the edge of the network [5], [6]. This allows users to access computing

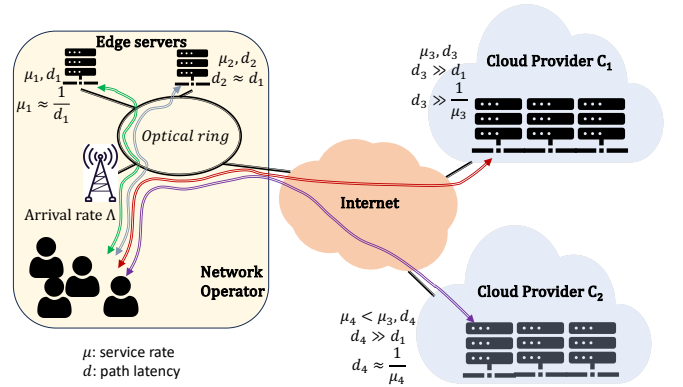


Fig. 1: Reference scenario: users connected through a network operator can offload computations to servers, using either edge or cloud servers. Service requests (tasks) are generated by users at rate λ . Server i is capable of processing μ_i requests per second, and is reached with path latency d_i . Edge-cloud systems are characterized by a large heterogeneity: edge servers are in the Network Operator’s premises, close to users, with small and similar path latency, but can have inhomogeneous speeds; cloud servers are reachable through the Internet, with higher path latency, and are typically faster. In some cases, the delay experienced by using cloud servers is dominated by the path latency (cf. Cloud Provider C_1 in the figure), while in other cases, processing time is comparable to path latency or dominating (cf. Cloud Provider C_2 in the figure). In all cases, reaching heavy server loads can cause long queueing delays that end up dominating the overall service delay. Hence, users must choose which server to use, considering not only path latency and processing capability, but also the congestion status of each server, and aiming at the minimization of the time to receive responses from the selected server.

resources in their proximity, rather than processing data in a remote location in the cloud [7]. For users of latency-critical services, edge computing is a solution under the condition that edge facilities are not overloaded; if they are, the advantage of a short delay to reach the edge server is reversed by its response time (queueing plus service). This means that an individual user accessing a latency-critical service must carefully select the server that can offer the minimum overall latency, including the round trip time to the computing facility and the response time, as shown in Fig. 1.

Analytical characterization of the optimal choice of server is challenging. This is proven by the fact that, in the previous literature on server selection, edge computing, routing, and service placement, a very large fraction of the work models network delay and/or roundtrip time with simple models such as constant latency or hop count [8]. These assumptions are

V. Mancuso and M. Ajmone Marsan have been supported by project TUCAN6-CM (TEC-2024/COM-460), funded by CM (ORDEN 5696/2024). P. Castagno has been supported by UNITA Alliance (through the TALENT starting grant).

A preliminary version of this work was presented at ASMTA 2024 [1].

common even in very recent papers [9] and are certainly legitimate when the computing resources to which tasks are offloaded can be vaguely assumed to be located in nearby positions, i.e., they are all in relative physical proximity and/or they are all generally located in the cloud.

We remark that the aforementioned references mostly consider the problem of gateway placements under homogeneous network latency conditions; however, the routing/path selection problem, which in this case must be extended to a multi-dimensional selection comprising the propagation (fixed) and the queueing (load-variable) components of the delay, is still relatively unexplored.

In addition, we must consider that, as argued in [10] and explicitly quantified with large scale measurements in [11], latency heterogeneity in geographically distributed scenarios is relevant. Indeed, as depicted in Fig. 1, edge/cloud-based reference scenarios significantly depart from the homogeneity assumption, and service delay of each server can be dominated by path latency, processing time or queueing time; however, each server's delay can be dominated by a distinct component of latency, which motivates the need for the unified analytical framework described in this article.

Once we realize that users need to optimize the latency of the entire path (including delay to the server, waiting, and processing at the server, and delay from the server, where the latency within the server depends on load), another question arises about the suitability of a distributed or centralized optimization. When a global perspective is adopted, establishing coordination among multiple users becomes a formidable task and is practically infeasible. The key issue thus becomes whether distributed approaches to server selection in extremely variegated network architectures can still be efficient [12]. Although this problem has received attention in the past [13], [14], [15], we argue that the available mathematical frameworks are inadequate to represent the edge-cloud scenario for the following reasons. First, most investigations consider some models for latency that remain the same among the alternatives [16]. This means that despite accounting for possible differences among multiple service options, the comparisons involve, e.g., fast vs. slow servers, but just with different parameters in the same formula. Instead, service alternatives in today's Internet have characteristics that are different not just in quantitative but also in qualitative terms (for example, some servers may support priorities while others do not).

One particular instance of this aspect is the fixed component of latency, which is largely due to the physical distance between the server and the user. In the edge/cloud context, servers are at extremely different locations on a global scale [17], and the fixed latencies to reach servers are independent of the congestion on the server itself or how fast the server is. We show how neglecting this aspect leads to extremely suboptimal choices. To complicate things further, one can observe that while the service capacity in the absence of congestion and/or fixed latencies due to distance can be possibly known to the user, the final performance also depends on the congestion at the chosen server, which is harder to estimate [18].

The direction that we explore in this work is that of algorithmic game theory [19], considering the service choices

performed by a multitude of atomic selfish non-cooperative users of real-time network services, interested in minimizing their own latency, i.e., the average time elapsing between the issue of a service request and the arrival of the result. In particular, we derive the *price of anarchy* (PoA) [20], [21] of a distributed allocation of computing tasks in scenarios comprising edge and cloud servers at different distances from end users. We remark that we do not *invent* new task allocation control policies, but rather propose how to compute the PoA efficiently with exact algorithms, by comparing the average latency of fully centralized and fully distributed approaches.

Compared to existing results, we take a general approach that can be applicable to any functional relationship describing the achieved latency, under very mild assumptions of positive first and second derivatives. We validate our analytical approach by verifying that it obtains known results in special cases, but at the same time, we argue that the location of servers in the network must be properly taken into account to obtain satisfactory performance. Also, we validate our quantitative results with real-world experiments, by accessing servers across Europe, and show that we are able to match experimental results by using our models, thanks to the generality of our analytical approach.

This article features multiple contributions as follows.

- For the server selection problem, we derive exact analytical results that do not neglect the presence of fixed delays between users and servers, and that are comparable to queueing and processing delays in modern networks that rely on edge infrastructures. Existing models neglect those fixed and heterogeneous delays because they model systems in which latency is dominated by queueing and processing times.
- We evaluate the efficiency of distributed choices by network users in an edge/cloud context, under an extremely general framework that is not found in the previous literature, where only special cases were considered. In particular, we investigate the Nash equilibrium (NE) of the distributed server selections, which is found to be unique. From this perspective, we assert that our work can be regarded as a valuable contribution similar to those emphasized in [22].
- We formulate exact and efficient algorithmic solutions to derive the optimal allocation point and the Nash equilibrium. Our design shows low complexity, which makes it suitable for integration in online network control and optimization mechanisms.
- We compute the PoA as a function of the network load. The worst-case PoA is proven to be the maximum among a finite number of cases. These results enable a fast search for lower performance bounds, enabling a fast decision on whether a central optimization is needed or whether a fully distributed and selfish approach can be enforced with contained performance loss.
- Therefore, we solve a long-standing limitation of existing models and provide tractable numerical tools based on analysis to assess and design server selection strategies in edge/cloud environments. This allows practitioners to move rapidly from theory to implementation in oper-

ational networks, and to make strategic decisions on whether to adopt centralized or distributed service allocation policies, depending on the network configuration.

- We perform an extensive numerical evaluation, which is corroborated by the experimental results.

The remainder of this article is organized as follows. In Section II, we discuss the related literature. The analysis of optimal allocation, NEs, and resulting PoA is presented in Section III under very general assumptions. In Section IV, we prove that our analysis can, among other things, generalize the results obtained in the literature for some special cases. We present numerical evaluations in Section V and conclude the article in Section VI.

II. RELATED WORK

In this paper, we consider a server selection problem for an infinite flow of computing requests (tasks). Ideally, each task should get a computation result in the shortest possible time. Tasks arrive according to a Poisson process and choose one of the available servers in the system, each represented by a queue with a given service rate. The delay resulting from waiting and service is a generic (not necessarily polynomial) function with positive first and second derivatives. A fixed path latency represents the time it takes for a task to reach the selected server.

This problem characterization finds matches in the literature at many levels, which can be loosely classified in these categories: (i) classic studies on routing models and efficient transport optimization, which led to well-known models of PoA in the early 2000s; (ii) extensions to multi-parametric choices, where servers or communication end-points are described by more than one performance indicator (e.g., multiple delay components); and (iii) recent studies in mobile edge computing, where latency optimization is often considered under centralized or cooperative frameworks. In contrast, we assume that decisions can be completely selfish, leading to a PoA characterization.

A. Classic studies on traffic routing/server selection

The problem can be seen as an instance of routing [23]. Research in transportation optimization, spearheaded by Braess [24] and Pigou [25], produced numerous findings that are also applicable to scheduling, routing, and server selection (see [26], [27] for a comprehensive discussion).

Several works align directly with our formulation. In particular, [12] and [16] study multi-server scenarios where customers minimize their mean waiting time and compare selfish strategies against globally optimal ones (see also [28] for a thorough and detailed discussion). Similarly, [29] examines M/G/1 servers under processor-sharing, focusing on the inefficiency of equilibria. Earlier analyses of routing games, such as [30], [31], [32], [33], also contribute to the theoretical foundations of selfish vs. social optimization.

While these works provide closed-form solutions for homogeneous queueing scenarios, they assume service time as the only source of delay. Introducing distinct path latencies per server makes these derivations intractable, leaving open

the question of whether exact analysis remains possible. The breadth of this literature is substantial (e.g., [32] alone has amassed several thousand citations), with numerous variants explored. Our focus here is therefore to position the specific contribution we make relative to these threads.

B. Server selection problem with multiple parameters

Some authors have considered more than one parameter in server selection, though with strong restrictions. For example, [12] extends a simpler analysis to M/G/1 queues with heterogeneous service distributions, but only under a common coefficient of variation. [29], [34], [35] analyze PoA in M/G/1 queues with processor sharing, where the delay depends only on the mean service time. In both cases, the presence of two parameters (mean and variability) reveals the mathematical difficulty of the problem, which remains unsolved for FCFS servers. Stidham, in [36] and [37] proposed approximate solutions in heavy traffic or specific cost-delay tradeoffs, while a survey in [22] reviews these variants.

Other studies further simplify, by assuming network over-provisioning, making transmission delays negligible [38], [13], [14], [15]. This assumption, plausible in early cloud scenarios, is contradicted by today's trend of distributing servers closer to the edge. With growing computational capacity reducing processing delays, network latency has become a significant performance bottleneck [39].

In particular, we show that both socially optimal allocations and Nash equilibria can be computed exactly by means of the MIND-IT algorithms in polynomial time.

C. Server selection optimization in mobile edge computing

More recent work connects server selection to mobile edge computing. For example, [40] optimizes both queueing and access delay via a divide-and-conquer approach, but assumes variable wireless rates rather than fixed path delays and no queueing buffer. Reference [41] incorporates queueing congestion in a Markov decision process, but under a cooperative setting without game-theoretic analysis. In [42], policies for optimally splitting traffic among multiple edge servers are investigated; The authors prove the indexability of the selection and derive an optimal index policy, but the decision criteria are only related to the freshness of the information versus the capacity and the status of the server congestion, and not different latency values due to the location of the server. The authors of [43] study server selection with a fixed cloud delay, but only one offloading alternative and a cooperative model. Similarly, [44] uses game theory for task allocation, focusing on task completion rather than latency.

Complementary studies combine game theory with reinforcement learning. A recent systematic survey by Jain *et al.* [45] reviews game-theoretic and reinforcement learning approaches in multi-agent decision making, mobile edge caching, cyber-physical systems, and related domains. Even though its scope is broader than server selection per se, it consolidates methodologies that are directly relevant to latency-aware formulations and to works such as [46], [47], [48]. Li *et al.* [46] model dynamic edge computing markets with

DRL, optimizing server selection, offloading size, and pricing, without explicitly characterizing fixed path latencies. Liang *et al.* [47] address multi-user offloading under interference and heterogeneous resources, relying on machine learning to overcome complexity, while we derive exact analytical solutions despite incorporating heterogeneous latencies. Xu *et al.* [48] analyze secure edge caching through a Stackelberg game with RL, where delay is only modeled as an access penalty. Our approach could complement such scenarios, since once heterogeneous path delays are derived, our analytical results can enrich their framework with tractable efficiency characterizations.

Overall, these studies confirm the relevance of incorporating latency into server selection models, but they typically rely on approximations, restrictive assumptions, or simplified network settings. This gap is precisely what our contribution addresses.

In recent years, a large body of work has emerged under the umbrella of edge/cloud or continuum computing, including both theoretical models and application-oriented proposals. Despite their heterogeneity, these approaches share some fundamental characteristics: the presence of a large number of heterogeneous servers, their distribution across different network locations, and the need for server selection mechanisms that are effective from an application perspective while remaining simple, scalable, and easily implementable (see, for instance, [49], [50], [51]).

From this standpoint, edge and cloud servers differ not only in their computational capabilities but also in the path latencies experienced by users. The demand for lightweight and scalable server selection strategies naturally motivates the adoption of decentralized, selfish decision-making mechanisms, where no centralized optimizer is assumed, and users independently select the server to which computation tasks are offloaded.

In this context, our work addresses a simple yet fundamental question that has remained largely unanswered: how do selfish server selection strategies perform with respect to globally optimal or socially efficient solutions when both response times and path latencies are explicitly accounted for? By providing exact analytical results that overcome the common assumption of negligible access delay, we characterize the conditions under which selfish choices are optimal or near-optimal, and quantify the inefficiency of decentralized equilibria. In doing so, we solve a long-standing modeling limitation and provide tractable tools to assess and design server selection mechanisms in edge/cloud systems. Moreover, we do this by means of exact and efficient algorithms that we have designed based on the analysis of the problem.

III. MIND-IT — ANALYSIS WITH MINIMUM DELAY INDEPENDENT OF TRAFFIC AND SERVICE

Assume that a group of users are connected to the same slice of an access network, from which n servers can be reached, as shown in Fig. 1 for $n = 3$. Although servers are accessible at different distances, they offer service to the group of users by running computing jobs that users request to implement the desired service, as typical of modern 3GPP networks [52], [53]. The servers are ordered according to the minimum

average delay experienced by a request, which equals the sum of path latency and average service time, excluding queueing delays (which, however, will be considered in the analysis).

$$\forall i, j \in \mathcal{S} = \{1, \dots, n\}, i < j \implies d_i + \frac{1}{\mu_i} \leq d_j + \frac{1}{\mu_j}, \quad (1)$$

where d_i is the two-way delay to reach server i and $1/\mu_i$ is the average service time at the same server. Offset values $d_i \geq 0$ make our study different from existing works, because they may make slow but close servers preferable to fast but far ones. Indeed, all previously published results only hold for the case $d_i = 0$ or, equivalently, for the case in which all d_i are the same for all servers and all users.

In the following, let $\mathbf{p}^* = \{p_i^*\}_{i=1, \dots, n}$ denote a probability vector that expresses the solution of the optimization problem consisting in allocating the load to servers such that the average latency is minimized. In addition, let $\mathbf{p}^\dagger = \{p_i^\dagger\}_{i=1, \dots, n}$ denote the probability vector corresponding to the Nash equilibrium point (NEP) achieved if users select their server selfishly, with a stochastic strategy aiming at minimizing their own expected latency. In this work, we assume that users decide where to send their service requests on a per-request basis; hence, the above-defined probabilities have to be interpreted as per-request allocation probabilities.

A. Optimal load allocation problem

The problem consists in finding the probabilities $\mathbf{p}^* = \{p_i^*\}_{i=1, \dots, n}$ that minimize the following cost function: [54]

$$U(\mathbf{p}) = \sum_{i=1}^n p_i \ell_i(p_i \Lambda), \quad (2)$$

where $\ell_i(x) > 0$ is the average latency of server i when it receives traffic with intensity $0 \leq x < \mu_i$, and μ_i is the capacity of that server, which cannot be reached without driving the queue into instability (i.e., the latency diverges as x approaches μ_i). Λ is the aggregate offered traffic.

Function $\ell_i(x)$ accounts for the path latency to reach the server (d_i), its waiting and service times, that depend on the service capacity (μ_i), and may also account for additional parameters (e.g., the variance of the service time). The minimum average latency must be the sum of distance plus one service interval, i.e., $\ell_i(0) = d_i + 1/\mu_i$.

Since we consider systems subject to congestion, $\ell_i(x)$ monotonically increases with x because of waiting times. Also, $\ell_i(x)$ is a convex function, as commonly observed for the latency of a queueing system. Hence, $\ell_i(x)$ and its first and second derivatives are positive functions.

The above problem is solved subject to the following simple and self-explanatory constraints:

$$p_i \geq 0, \quad \forall i; \quad (3)$$

$$p_i \Lambda \leq \mu_i, \quad \forall i; \quad (4)$$

$$\sum_i p_i = 1. \quad (5)$$

The problem is strictly convex because $U(\mathbf{p})$ has a (diagonal) positive-definite Hessian in the decision variables p_i .

Thus, the solution is unique. The result is trivial due to the fact that

$$\frac{\partial}{\partial p_i} \frac{\partial U(\mathbf{p})}{\partial p_j} = 0, \quad \forall i \neq j; \quad (6)$$

$$\frac{\partial^2 U(\mathbf{p})}{\partial p_i^2} = \Lambda^2 \frac{d^2 \ell_i(x)}{dx^2} \Big|_{x=p_i \Lambda} > 0; \quad (7)$$

therefore, the Hessian matrix is diagonal, and its determinant is the product of positive factors. Since the solution is unique, any valid optimization method will find the same solution. In what follows, we derive an optimization methodology that relies on the Lagrangian-based method to reach such solution, either analytically or numerically, depending on whether latency functions are analytically invertible.

The Lagrangian, with α_i , β_i , and γ multipliers, is

$$\begin{aligned} \mathcal{L} = & \sum_i p_i \ell_i(p_i \Lambda) - \sum_i \alpha_i p_i + \sum_i \beta_i (p_i \Lambda - \mu_i) \\ & + \gamma \left(1 - \sum_i p_i\right). \end{aligned} \quad (8)$$

As customary in this kind of problems [20], [55] we impose the following Karush-Kuhn-Tucker (KKT) conditions to hold [56], which give necessary conditions for a vector $\mathbf{p}^*$ to be a solution of the problem:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \ell_i(p_i \Lambda) + p_i \Lambda \ell'_i(p_i \Lambda) - \alpha_i + \Lambda \beta_i - \gamma = 0, \quad \forall i; \quad (9)$$

$$\alpha_i p_i = 0, \quad (10)$$

$$\beta_i (p_i \Lambda - \mu_i) = 0, \quad \forall i; \quad (11)$$

$$\gamma \left(1 - \sum_i p_i\right) = 0. \quad (12)$$

Notice that $\ell_i(x) + p_i \Lambda \ell'_i(x)$ is the derivative of the weighted latency of the i -th server, with weight p_i , computed with respect to p_i . Such a quantity has the dimension of a latency and is always non-smaller than the latency observed at the i -th server. We will refer to it as to the *augmented latency* of server i . Multipliers α_i and β_i must be non-negative, while γ can take any real value (because it is associated with an equality constraint).

To identify which servers are used, consider the properties of the KKT conditions at the optimal point $\mathbf{p}^*$.

Case $p_j^* = 0$. If the optimal solution consists of assigning zero load to server j , then the KKT conditions imply that

$$\alpha_j \geq 0, \quad \beta_j = 0, \quad \ell_j(0) - \alpha_j - \gamma = 0; \quad (13)$$

$$\Rightarrow \ell_j(0) \geq \gamma, \quad (14)$$

i.e., servers for which the latency computed at empty queue satisfies the above condition are inactive. However, with no wait, a service request is served in $\ell_j(0) = d_j + 1/\mu_j$, for any function ℓ_j . Therefore, if a server j is not active, then any server $i > j$, for which we know that $d_i + 1/\mu_i \geq d_j + 1/\mu_j$, is also inactive.

Being the set of servers ordered, there exists an integer $j^* \leq n$ such that all servers from 1 to j^* receive some traffic, while the remaining servers remain inactive. Finding the value of j^* is part of the solution of the optimization problem and we

will soon show how to find it. Next, we analyze the KKT conditions for servers that receive non-zero traffic.

Case $p_j^* > 0$. For all servers $j = 1, \dots, j^*$, which, as observed in the analysis of the previous case, and as a consequence of the ordering of servers established in (1), must receive non-zero load in the optimal configuration, the following conditions must hold:

$$\alpha_j = \beta_j = 0, \quad (15)$$

$$\gamma = \ell_j(p_j^* \Lambda) + p_j^* \Lambda \ell'_j(p_j^* \Lambda). \quad (16)$$

Therefore, γ , which is larger than the largest latency of any active server, and thus we will refer to it as to the *optimal augmented latency*, must be positive and, since $p_j^* > 0$, must also be greater than any $\ell_j(0)$ for any active server, which leads to

$$\gamma > d_{j^*} + \frac{1}{\mu_{j^*}}. \quad (17)$$

Since there is only one multiplier γ that must obey all conditions expressed in (14) and (16), the above result for servers with loads ($p_j^* > 0$) is coherent with (14), which had been derived for servers without load ($p_j^* = 0$). In particular, conditions (14) and (17) are complementary: at the optimum, γ must be below or equal to the minimum latency of a server if the server is not loaded, and above its minimum latency if the server must receive non-zero load. The result in (16) also proves that γ is monotonically increasing, because it is a sum of monotonic functions, hence it is also an invertible function. Let $h_j(x) = \ell_j(x) + x \ell'_j(x)$ indicate such a function, which has to be computed in $x = p_j^* \Lambda$ at the optimum. Thus, we have

$$p_j^* = \frac{1}{\Lambda} h_j^{-1}(\gamma), \quad (18)$$

where h_j^{-1} indicates the inverse function. Considering that the latency functions ℓ_j and the corresponding derivatives are defined only for offered traffic falling between 0 and μ_j , the above function $h_j^{-1}(\gamma)$ can only take values in the interval $[1, \mu_j)$. Hence, h_j^{-1}/Λ cannot be larger than 1. Indeed, the sum of all non-zero probabilities must be 1, so:

$$1 = \sum_{j=1}^{j^*} p_j^* = \frac{1}{\Lambda} \sum_{j=1}^{j^*} h_j^{-1}(\gamma). \quad (19)$$

Since the sum of increasing functions is also increasing, the R.H.S of (19) is invertible. In addition, the R.H.S. must be a number between 0 and $\sum_{j=1}^{j^*} \mu_j / \Lambda \geq 1$, so that all offered traffic can be served. Thus, there exists a unique value of γ that satisfies (19).

Due to conditions (14) computed on j^*+1 and (17) for j^* , γ must be found in a specific interval:

$$\gamma \in \left(d_{j^*} + \frac{1}{\mu_{j^*}}, d_{j^*+1} + \frac{1}{\mu_{j^*+1}} \right], \quad (20)$$

where $d_{j^*+1} + \frac{1}{\mu_{j^*+1}}$ has to be taken as infinite if $j^*=n$. Hence, the number of active servers j^* is uniquely identified by the optimal augmented latency γ .

In practice, determining the value of j^* beforehand is key to compute the solution of the problem efficiently. Indeed, once

j^* is known, by inverting the normalization expression (19), one can compute the value of γ , and, in turn, the value of γ can be used in (18) to compute the optimal probabilities p_j^* .

Probabilities p_j^* depend on Λ , although so far we have treated Λ as a constant. If we instead take Λ as a variable, we can see that γ must be a monotonically increasing function of such a variable, because so are all functions ℓ_j and the associated derivatives. Thus, as traffic Λ increases, the optimal augmented latency γ also increases, so that, as inferred from (20), servers are progressively activated following the order defined by (1). In particular, if server j is activated at $\Lambda_j^{(\text{Opt})}$, with that load p_j^* is still 0 and so, according to (16), $\gamma = \ell_j(0) = d_j + \frac{1}{\mu_j}$. Hence, at $\Lambda_j^{(\text{Opt})}$ we have:

$$p_i^* \Lambda_j^{(\text{Opt})} = h_i^{-1} \left(d_j + \frac{1}{\mu_j} \right), \quad \forall i \leq j, \quad (21)$$

and by summing over servers with non-zero probability p_i^* , i.e., from 1 to $j-1$, we obtain the traffic threshold:

$$\Lambda_j^{(\text{Opt})} = \sum_{i=1}^{j-1} h_i^{-1} \left(d_j + \frac{1}{\mu_j} \right). \quad (22)$$

A simple comparison between the activation thresholds $\Lambda_j^{(\text{Opt})}$ and the offered traffic Λ reveals the value of j^* . Of course, $\Lambda_1^{(\text{Opt})} = 0$ since at least the first server has to be active as soon as non-zero traffic is offered to the network.

Considering the analysis above, the optimization of the formulated problem can be done with an algorithm that requires the numerical inversion of a few monotonic functions, which can be done in closed form only in specific cases, whereas in the general case it requires numerical inversion with lightweight algorithms as the dichotomous search algorithm. In particular, the following MIND-IT (Opt) (Minimum Delay Independent of Traffic and Service - Optimum) algorithm finds the optimal probability vector $\mathbf{p}^*$ at which the average system latency is minimized.

MIND-IT (Opt)

Input: Sorted servers $j \in \{1, \dots, n\}$, Λ .

Step 1: Compute activation thresholds $\Lambda_j^{(\text{Opt})}$ with (22).

Step 2: Compute j^* by comparing Λ to the thresholds.

Step 3: Set $p_j^* = 0, \forall j > j^*$ and compute γ inverting (19).

Step 4: Compute $p_j^*, \forall j \leq j^*$, with (18).

Output: $p_j^*, j \in \{1, \dots, n\}$.

Theorem 1. *The MIND-IT (Opt) algorithm is exact and polynomial.*

Proof. The algorithm finds the optimum because it reaches a feasible solution $\mathbf{p}^*$ for which all KKT conditions are satisfied. The algorithm finds the activation thresholds, which requires computing a number of terms quadratic in n , and inverts (19). The latter consists in finding the zero of a monotonically increasing function with up to $n+1$ invertible terms, which can be done, e.g., with a dichotomous search on each term, with complexity $\mathcal{O}(n \log r)$, where r is the target numerical resolution. The computation complexity of each of the n probabilities, with (18), is the complexity of one inversion, i.e., $\mathcal{O}(\log r)$. Therefore, the overall complexity

of the MIND-IT (Opt) algorithm is $\mathcal{O}(n^2 + n \log r)$ and is polynomial for any value of r . $\square$

Remark 1. Since MIND-IT (Opt) obtains the optimal server selection when the latency path is taken into account, its results can be related to the optimization approaches presented in the literature [40], [41]. These references do not consider queueing delays at the server and instead apply delay management on the uplink channel (also, they split the optimization into sub-problems, hence the solution is just suboptimal). However, if MIND-IT (Opt) is evaluated in the same context, its evaluation would coincide.

B. The Nash equilibrium point (NEP)

We can consider a distributed version for the minimization of (2), in which each user sending traffic minimizes her latency with a probabilistic strategy $\mathbf{p}_u^\dagger$.

With sorted servers, it is easy to see that a user sends traffic to j only if the average observed latency is above the minimum possible latency at that server, $\ell_j(0)$.

An NEP exists and is unique due to the fact that $U(\mathbf{p})$ is strictly convex [12]. At the NEP, all users have the same strategy $\mathbf{p}_u^\dagger = \mathbf{p}^\dagger$ characterized by the fact that all servers that receive traffic experience the same average latency τ , so that no user has any incentive to deviate from strategy $\mathbf{p}^\dagger$:

$$\ell_j(p_j^\dagger) = \tau, \quad p_j^\dagger > 0, \quad \forall j \leq j^\dagger, \quad (23)$$

where $j^\dagger$ is the number of servers used ($p_j^\dagger = 0$ for other servers $j > j^\dagger$).

As Λ increases, τ has to increase as well because, to maintain the same latency at all active servers, the incremental arrival rate has to be distributed over all of them and no server can receive less traffic than before the increase. Notice also that, for a used server ($p_j^\dagger > 0$) the latency lower bound is

$$\tau > \ell_j(0) = d_j + \frac{1}{\mu_j}, \quad \forall j \leq j^\dagger. \quad (24)$$

Expressing probabilities vs. the average latency τ and normalizing, we obtain:

$$p_j^\dagger = \frac{1}{\Lambda} \ell_j^{-1}(\tau), \quad \forall j \leq j^\dagger; \quad (25)$$

$$1 = \frac{1}{\Lambda} \sum_{j=1}^{j^\dagger} \ell_j^{-1}(\tau), \quad (26)$$

where ℓ_j^{-1} indicates the inverse function of ℓ_j .

Inverting the above expression yields the value of $\tau > 0$, which must be unique since the R.H.S. of (26) is a monotonic increasing function upper-bounded by $(1/\Lambda) \sum_{j=1}^{j^\dagger} \mu_j \geq 1$, as the traffic offered to each server cannot exceed the server capacity (i.e., $\ell_j^{-1}(\tau) \leq \mu_j$) and Λ cannot exceed the aggregate capacity of active servers. Notice that τ must be $d_{j^\dagger} + \frac{1}{\mu_{j^\dagger}}$ when server $j^\dagger$ is switched on, and as soon as $p_{j^\dagger}^\dagger$ becomes larger than zero, τ must be included in the following interval:

$$\tau \in \left(d_{j^\dagger} + \frac{1}{\mu_{j^\dagger}}, d_{j^\dagger+1} + \frac{1}{\mu_{j^\dagger+1}} \right), \quad \text{at } p_{j^\dagger}^\dagger > 0, \quad (27)$$

$$\text{and } p_j^\dagger = 0, \quad \forall j > j^\dagger. \quad (28)$$

This implies that the average latency τ also has to increase when a new server is activated due to an increase in Λ . We conclude that τ always increases with Λ and the servers are progressively activated following the sorting order (1), as Λ increases.

Similarly to what we have seen for the case of γ , the above equations and considerations show that τ is the only real root of (26), leading to a feasible strategy vector of probabilities $\mathbf{p}^\dagger$. Thus, we can conclude that there exists a unique NEP in which all servers have the same latency.

Since at activation of j the value of $p_j^\dagger$ is 0 and therefore $\tau = d_j + \frac{1}{\mu_j}$, and the sum of non-zero probabilities must be equal to 1, the threshold $\Lambda_j^{(\text{NEP})}$ can be computed similarly to the threshold in the optimal case (of course, $\Lambda_1^{(\text{NEP})} = 0$):

$$\Lambda_j^{(\text{NEP})} = \sum_{i=1}^{j-1} \ell_j^{-1} \left(d_i + \frac{1}{\mu_i} \right). \quad (29)$$

In the NEP calculation, function ℓ_j plays the role that h_j plays in the calculation of the optimum. Since both functions are positive and monotonic increasing, the algorithm for finding the NEP is very similar to the one for the optimum, as shown in the following MIND-IT (NEP) (Minimum Delay Independent of Traffic and Service – Nash equilibrium point) algorithm.

MIND-IT (NEP)

Input: Sorted servers $j \in \{1, \dots, n\}$, Λ .

Step 1: Compute thresholds $\Lambda_j^{(\text{NEP})}$ with (29).

Step 2: Compute $j^\dagger$ by comparing Λ to the thresholds.

Step 3: Set $p_j^\dagger = 0, \forall j > j^\dagger$ and compute τ inverting (26).

Step 4: Compute $p_j^\dagger, \forall j \leq j^\dagger$, with (25).

Output: $p_j^\dagger, j \in \{1, \dots, n\}$.

Theorem 2. The MIND-IT (NEP) calculation algorithm is polynomial.

Proof. The proof proceeds as in Theorem 1 and finds complexity $\mathcal{O}(n^2 + n \log r)$ for any inversion precision r . $\square$

Remark 2. Similar to Remark 1, the results of MIND-IT (NEP) would coincide with those of other approaches in the literature such as [43], [44] if applied to their same context. However, our approach generalizes over them, and can be directly compared with the exact optimum of MIND-IT (Opt), unlike them [44].

C. Properties

Here we derive some remarkable properties of the problem at hand, which will help in comparing NEP and optimal solutions and their behavior as the offered traffic changes.

Lemma 1. The optimal augmented latency γ is an upper bound of the optimal latency. The distance of the bound is proportional to the derivative of optimal latency, which is a continuous function of traffic Λ :

$$\frac{d}{d\Lambda} U(\mathbf{p}^*) = \frac{\gamma - U(\mathbf{p}^*)}{\Lambda} = \sum_{j=1}^{j^*} (p_j^*)^2 \ell_j'(p_j^* \Lambda) > 0. \quad (30)$$

Proof. We proceed by re-writing the expression of the optimal augmented latency γ and then by calculating the derivative of γ in Λ from its definition:

$$\begin{aligned} \gamma &= \sum_{j=1}^{j^*} p_j^* \gamma = \sum_{j=1}^{j^*} p_j^* (\ell_j(p_j^* \Lambda) + p_j^* \Lambda \ell_j'(p_j^* \Lambda)) \\ &= U(\mathbf{p}^*) + \Lambda \sum_{j=1}^{j^*} (p_j^*)^2 \ell_j'(p_j^* \Lambda) \geq U(\mathbf{p}^*). \end{aligned} \quad (31)$$

The above results show that γ is an upper bound of the optimum latency.

Moreover, we have the following expression for the derivative of optimal utility:

$$\begin{aligned} \frac{dU(\mathbf{p}^*)}{d\Lambda} &= \sum_{j=1}^{j^*} \left(\frac{dp_j^*}{d\Lambda} \ell_j(p_j^* \Lambda) + p_j^* \frac{d(p_j^* \Lambda)}{d\Lambda} \ell_j'(p_j^* \Lambda) \right) \\ &= \gamma \sum_{j=1}^{j^*} \frac{dp_j^*}{d\Lambda} + \sum_{j=1}^{j^*} (p_j^*)^2 \ell_j'(p_j^* \Lambda) \sum_{j=1}^{j^*} (p_j^*)^2, \end{aligned} \quad (32)$$

where, because of the linearity of the derivation operator, the following result holds:

$$\sum_{j=1}^{j^*} \frac{dp_j^*}{d\Lambda} = \frac{d}{d\Lambda} \sum_{j=1}^{j^*} p_j^* = \frac{d}{d\Lambda} 1 = 0. \quad (33)$$

Considering the above expressions for γ and $\frac{dU(\mathbf{p}^*)}{d\Lambda}$, we can write $\frac{dU(\mathbf{p}^*)}{d\Lambda} = \frac{\gamma - U(\mathbf{p}^*)}{\Lambda}$, which is continuous at any point Λ because both γ and $U(\mathbf{p}^*)$ are continuous functions of Λ . $\square$

Beside being an upper bound for the optimal latency, it is useful to see that γ is also larger than the latency at the NEP, as expressed in the following lemma.

Lemma 2. The second derivative of the optimal latency is

$$\frac{d^2 U(\mathbf{p}^*)}{d\Lambda^2} = \frac{\gamma'}{\Lambda} - 2 \frac{\gamma - U(\mathbf{p}^*)}{\Lambda^2}, \quad \forall \Lambda. \quad (34)$$

Proof. The result can be obtained by taking the derivative of (30) and noticing that the derivative of γ is invariant with respect to the server j and can be written as follows:

$$\gamma' = (2\ell_j'(p_j^* \Lambda) + p_j^* \Lambda \ell_j''(p_j^* \Lambda)) \frac{d(p_j^* \Lambda)}{d\Lambda} > 0, \quad \forall j \leq j^*. \quad (35)$$

Therefore, we can write:

$$\begin{aligned} \frac{d^2 U(\mathbf{p}^*)}{d\Lambda^2} &= \frac{d}{d\Lambda} \frac{1}{\Lambda^2} \sum_{j=1}^{j^*} (p_j^* \Lambda)^2 \ell_j'(p_j^* \Lambda) \\ &= \frac{1}{\Lambda^2} \sum_{j=1}^{j^*} p_j^* \Lambda \gamma' - \frac{2}{\Lambda} \frac{dU(\mathbf{p}^*)}{d\Lambda} = \frac{\gamma' - 2 \frac{dU(\mathbf{p}^*)}{d\Lambda}}{\Lambda}. \end{aligned} \quad (36)$$

Considering Lemma (1), the above expression simplifies into (34). $\square$

The second derivative of the optimal latency does not have to be positive. Indeed, we have observed negative values for $\frac{d^2 U(\mathbf{p}^*)}{d\Lambda^2}$ as the load increases immediately after the activation of a new server. This occurs because the activation of a new

server, which absorbs some traffic, slows the growth rate of the latency. So, in general, the optimal latency, seen as a function of Λ , is increasing but not convex.

Lemma 3. $\Lambda > 0 \implies \gamma > \tau$.

Proof. If $\Lambda > 0$ we have the NEP at $\mathbf{p}^\dagger$, with average latency τ , and the optimum (i.e., the minimum latency) at $\mathbf{p}^*$, with average latency no greater than τ . If $\mathbf{p}^\dagger \neq \mathbf{p}^*$, at the optimum at least one server receives more traffic than at the NEP. Now, since at $\mathbf{p}^\dagger$ all latency values of used servers $\ell_j(p_j^\dagger \Lambda)$ must be equal to τ , the latency of servers that are assigned a higher probability at the optimum than at the NEP must be higher than τ . At the optimum, the server with the highest latency is therefore a server that received more traffic than at the NEP (this also applies to servers that were not active at the NEP), i.e., if $j' = \arg \max_j \ell_j(p_j^* \Lambda)$ denotes the server with the highest latency at the optimum, then $p_{j'}^* > p_{j'}^\dagger$, and hence $\ell_{j'}(p_{j'}^* \Lambda) > \ell_{j'}(p_{j'}^\dagger \Lambda) = \tau$. However, $\gamma > \ell_j(p_j^* \Lambda)$ for all active servers, so γ is greater than the largest latency of used servers, which must be greater than τ .

The proof is complete by considering the remaining case $\mathbf{p}^* = \mathbf{p}^\dagger$, for which $\tau = \ell_j(p_j^\dagger \Lambda) = \ell_j(p_j^* \Lambda)$ on any active server. However, $\gamma > \ell_j(p_j^* \Lambda)$ at any j for which $p_j^* > 0$, then $\gamma > \tau$. $\square$

Lemma 4. $\Lambda_j^{(\text{NEP})} \geq \Lambda_j^{(\text{Opt})}, \forall j \in \{1, \dots, n\}$.

Proof. For $j = 1$, both thresholds at the optimum and NEP are 0. If there are multiple servers with the same value of $d_j + 1/\mu_j = d_1 + 1/\mu_1$, all of them have to comply with the same activation conditions, so that they are all activated as soon as non-zero traffic is offered, together with $j = 1$. Hence, their thresholds are all 0 both at the optimum and in the NEP configuration, and the theorem holds for this subset.

For all other servers with $d_j + 1/\mu_j > d_1 + 1/\mu_1$, which must be activated for some positive values of the thresholds, $\gamma|_{\Lambda=\Lambda_j^{(\text{Opt})}} = \ell_j(0)$ and $\tau|_{\Lambda=\Lambda_j^{(\text{NEP})}} = \ell_j(0)$, i.e., they are equal. However, both τ and γ increase with Λ and $\gamma > \tau$ for $\Lambda > 0$ (Lemma 3), so that the above equality can only hold for $\Lambda_j^{(\text{NEP})} > \Lambda_j^{(\text{Opt})}$, and the theorem follows. $\square$

Corollary 1. $\Lambda_j^{(\text{NEP})} = \Lambda_j^{(\text{Opt})} \iff \Lambda_j^{(\text{Opt})} = 0$ and all thresholds are equal to zero if and only if all servers have the same minimum latency, i.e., $d_i + 1/\mu_i$ is the same for all servers.

The latency at the NEP is an increasing function of the offered traffic as the optimal latency, and is also convex.

Lemma 5. Quantities $p_j^\dagger \Lambda$ and $p_j^* \Lambda$ are non-decreasing functions of Λ .

Proof. It is easy to see that $p_j^\dagger \Lambda$ cannot decrease with increasing Λ . Indeed, if Λ increases and since $\sum_{j=1}^n p_j^\dagger \Lambda = \Lambda$, either all probabilities remain the same or at least one probability must increase. If one probability increases, then $\exists j | \ell_j(p_j^\dagger \Lambda) = \tau$ increases, but then all terms $\ell_i(p_i^\dagger \Lambda)$ must increase for the property of the NEP. Since functions ℓ_i are monotonically increasing, that means that all values of $p_i^\dagger \Lambda$ must increase.

The case of $p_j^* \Lambda$ is similar and the proof can be derived based on the fact that γ monotonically increases with Λ and is an invariant with respect to the server. $\square$

Lemma 6. $\tau' \geq 0$ and $\tau'' \geq 0$ at any traffic $\Lambda \geq 0$.

Proof. By deriving τ at any server $j \leq j^\dagger$, we obtain:

$$\tau' = \frac{d(p_j^\dagger \Lambda)}{d\Lambda} \ell_j'(p_j^\dagger \Lambda) \geq 0. \quad (37)$$

Indeed, the first factor is non-negative (cf. Lemma 5) and the second is positive by definition.

For the second derivative, we have:

$$\tau'' = \frac{d^2(p_j^\dagger \Lambda)}{d\Lambda^2} \ell_j'(p_j^\dagger \Lambda) + \left(\frac{d(p_j^\dagger \Lambda)}{d\Lambda} \right)^2 \ell_j''(p_j^\dagger \Lambda), \quad (38)$$

in which all terms are non-negative. In particular, notice that the following result holds:

$$\sum_{j=1}^{j^\dagger} \frac{d^2(p_j^\dagger \Lambda)}{d\Lambda^2} = \frac{d^2}{d\Lambda^2} \left(\sum_{j=1}^{j^\dagger} p_j^\dagger \Lambda \right) = 0, \quad (39)$$

which leads to the conclusion that either all second derivatives of $p_j^\dagger \Lambda$ are zero or at least one is positive and one negative. Taking a non-negative one and considering that τ'' is an invariant of the servers, the result follows. $\square$

D. Price of anarchy

The PoA is the ratio between the average latencies at the NEP and the optimum. It can be computed in general for any value of the offered traffic, denoted as $\eta(\Lambda)$, and can only assume values greater than or equal to 1:

$$\eta(\Lambda) = \frac{U(\mathbf{p}^\dagger)}{U(\mathbf{p}^*)} = \frac{\tau}{U(\mathbf{p}^*)} \geq 1. \quad (40)$$

With the MIND-IT formulation, the PoA can be computed in polynomial time, as a consequence of the previous algorithms being polynomial complexity. Moreover, further notable properties can be discussed in the following.

Conjecture 1. The PoA curve vs Λ is piece-wise convex.

The conjecture is motivated as follows. The PoA is the ratio of continuous functions of Λ , with the additional property that the denominator has a continuous and positive derivative because the condition used to compute γ at the optimum preserves the continuity of the derivative (see Lemma 1). The numerator is non-decreasing and convex (cf. Lemma 6), but its derivative can be discontinuous at the activation points of servers at the NEP. Therefore, consider the adjacent segments $[\Lambda_j^{(\text{NEP})}, \Lambda_{j+1}^{(\text{NEP})}]$. At $\Lambda_j^{(\text{NEP})}$, server j is activated, and the NEP solution assigns probabilities $\mathbf{p}^\dagger$ so that the curve of τ vs Λ is continuous. Immediately before and after the activation of server j the same latency is observed with a different number of servers, so that the last activated server j absorbs some load and the rest of the servers observe a slowdown in the rate of increase of their latency. This corresponds to a sudden decrease in the derivative of τ . This means that while τ vs Λ experiences a drop in its growth rate at any point at

which a new server is incorporated in the NEP, the latency at the optimum does not observe such a discontinuity in the derivative. The result is that, for arrival rates slightly higher than $\Lambda_j^{(\text{NEP})}$, the PoA can decrease. However, the growth rate of τ must increase quickly again and faster than the growth rate of the latency at the optimum because the NEP does not allow any server, not even the faster, to experience less latency than the slower. Therefore, the growth rate of τ is that of the slowest active server, while at the optimum this cannot occur by construction. Since the numerator of the PoA increases faster than the denominator, the PoA curve between two consecutive server activation events at the NEP must be convex. This behavior holds for all feasible load segments, including those from $\Lambda_n^{(\text{NEP})}$ to $\sum_{j=1}^n \mu_j$.

If the above conjecture holds, which is intuitive and has been confirmed in our experiments, then searching for the worst-case PoA becomes a simple task.

Result 1. *The maximum of the PoA vs Λ occurs at $\Lambda > 0$, at a point of activation of a server at the NEP or at $\rho = 1$.*

Result 1 tells that finding the maximum of the PoA can be done by evaluating a finite set of points, each of which requires to run in polynomial time the exact algorithms MIND-IT (NEP) and MIND-IT (Opt). Hence, the cost of evaluating the worst case behavior of the distributed approach is polynomial and comparable to the complexity of the exact algorithms for the computation of optimum and NEP.

IV. SPECIAL CASES

A. M/M/1 queues

With **M/M/1 queues**, ℓ_i and h_i can be inverted in closed form, since they have simple expressions:

$$\ell_i(x) = d_i + \frac{1}{\mu_i - x}; \quad (41)$$

$$h_i(x) = d_i + \frac{\mu_i}{(\mu_i - x)^2}. \quad (42)$$

General case expressions for optimization simplify into:

$$\gamma = d_j + \frac{\mu_j}{(\mu_j - p_j^* \Lambda)^2}, \quad p_j^* > 0, \quad \forall j \leq j^*; \quad (43)$$

$$p_j^* = \frac{1}{\Lambda} \left(\mu_j - \sqrt{\frac{\mu_j}{\gamma - d_j}} \right), \quad \forall j \leq j^*; \quad (44)$$

$$\frac{1}{\Lambda} \sum_{j=1}^{j^*} \left(\mu_j - \sqrt{\frac{\mu_j}{\gamma - d_j}} \right) = 1. \quad (45)$$

Notice that the complexity of inverting (45) is $\mathcal{O}(\log r)$ instead of $\mathcal{O}(n \log r)$ observed in the general case, because the inversion can be performed directly over the sum. However, the overall complexity of the exact optimization remains $\mathcal{O}(n^2 + n \log r)$.

The expressions needed to study the NEP become:

$$\tau = d_j + \frac{1}{\mu_j - p_j^\dagger \Lambda}, \quad p_j^\dagger > 0, \quad \forall j \leq j^\dagger; \quad (46)$$

$$p_j^\dagger = \frac{1}{\Lambda} \left(\mu_j - \frac{1}{\tau - d_j} \right), \quad \forall j \leq j^\dagger; \quad (47)$$

$$\frac{1}{\Lambda} \sum_{j=1}^{j^\dagger} \left(\mu_j - \frac{1}{\tau - d_j} \right) = 1. \quad (48)$$

As for the general case, the maximum value of the PoA can be found at any of the NEP activation points or for $\Lambda \rightarrow \sum_{j=1}^n \mu_j$. With M/M/1 queues, the latter can be computed in closed form.

Theorem 3. *With M/M/1 queues,*

$$\lim_{\Lambda \rightarrow \sum_{j=1}^n \mu_j} \eta(\Lambda) = n \sum_{j=1}^n \mu_j / \left(\sum_{j=1}^n \sqrt{\mu_j} \right)^2. \quad (49)$$

Sketch of the proof. The average latency at the NEP tends to diverge and can be computed in closed form by neglecting the fixed delay. By writing the expression of p_j^* vs γ in near-saturation conditions, with $\gamma \gg d_j$, and by summing all probabilities, we can derive the asymptotic expression for γ , hence derive p_j^* and the average latency at the optimum. $\square$

In case of **M/M/1 queues with equal delays** $d_j = d^*$, closed forms can be found from the above formulas, which generalizes the conclusions of Haviv and Roughgarden [16], who studied the optimization and NEP of systems with M/M/1 queues with $d^* = 0$.

If $d_j = d^*$ for all active servers, and all servers are M/M/1, then we obtain, in closed form:

$$p_i^* = \frac{1}{\Lambda} \left(\mu_i - \sqrt{\mu_i} \frac{\sum_{j=1}^{j^*} \mu_j - \Lambda}{\sum_{j=1}^{j^*} \sqrt{\mu_j}} \right), \quad (50)$$

$$= d^* + \frac{1}{\Lambda} \left(\frac{\left(\sum_{j=1}^{j^*} \sqrt{\mu_j} \right)^2}{\sum_{j=1}^{j^*} \mu_j - \Lambda} - j^* \right). \quad (51)$$

The result is a generalization of the conclusions of Haviv and Roughgarden [16].

The price of anarchy with M/M/1 and equal fixed delays is

$$\eta = \frac{d^* + \frac{j^\dagger}{\sum_{j=1}^{j^\dagger} \mu_j - \Lambda}}{d^* + \frac{1}{\Lambda} \left(\frac{\left(\sum_{j=1}^{j^\dagger} \sqrt{\mu_j} \right)^2}{\sum_{j=1}^{j^\dagger} \mu_j - \Lambda} - j^* \right)}. \quad (52)$$

We conclude that, with M/M/1 queues and homogeneous delays $d_j = d^*$, all quantities of interest for the performance evaluation can be expressed in closed form.

B. M/D/1 queues

With **M/D/1 queues**, the following expressions hold as long as the queues are stable (i.e., $x < \mu_j$):

$$\ell_j(x) = d_j + \frac{1}{2\mu_j} \frac{2 - x/\mu_j}{1 - x/\mu_j}, \quad (53)$$

$$h_j(x) = d_j + \frac{1}{2\mu_j} \left(1 + \frac{1}{(1 - x/\mu_j)^2} \right). \quad (54)$$

At the optimum, we have:

$$p_j^* = \frac{\mu_j}{\Lambda} \left(1 - \frac{1}{\sqrt{2\mu_j(\gamma - d_j) - 1}} \right), \quad \forall j \leq j^* \quad (55)$$

$$\frac{1}{\Lambda} \sum_{j=1}^{j^*} \mu_j \left(1 - \frac{1}{\sqrt{2\mu_j(\gamma - d_j) - 1}} \right) = 1. \quad (56)$$

Differently from the M/M/1 case, this time the optimal augmented latency γ cannot be put in closed form. Similarly, at the NEP, the average delay τ does not have a closed form:

$$p_j^\dagger = \frac{2\mu_j}{\Lambda} \frac{\mu_j(\tau - d_j) - 1}{2\mu_j(\tau - d_j) - 1}, \quad \forall j \leq j^\dagger, \quad (57)$$

$$\frac{2}{\Lambda} \sum_{j=1}^{j^\dagger} \mu_j \frac{\mu_j(\tau - d_j) - 1}{2\mu_j(\tau - d_j) - 1} = 1. \quad (58)$$

With some simple algebra it is possible to find that the asymptotic behavior of the PoA with M/D/1 queues does not change with respect to the M/M/1 case, hence (52) holds for M/D/1 queues as well.

C. M/G/1 queues

With **M/G/1 queues**, the following expressions hold as long as queues are stable (i.e., for $x < \mu_j$):

$$\ell_j(x) = d_j + \frac{1}{\mu_j} \left(1 + \frac{1 + C_j^2}{2} \frac{x}{\mu_j - x} \right), \quad (59)$$

$$h_j(x) = d_j + \frac{1}{\mu_j} \left(1 + \frac{1 + C_j^2}{2} \frac{2\mu_j - x}{(\mu_j - x)^2} x \right), \quad (60)$$

with C_j the coefficient of variation of server j 's service time.

At the optimum and at the NEP we have:

$$p_j^* = \frac{\mu_j}{\Lambda} \left(1 - \frac{1}{\sqrt{1 - \frac{2}{1+C_j^2} (1 - \mu_j(\gamma - d_j))}} \right), \quad \forall j \leq j^*; \quad (61)$$

$$\frac{1}{\Lambda} \sum_{j=1}^{j^*} \mu_j \left(1 - \frac{1}{\sqrt{1 - \frac{2}{1+C_j^2} (1 - \mu_j(\gamma - d_j))}} \right) = 1; \quad (62)$$

$$p_j^\dagger = \frac{\mu_j}{\Lambda} \left(1 - \frac{1}{1 - \frac{2}{1+C_j^2} (1 - \mu_j(\tau - d_j))} \right), \quad \forall j \leq j^\dagger; \quad (63)$$

$$\frac{1}{\Lambda} \sum_{j=1}^{j^\dagger} \mu_j \left(1 - \frac{1}{1 - \frac{2}{1+C_j^2} (1 - \mu_j(\tau - d_j))} \right) = 1. \quad (64)$$

The asymptotic expressions for the PoA is as follows:

$$\eta \simeq \frac{\Lambda \sum_{j=1}^n \frac{1+C_j^2}{2}}{\left(\sum_{j=1}^n \sqrt{\mu_j \frac{1+C_j^2}{2}} \right)^2} \rightarrow \frac{\sum_{j=1}^n \frac{1+C_j^2}{2} \sum_{j=1}^n \mu_j}{\left(\sum_{j=1}^n \sqrt{\mu_j \frac{1+C_j^2}{2}} \right)^2}. \quad (65)$$

It is interesting to notice that the asymptotic behavior of the PoA differs from the M/M/1 case due to terms $(1 + C_j^2)/2$. Moreover, as it can be expected, setting $C_j = 1$ for all servers yields the results previously found for the M/M/1 queue, while $C_j = 0$ would lead to results valid for the M/D/1 queue. Setting all C_j values to the same value would also lead to the

result derived in (52) for M/M/1 queues, which justifies why the asymptotic behavior of the PoA with M/D/1 and M/M/1 queues is the same.

V. PERFORMANCE EVALUATION

In this section we report results obtained with both our experimental setup that uses real servers reached over the Internet, and our analytical models.

A. Experimental apparatus

We set up a distributed measurement apparatus with the aim of obtaining a dependable ground truth to contrast with the predictions of our approach **MIND-IT** (in both versions of optimal server selection and selfish NEP). The apparatus enables the configuration of a group of servers (in our experiment, three servers were utilized—with one server also working as NAT to multiplex routes, resulting in a configuration comprising five servers with different communication latencies and isolated computing resources) with varying service capacities and diverse network locations. This tool is coded in Golang [57], utilizing microservices for deployment, and employs QUIC as the transport layer protocol [58]. The tool specifies three primary entities: clients, servers, and routing nodes. The server instantiates a configurable web server that exposes one or multiple services with distinct computational demands. Client and server support fine-tuning of applications and traffic shape characteristics. The routing element manages the traffic generated by other entities, directing it to the designated destination.

The tool collects and stores networking and traffic routing events, including packet arrivals and departures to and from various elements, and the dynamic count of available threads dedicated to handling incoming traffic at the servers. It also allows for the easy deployment of application-specific measurements.

The experimental setting comprises five servers deployed throughout Europe, each characterized by almost deterministic service times, practically no limits on available memory, and one working thread. The application requires clients to periodically issue 100-byte update packets to any of the available servers; the server computes the updated information or instruction for each client and then sends it back to her. The application, being aware of the available servers, randomly selects the server for each request according to a probability distribution on the server set. At startup, this distribution is initialized with a uniform selection probability. At run time, the application collects latency statistics from servers over a fixed observation window and updates the distribution accordingly. In particular, it decreases the selection probability of the server exhibiting the highest latency and increases that of the server with the lowest latency by the same factor. This probability adjustment factor is doubled if the ordering of the servers in terms of latency remains unchanged with respect to the previous observation window and halved otherwise.

In our setting, several co-located client elements issue packets to the five servers; clients generate traffic which is tuned so as to modulate the aggregate system load to values

TABLE I: Settings of the five scenarios considered for performance analysis. The notation $U\{x, y\}$ indicates a discrete uniform probability density function between x and y . The notation $\mathcal{U}(a, b)$ indicates a continuous uniform probability density function between a and b .

Scenario	Configuration	
0	Server type	M/D/1
	Number of servers	5
	Delays	$d = [36, 33, 31, 34, 3]$ ms
	servers' capacity	$\mu = [18, 12, 10, 8, 6]$ services/s
1	Server type	M/M/1
	Number of servers	5
	Delays	$d = [40, 30, 150, 70, 45]$ ms
	servers' capacity	$\mu = [15, 9, 30, 20, 15]$ services/s
2	Server type	M/M/1
	Number of servers	5
	Delays	$d = [10, 20, 10, 15, 12]$ ms
	servers' capacity	$\mu = [300, 200, 50, 10, 20]$ services/s
3	Server type	M/G/1
	Number of servers	3
	Delays	$d = [10, 12, 20]$ ms
	servers' capacity	$\mu = [300, 100, 200]$ services/s
4	Server type	M/M/1
	Number of servers	5
	Delays	$d = [10, 20, 10, 15, 12]$ ms
	servers' capacity	$\mu = [300, 200, 50, 10, 20]$ services/s
5	Server type	M/M/1
	Number of servers	$\sim U\{2, 10\}$
	Delays	$d_{\text{edge}} \sim \mathcal{U}(10, 25)$ ms
		$d_{\text{cloud}} \sim \mathcal{U}(50, 100)$ ms
	servers' capacity	$\mu_{\text{edge}} \sim \mathcal{U}(5, 20)$ services/s
		$\mu_{\text{cloud}} \sim \mathcal{U}(75, 150)$ services/s

in the range from 5% to 95%. In the experimental evaluation, we measure the round trip time between each client and the five servers; then, we use it to compute the allocation of traffic to servers, and we feed this configuration to the routing element and run the experiments. In the case of a game for the decentralized version of MIND-IT, this is equivalent to saying that rational players can do the math and directly choose their actions at the NEP without any iteration.

B. Results and validation

We used six significant scenarios, whose parameters are reported in Table I, to validate our analytical model and showcase the analytical and algorithmic results derived in this work. As detailed in what follows, each scenario serves different purposes, from validating the model in real network conditions to highlighting interesting properties of the PoA under ideal or more realistic assumptions on server behaviors and system-state knowledge at the involved servers.

Scenario 0—Validation. We start by validating MIND-IT via experiments in a real network context. We observed the two-way delay between users and servers with ping packets and used the average round-trip time (RTT) to run our analytical model. The characteristics of the available servers, in order of activation, are $d = [36, 33, 31, 34, 3]$ ms and $\mu = [18, 12, 10, 8, 6]$ services/s. We remark that here, in the results computed with the model, we use the minimum network delay values observed in the experimental implementation described above, which encompasses servers and users in two different countries.

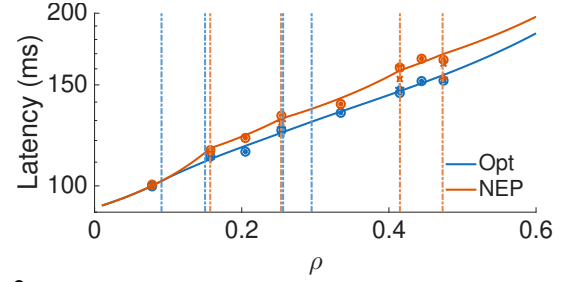


Fig. 2: Validation in Scenario 0; markers represent measurements for different system load estimation: i) no errors - circles, ii) -10% - +, and iii) +10% - x

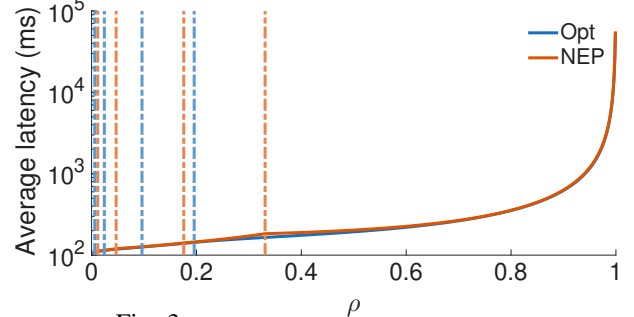


Fig. 3: Latency vs offered load in Scenario 1

Fig. 2 reports results generated by model and experiments (averages are reported as markers, while 95% confidence intervals – barely visible – are delimited by dots). The figure shows the measured latency observed at the activation thresholds of the servers, which is also reported with vertical dotted lines using the same color as the corresponding latency curves, and at a point in between two activation points. Thresholds are computed analytically, based on the average two-way delay observed. The figure also shows the effect of errors of $\pm 10\%$ on the estimation of the system load at the activation point.

Analytical results show that performance differences between the optimum and the NEP are generally not large and can only be experienced when at least two servers are active because with one active server there is nothing to optimize.

Experimental results closely match the analysis, confirming that assuming a constant two-way delay, rather than adopting a stochastic model, provides a practical and accurate simplification. Moreover, performance under load estimation errors shows minimal deviation from the error-free case, demonstrating that the system is resilient even to significant inaccuracies.

Scenario 1—M/M/1 Edge & Cloud. In this scenario, we consider five M/M/1 servers with different capacity. This scenario serves the purpose of comparing latencies under simple (Markovian) server configurations that have been studied in the literature. Server distances and speed values are chosen to represent a typical edge-cloud scenario with a few (4 in our case) slow servers in the edge and one fast server in the cloud. Indeed, one of the five servers is much faster than the others, but much farther apart from the users than the other four servers. The characteristics of the available servers, in order of activation, are $d = [40, 30, 150, 70, 45]$ ms and $\mu = [15, 9, 30, 20, 15]$ services/s.

The average system latency as a function of the offered load in this scenario is shown in Fig. 3. The figure illustrates how, as

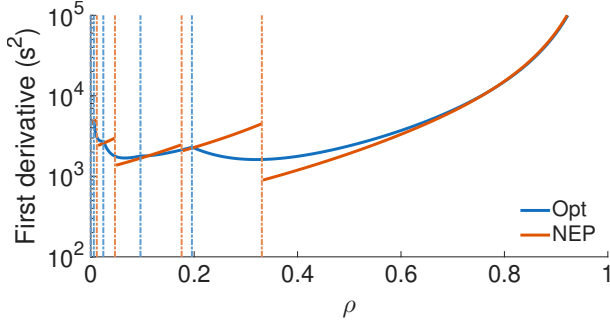


Fig. 4: First derivative of latency vs offered load in Scenario 1

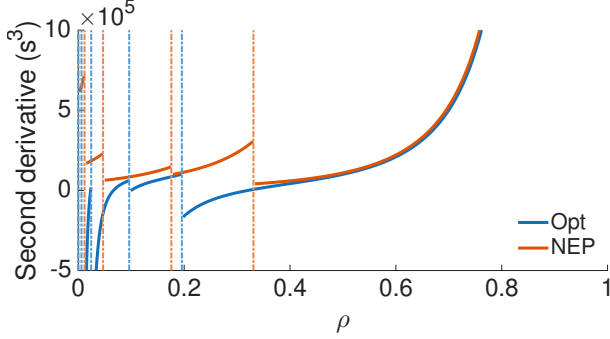


Fig. 5: Second derivative of latency vs offered load in Scenario 1

the second server gets activated at the optimum, the latency at the NEP starts to increase faster than in the optimized system. Indeed, Fig. 4 reports the behavior of the first derivative of the latency at the optimum and at the NEP, and shows that the NEP latency exhibits an always increasing derivative between the activation points, while the optimum can smoothly decrease and then increase again. However, the activation of servers at the NEP causes a sudden decrease (a discontinuity) in the rate of increase of the latency, after which it goes up quickly. The result is that the rate of growth of the latency at the NEP can be smaller than at the optimum after an NEP activation point.

Moreover, as shown in Fig. 5, the second derivative of the latency at the NEP is higher than at the optimum. Both curves in the figure exhibit discontinuities with drops at the respective activation points, and the curve at the optimum can go negative while at the NEP we observe only positive values, as predicted by the analysis (cf. Lemma (6)).

This behavior causes the piece-wise convexity of the resulting PoA curve, shown in Fig. 6. In this case, the peak of the PoA is reached at the activation of the last server at the Nash equilibrium. The corresponding value is neither negligible nor very large (below 1.15). It is interesting to notice that the PoA can be quite different at different loads and that a distributed implementation of the service request allocation could work well at quite high loads in this scenario. However, this result does not apply in general, as we will see in the analysis of the following scenario.

Scenario 2—M/M/1 Heterogeneous Edge. Here, we use five M/M/1 edge servers, all close to users, with significant differences in capacity. This is a possible scenario in which only edge servers can be accessed, e.g., due to restrictions

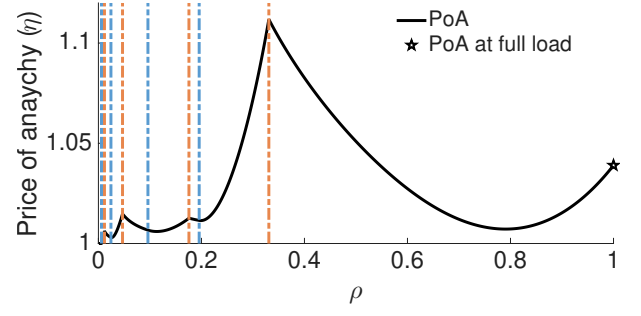


Fig. 6: Price of anarchy vs load in Scenario 1. The marked point indicates the value computed with (49).

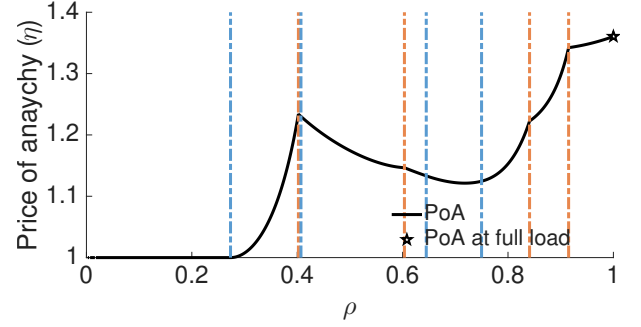


Fig. 7: Price of anarchy vs load in Scenario 2. The marked point indicates the value computed with (49).

in the policy enforced by network operators on services offered to their customers. Moreover, edge servers here are not homogeneous, representing the typical case of network slicing in which different servers might have to be devoted to several services with non-homogeneous geographical requirements. Hence, this scenario is useful to understand whether full-fledged edge deployments can suffer high PoA values. In order of earlier server activation, we have $d = [10, 20, 10, 15, 12]$ ms and $\mu = [300, 200, 50, 10, 20]$ services/s. The results of this scenario are shown in Fig. 7. In this case, the optimization activates all servers before the NEP dictates the activation of the fourth server due to the reduced capacity of the last two servers.

Interestingly, in this case the PoA increases also after the activation of both the fourth and fifth servers at the NEP. which happens because those servers are activated at the NEP after the optimum has activated all of them; hence, the capacity used at the optimum remains higher than the one used at the NEP even after all servers are also active for the NEP.

In this case, the shape of the PoA curve is very different from that of Scenario 1, and a distributed implementation might be suitable only for low loads. In general, the shape of η vs. ρ is a concatenation of convex functions, although predicting the position and value of the peak is not straightforward, as the shape of the curve can change significantly with the distance and capacity of the available servers.

Scenario 3—Impact of service time variance. We consider three M/G/1 servers to compare the impact of the coefficients of variation of service times. This scenario is therefore useful for evaluating more realistic and generic server behaviors.

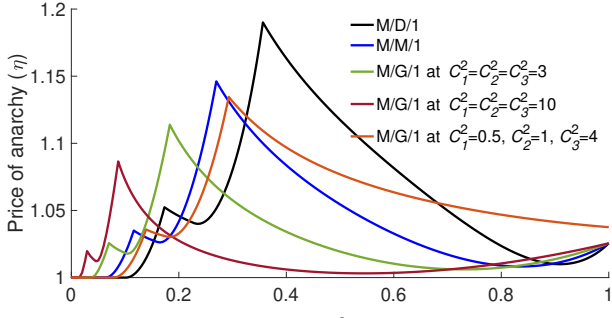


Fig. 8: Scenario 3: Price of anarchy vs load at various levels of variance

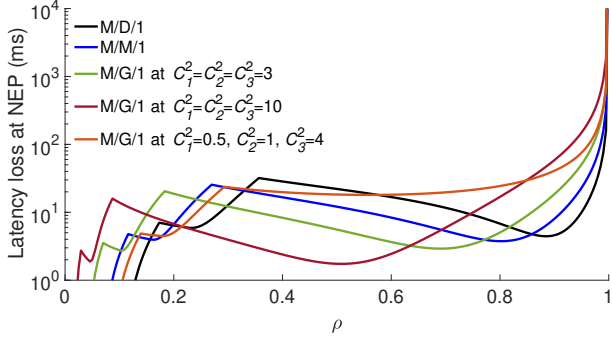


Fig. 9: Scenario 3: Extra delay at the NEP, with respect to the optimum, vs load at various levels of variance

Results are reported in Figs. 8 and 9. The figures include the special cases of M/D/1 and M/M/1 servers, obtained using coefficients of variation equal to 0 and 1, respectively. They also include two cases with high and very high variance, obtained by setting all coefficients of variation to the same value, which is either 3 or 10, as well as a case with different coefficients of variation for each server: one is set low ($C_1^2 = 0.5$), one corresponds to the exponential case ($C_2^2 = 1$) and one to high variance ($C_3^2 = 4$). All servers are close to the users $d = [10, 12, 20]$ ms, and have different service capacity $\mu = [300, 100, 200]$ services/s.

We can see that the peak of PoA decreases with variance, as shown in Fig. 8. The PoA peak moves to the right of the graphs (i.e., it occurs at higher loads) as the variance of the service time decreases. This happens because the denominator of the PoA increases with the variance, and hence partially hides the excess latency due to operating the service at the NEP. In fact, Fig. 9 shows that the net latency loss due to a distributed implementation has peaks comparable for the different cases and tends to be very large only when the load approaches 1. We see that the case with three different values of coefficient of variation has a shape similar to the other cases but significantly higher values of PoA at high loads. It is also interesting to note that the asymptotic value of the PoA (for $\rho \rightarrow 1$) is 1.0255 in all cases in which the coefficients of variation are homogeneous (first four curves in the figure), while the case with different coefficients yields a higher value of 1.0375, visible in the figure, and calculated with (65) in all cases. In general, the results for scenario 3 show that the interplay of heterogeneous server characteristics affects the value of the PoA and cannot be neglected.

Eventually, with the experiment of Fig. 10, we discuss the

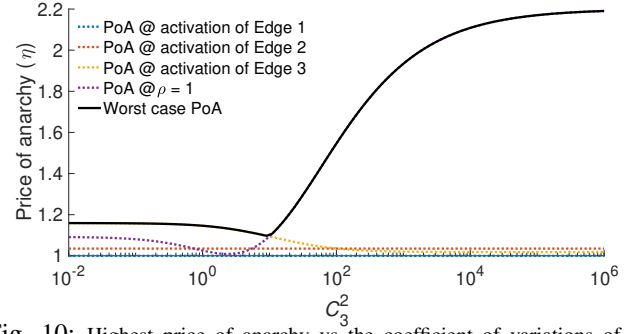


Fig. 10: Highest price of anarchy vs the coefficient of variations of the server farther apart (Edge 3)

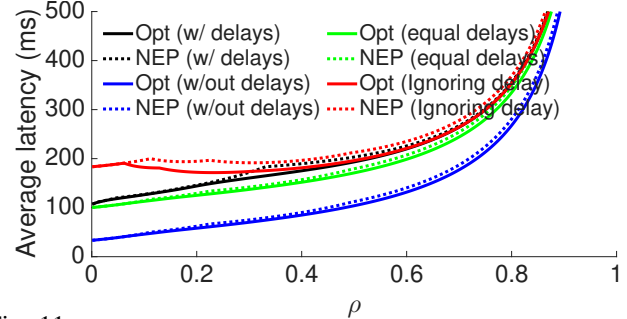


Fig. 11: Latency vs offered load in Scenario 4 with variants (with respect to Scenario 1) on the evaluation and consideration of fixed delays

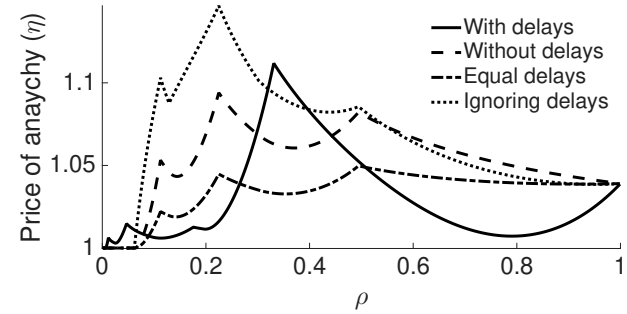
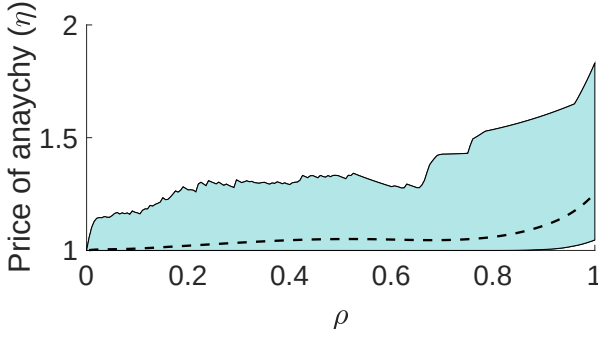


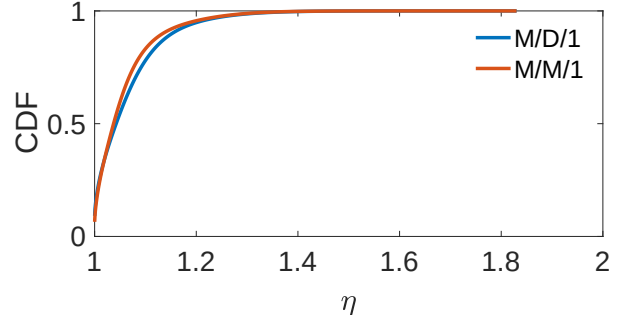
Fig. 12: PoA vs load in Scenario 4 with variants (with respect to Scenario 1) on the evaluation and consideration of fixed delays

impact of the variance observed at a single server while the rest of the servers use M/M/1 queues. The dotted lines in the figure show the value of the PoA points that are candidates for the worst case, i.e., NEP activation points and full load. As shown in the figure, the worst case (solid line) is dominated by one of the activation points until the point at full load takes the lead and the system starts behaving far from optimally. Here we have reported the case in which we vary the coefficient of variation of the service time in the server farther apart, but the behavior is qualitatively the same for all servers. The curve of the worst case PoA has an asymptotic value which is easily obtained from (65), whose value is $\sum_k \mu_k / \mu_j$, for the case where we vary C_j^2 . However, the fact that the worst PoA can be high when it is dominated by the behavior at full load is not critical for two reasons: first, the system should not run close to saturation and, second, other candidate points for the worst PoA value never yield high PoA values, so that a service that runs at reasonable load will never suffer high PoA.

Scenario 4—The importance of accounting for fixed delays. Figs. 11 and 12 report results for a network con-



(a) Area covered by PoA curves for the M/M/1 case, with average plotted as a dashed line



(b) Empirical distributions of the PoA observed for the cases of M/M/1 and M/D/1 servers

Fig. 13: Comparison of PoA with 500 000 scenarios generated at random (Monte Carlo style) with 1 to 5 “edge” servers (10 to 25 ms of latency, 5 to 20 services/s each) and 1 to 5 “cloud” servers (50 to 100 ms of latency, 75 to 150 services/s each)

figuration like that of Scenario 1, although optimization and NEP are calculated under different assumptions. This is not a different scenario *per se*, but rather the evaluation of the potential impact of partial system knowledge in the decision made in a distributed setup. Specifically, we consider a first variant in which we neglect the existence of fixed delays in the optimization algorithm and in the NEP calculation (assuming that the user is not actually observing latency but simply trusting her math). This case is labeled as “Ignoring delays” in the figures and shows large errors with respect to the correct optimization or NEP calculation. Thus, accounting for fixed delays makes a huge difference. We further compare the results discussed for Scenario 1 with the case in which fixed delays d_j are set to zero (“Without delays”, in the figures). In this case, the results are sensibly better than in the original setup considered for Scenario 1 (“With delays”, in the figures), at least in terms of latency, and only until the load becomes high. This means that, again, fixed delays play an important role and neglecting them is not possible before the average sojourn time into servers becomes predominant. Another variant is obtained by considering a modified Scenario 1 with fixed delays, equal for all servers (“Equal delays”, in the figures), set to the average value of the original Scenario 1 (i.e., $d_j=67$ ms for all servers). The latency curves in this case are smoother, and the difference from Scenario 1 vanishes as the load approaches 1.

Note that the dotted curve in the figure, obtained by ignoring the fixed delays in optimization and NEP calculation, is not even (piecewise) convex. However, delays are present, so they are properly accounted for in the aftermath of the optimization/NEP evaluation. This result does not contradict Theorem 1 as the dotted curve is the result of an inaccurate mathematical approximation that could lead to inaccurate optimizations and/or NEP calculations, for the latter case, selfish users acting distributedly can easily see it. The dotted curve is intentionally included for the purpose of showing that inaccurate assumptions might not only lead to wrong decisions but also hide important properties.

In all cases, the differences in terms of the maximum value observed in the PoA are limited, although (i) when delays are uniform, the PoA tends to be smaller and (ii) the shapes of the curves are very different, with peaks occurring at very different loads.

Scenario 5 – Impact of the number of servers. MIND-IT algorithms allow for fast computation of system performance at optimum and NE, for any load for which the system is stable. To generalize the findings presented in the previous subsections, which hold for a limited number of system parameters, here we use our algorithms to evaluate a large number of randomly generated system realizations in the M/D/1 and M/M/1 server cases. Fig. 13 shows results for 500 000 different realizations generated with a typical Monte Carlo approach. In each realization, we use two different classes of servers: “edge” and “cloud”. By edge server, we mean a server that is relatively close to users and has limited capacity. For edge servers, we use the continuous range that goes from 10 to 25 ms of latency (picked uniformly at random and independently for each server). Each edge server can handle, on average, between 5 and 20 requests per second (for each server, we pick a value uniformly at random in that continuous range). In contrast, by cloud server we intend a fast but far server. We pick cloud server latencies in the range 50 to 100 ms, and cloud server capacity between 75 and 150 requests/s. The number of edge servers, in each realization, is chosen at random in the integer range 1 to 5; likewise, each realization involves 1 to 5 cloud servers, so as to have between 2 and 10 servers in total.

Fig. 13a illustrates the area covered by the PoA curves generated in 500 000 experiments in the M/M/1 case. The figure illustrates how the η values typically span low ranges, with peaks that can be higher under near-saturation conditions. The dashed line in the figure marks the average PoA observed for each load value ρ , having normalized the traffic offered to the aggregate capacity of all servers chosen and configured at random in each experiment. The average values are typically very low, exceeding 1.05 only for loads higher than 75%, reaching 1.1 about 90% of load, and the average remains below 1.255 in all cases. Throughout the set of experiments, we have not observed any PoA value above 1.83, and, as shown in Fig. 13b, the 99-th percentile of η is 1.318 for the M/D/1 case and 1.307 for the M/M/1 case. Fig. 13b also shows that, in terms of distribution of PoA values, there is little difference between what was observed for M/M/1 and M/D/1 servers.

VI. CONCLUSIONS

The analysis and optimization of computing task allocation in the edge-cloud continuum require attention to path latencies, whose effect was traditionally neglected. We have shown that incorporating them in the analysis of the delay experienced by end-users substantially complicates the optimization and identification of the NEP with respect to the simpler cases that were previously studied. Yet, our study shows that both the optimal allocation and the NEP can be characterized analytically, and MIND-IT can compute them exactly and in polynomial time. The analysis of the system PoA is also performed, which reveals useful aspects for network and system designers, allowing them to make informed decisions on whether to approach request allocation in a distributed or centralized manner. We remark that we have studied and evaluated the distance between fully centralized and fully distributed decisions, showing how to compute efficiently how much efficiency is lost when users act selfishly rather than coordinating to optimize the access to shared network servers.

The analysis we presented is general, and only requires that the sojourn time of a task in a server be an increasing and convex function of the server load, which is a common property of non-fully-deterministic systems. Our analytical results embed, generalize, and extend the results of previous studies for any kind of queueing discipline.

Our findings were validated through the deployment of a real distributed system that spans a multiparty laboratory in different countries. Our results show that (i) optimal configurations and selfish Nash equilibria are not necessarily intuitive to find, as they exhibit a strong dependency on relative differences between available servers, in terms of capacity and distance from the user, and (ii) distributed and selfish optimizations incur limited performance costs, unless the system is driven into deep saturation and the service time variance becomes unrealistically high.

REFERENCES

- [1] V. Mancuso, P. Castagno, L. Badia, M. Sereno, and M. Ajmone Marsan, "Optimal allocation of tasks to networked computing facilities," in *Proc. ASMTA (ACM Sigmetrics workshop)*, Jun. 2024.
- [2] Z. Hou, C. She, Y. Li, D. Niyato, M. Dohler, and B. Vucetic, "Intelligent communications for tactile internet in 6g: Requirements, technologies, and challenges," *IEEE Communications Magazine*, vol. 59, no. 12, pp. 82–88, 2021.
- [3] R. W. Coutinho and A. Boukerche, "Design of edge computing for 5g-enabled tactile internet-based industrial applications," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 60–66, 2022.
- [4] N. Sehadi, L. Bariah, W. Hamidouche, H. Hellouli, R. Jäntti, and M. Debbah, "Generative ai for immersive communication: The next frontier in internet-of-senses through 6g," *Comm. Mag.*, vol. 63, no. 2, p. 31–43, Feb. 2025. [Online]. Available: <https://doi.org/10.1109/MCOM.001.2400199>
- [5] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, 2019.
- [6] C.-X. Wang, X. You *et al.*, "On the road to 6G: Visions, requirements, key technologies and testbeds," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2023.
- [7] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2131–2165, 2021.
- [8] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, 2018.
- [9] S. Zarei, S. Azizi, and A. Ahmed, "Optimizing edge server placement and load distribution in mobile edge computing using ACO and heuristic algorithms," *J. Supercomput.*, vol. 81, no. 1, p. 257, 2025.
- [10] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, A. de la Oliva, and A. Azcorra, "Modeling mobile edge computing deployments for low latency multimedia services," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 464–474, 2019.
- [11] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, X. Wang, V. C. Leung, and W. Wang, "Large-scale measurements and optimizations on latency in edge clouds," *IEEE Trans. on Cloud Comput.*, vol. 12, no. 4, pp. 1218–1231, 2024.
- [12] C. E. Bell and S. Stidham, "Individual versus social optimization in the allocation of customers to alternative servers," *Manag. Sc.*, vol. 29, pp. 831–839, 1983.
- [13] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," in *Proc. ACM SIGCOMM*, 2003, p. 151–162.
- [14] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint IP backbone," *IEEE Netw.*, vol. 17, no. 6, pp. 6–16, 2003.
- [15] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 219–232, 2004.
- [16] M. Haviv and T. Roughgarden, "The price of anarchy in an exponential multi-server," *Op. Res. Lett.*, vol. 35, no. 4, pp. 421–426, 2007.
- [17] D. Milojevic, "The edge-to-cloud continuum," *Computer*, vol. 53, no. 11, pp. 16–25, 2020.
- [18] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for Tor," in *Proc. Financial Crypt. Data Sec. Conf.* Springer, 2012, pp. 98–113.
- [19] T. Roughgarden, "The price of anarchy is independent of the network topology," in *Proc. ACM STOC*, 2002, p. 428–437.
- [20] R. Johari and J. N. Tsitsiklis, "Efficiency loss in a network resource allocation game," *Math. Op. Res.*, vol. 29, no. 3, pp. 407–435, 2004.
- [21] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," *Comput. Sci. Rev.*, vol. 3, no. 2, p. 65–69, 2009.
- [22] S. Ghosh and R. Hassin, "Inefficiency in stochastic queueing systems with strategic customers," *Eur. J. Op. Res.*, vol. 295, no. 1, pp. 1–11, 2021.
- [23] Y. Sheffi, *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [24] D. Braess, "Über ein Paradoxon aus der Verkehrsplanung," *Unternehmensforschung*, vol. 12, 1969.
- [25] A. C. Pigou, *The Economics of Welfare*. Macmillan, London, 1920.
- [26] R. Feldmann, M. Gairing, T. Lucking, B. Monien, and M. Rode, "Selfish routing in non-cooperative networks: a survey," in *Proc. MFCS*. Springer, 2003, p. 21–45.
- [27] T. Roughgarden, *Routing and the Price of Anarchy*. The MIT Press: Cambridge, MA, USA, 2005.
- [28] R. Hassin and M. Haviv, *To queue or not to queue: Equilibrium behavior in queueing systems*. Springer Science & Business Media, 2003, vol. 59.
- [29] T. Wu and D. Starobinski, "A comparative analysis of server selection in content replication networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1461–1474, 2008.
- [30] A. Orda, R. Rom, and N. Shimkin, "Competitive routing in multiuser communication networks," *IEEE/ACM Trans. Netw.*, vol. 1, no. 5, pp. 510–521, 1993.
- [31] S. C. Dafermos and F. Sparrow, "The traffic assignment problem for a general network," *Journal of Research of the National Bureau of Standards*, vol. 738, no. 2, pp. 91–118, 1969.
- [32] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Econometrica*, pp. 520–534, 1965.
- [33] W. H. Sandholm, "Potential games with continuous player sets," *Journal of Economic theory*, vol. 97, no. 1, pp. 81–108, 2001.
- [34] D. Starobinski and T. Wu, "Performance of server selection algorithms for content replication networks," in *Int. Conf. Res. Netw.*, 2005, pp. 443–454.
- [35] T. Wu and D. Starobinski, "On the price of anarchy in unbounded delay networks," in *Proc. ACM GameNets*, 2006, pp. 13–es.
- [36] S. Stidham, *The Price of Anarchy for a Network of Queues in Heavy Traffic*. Boston, MA: Springer US, 2014, pp. 91–121.
- [37] —, *Optimal Design of Queueing Systems*. Chapman & Hall/CRC, 2009.
- [38] S. Ranjan, R. Karrer, and E. W. Knightly, "Wide area redirection of dynamic content by Internet data centers," in *Proc. IEEE Infocom*, 2004, pp. 816–826.

- [39] T. H. Szymanski, "An ultra-low-latency guaranteed-rate Internet for cloud services," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 123–136, 2014.
- [40] 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.
- [41] W. Fan, F. Xiao, Y. Pan, X. Chen, L. Han, and S. Yu, "Latency-aware joint task offloading and energy control for cooperative mobile edge computing," *IEEE Trans. Serv. Comput.*, 2025, early access.
- [42] N. Yang, Y. Liu, S. Chen, M. Zhang, and H. Zhang, "Minimizing AoI in mobile edge computing: Nested index policy with preemptive and non-preemptive structure," *arXiv preprint arXiv:2508.20564*, 2025.
- [43] L. Wu, P. Sun, Z. Wang, Y. Li, and Y. Yang, "Computation offloading in multi-cell networks with collaborative edge-cloud computing: A game theoretic approach," *IEEE Trans. Mob. Comput.*, vol. 23, no. 3, pp. 2093–2106, 2024.
- [44] H. Teng, Z. Li, K. Cao, S. Long, S. Guo, and A. Liu, "Game theoretical task offloading for profit maximization in mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 22, no. 9, pp. 5313–5329, 2023.
- [45] G. Jain, A. Kumar, and S. A. Bhat, "Recent Developments of Game Theory and Reinforcement Learning Approaches: A Systematic Review," *IEEE Access*, vol. 12, pp. 9999–10011, 2024.
- [46] S. Li, X. Hu, and Y. Du, "Deep Reinforcement Learning and Game Theory for Computation Offloading in Dynamic Edge Computing Markets," *IEEE Access*, vol. 9, pp. 121 456–121 466, 2021.
- [47] S. Liang, H. Wan, T. Qin, J. Li, and W. Chen, "Multi-User Computation Offloading for Mobile Edge Computing: A Deep Reinforcement Learning and Game Theory Approach," in *Proceedings of the IEEE 20th International Conference on Communication Technology (ICCT)*, 2020, pp. 1534–1539.
- [48] Q. Xu, Z. Su, and R. Lu, "Game Theory and Reinforcement Learning Based Secure Edge Caching in Mobile Social Networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3415–3429, 2020.
- [49] K. Li, "Scheduling independent tasks on multiple cloud-assisted edge servers with energy constraint," *Journal of Parallel and Distributed Computing*, vol. 184, p. 104781, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S074373152300151X>
- [50] X. Jin, S. Zhang, Y. Ding, and Z. Wang, "Task offloading for multi-server edge computing in industrial internet with joint load balance and fuzzy security," *Scientific Reports*, vol. 14, no. 1, p. 27813, 2024. [Online]. Available: <https://doi.org/10.1038/s41598-024-79464-2>
- [51] A. Ortiz, "Learning and forgetting for server selection in mobile edge computing: a perspective," *e+i Elektrotechnik und Informationstechnik*, vol. 142, no. 3, pp. 216–220, 2025. [Online]. Available: <https://doi.org/10.1007/s00502-025-01328-y>
- [52] European Telecommunications Standards Institute, "ETSI TS 123 501 - V15.3.0 - 5G; System Architecture for the 5G System (3GPP TS 23.501 version 15.3.0 Release 15)," Technical Specification, 2018.
- [53] ETSI, "TS 128 531 - V18.2.0 - 5G; Management and orchestration; Provisioning (Release 16)," ETSI, Technical Specification, 2023.
- [54] X. Lin and N. B. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 766–781, 2006.
- [55] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.
- [56] R. Srikant and L. Ying, *Communication networks: An optimization, control and stochastic networks perspective*. Cambridge University Press, 2014.
- [57] Golang developers, "Golang." [Online]. Available: <https://go.dev>
- [58] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-based secure and reliable transport for HTTP/2." [Online]. Available: tools.ietf.org/html/draft-tsvwg-quic-protocol-02



Vincenzo Mancuso is Associate Professor at the University of Palermo, Italy, and Research Associate Professor at IMDEA Networks, Madrid, Spain. Previously, he was with INRIA (France) and Rice University (USA). He obtained his Ph.D. in 2005 from University of Palermo. His research focus is on analysis, design, and experimental evaluation of wireless and intelligent edge architectures for communication and computation.



Paolo Castagno is currently serving as an Assistant Professor at the Computer Science Department of the University of Torino in Italy. He obtained his Ph.D. from the same university in 2018. His research area mainly revolves around the study, design, and assessment of distributed systems and services, with a special focus on wired and wireless network applications.



Leonardo Badia (Senior Member, IEEE) received the Ph.D. degree in Information Engineering from the University of Ferrara, Italy, in 2004. From 2002 to 2003, he was a Visiting Scholar at the Radio System Technology Labs (currently Wireless@KTH), Royal Institute of Technology, Stockholm, Sweden. In 2006, he joined the IMT Institute for Advanced Studies, Lucca, Italy. In 2011, he moved to the University of Padova, Italy, where he is currently a Full Professor. He has published more than 300 research papers. His research interests include cross-

layer optimization in wireless communications, transmission protocol modeling, and the application of game theory to radio resource management. He has also served on the editorial boards of several scientific journals and remains an active reviewer for many international periodicals, as well as a technical program committee member for conferences in the broad areas of communications and networking.



Matteo Sereno was born in Nocera Inferiore, Italy. In 1987, he earned a Laurea degree in Computer Science from the University of Salerno, followed by a Ph.D. in Computer Science from the University of Torino in 1992. Presently, he holds the position of Full Professor at the Computer Science Department of the University of Torino. His current research focuses on the performance evaluation of computer systems, communication networks, distributed systems, game theory, queueing networks, stochastic Petri net models, and their application in analyzing

computer and telecommunication systems.



Marco Ajmone Marsan is a research professor at the IMDEA Networks Institute in Spain. From 1974 to 2021 he was at the Politecnico di Torino, with an interruption from 1987 to 1990, when he was at the University of Milan. He is IEEE Fellow, and member of the Academia Europaea and of the Academy of Sciences of Torino. He is qualified as "ISI Highly Cited Researcher" in computer science. He received a honorary degree from the Budapest University of Technology and Economics.