Optimal Allocation of Tasks to Networked Computing Facilities

Vincenzo Mancuso[1][0000−0002−4661−381X], Paolo Castagno[2][0000−0002−1349−1844], Leonardo Badia[3][0000−0001−5770−1199], Matteo Sereno[2][0000−0002−5339−3456], and Marco Ajmone Marsan[1][0000−0002−9560−7053]

[1] IMDEA Networks Institute, Madrid, Spain
{vincenzo.mancuso, marco.ajmone}@imdea.org
[2] Department of Informatics, University of Turin, Turin, Italy
{paolo.castagno, matteo.sereno}@unito.it
[3] University of Padova, Padua, Italy leonardo.badia@unipd.it

Abstract. Distributed allocation of computing tasks over network resources is meant to decrease the cost of centralized allocation. However, existing analytical models consider practically indistinguishable resources, e.g., located in the data center. With the rise of edge computing, it becomes important to account for the impact of diverse latency values imposed by edge/cloud data center locations. In this paper, we study the optimization of computing task allocation considering both the delays to reach edge/cloud data centers and the response times of servers. We explicitly evaluate the resulting performance under different scenarios. We show, through numerical analysis and real experiments, that differences in delays to reach data center locations cannot be neglected. We also study the price of anarchy of a distributed implementation of the computing task allocation and unveil important properties such as the price of anarchy being generally small, except when the system is overloaded, and its maximum can be computed with low complexity.

Keywords: Network servers; Optimization with network latency constraints; Next generation networking; Game Theory; Price of Anarchy

1 Introduction

Meeting latency requirements is fundamental to achieve the desired quality of service for real-time applications [23, 28]. A paradigm often adopted in mobile Internet architectures to tackle this issue is that of edge computing, i.e., bringing computing resources closer to end users, rather than processing all data in the cloud [12]. However, as the mobile Internet becomes more pervasive, the management of distributed computing infrastructure is evolving towards an edge-cloud continuum, rather than a dichotomy between edge or cloud [3].

From the standpoint of an individual user, the problem is limited to the choice of the best (i.e., minimum latency) path [13]. When a global perspective is adopted, establishing coordination among multiple users becomes of formidable

complexity and is practically infeasible. The crux of the matter becomes whether distributed approaches to server selection in extremely variegate network architectures can still be efficient [1].

While this problem received attention in the past, we argue that the available frameworks are inadequate to represent the edge-cloud continuum. Most investigations consider a homogeneous latency model among the alternatives [10], so that the comparisons involve, e.g., fast vs slow servers but just with different parameters in the same formula. Instead, the alternatives in the mobile Internet are different not just in quantitative but also in qualitative terms.

One particular instance of this aspect is the fixed component of the latency, which comes for the most from the physical distance of the server in the edge-cloud continuum [15]. We will show how neglecting this aspect leads to suboptimal choices. To complicate things further, one can observe that, while service capacity and fixed latency can be possibly known to the user, the final performance depends on congestion at the chosen server, which is harder to estimate [25].

Following an algorithmic game theory pathway [19], we consider service choices made by a multitude of atomic non-cooperative agents interested in minimizing their own latency, and we derive the price of anarchy (PoA) [11]. Compared to results available in the literature, we take a general approach applicable to any functional relationship describing the latency, under very mild assumptions of positive first and second derivative. This results in a direct low-complexity implementation akin to water-filling algorithms [27]. Moreover, thanks to the generality of our approach, we can validate our quantitative results with real-world experiments.

Thus, we present multiple contributions: First, we evaluate the efficiency of distributed choices by network users in the edge-cloud continuum, under a general framework not found in the previous literature. We investigate the Nash equilibrium (NE) of distributed selections, which is found to be unique. Second, we formulate exact algorithms to find the optimal allocation point and the NE. Third, we compute the PoA as a function of network load. The worst-case PoA is proven to be the maximum among a finite number of cases. Eventually, we perform an extensive evaluation corroborated by experimental results.

2 Related Work

Server selection for a computing task typically represents and compares the servers as queues. The novelty of this paper is to include a fixed delay term to reach the selected server, after which the waiting and service time in the queue depend on the load as a general function with positive first and second derivatives. This resonates in the literature at many levels. To start, research in transportation optimization, led by Braess [2] and Pigou [17], has yielded significant findings that have applications in computer networks, routing, and server selection optimization. In-depth discussions can be found in [6] or [19].

A close alignment with our setup of server selection can be found in [1, 10, 26]. These works compare a selfish strategy, where users choose servers to minimize

their own mean waiting time, against the social (i.e., global) optimization. In [16], the problem is addressed as a network routing problem, where each user aims to optimize its own performance. This leads to a non-cooperative game, with an emphasis on the conditions for an NE. Studies [1] and [10] analyze n exponential servers with FCFS discipline and develop a closed-form solution for the PoA. They consider service (i.e., queue response time) as the only parameter. If we introduce a fixed delay for each server, these PoA derivations become unfeasible.

Many more works exist in this area, since multiple authors worked on this and related issues. The highly influential work [18] led to scrutinizing the problem through diverse lenses, exploring numerous variants. For instance, an attempt to go beyond an exponential service time distribution is made in [26] for $M/G/1$ servers. An exact analysis of the PoA is presented only for a processor-sharing queuing discipline, where it is known that the average delay depends on the service time distribution only through its mean. Paper [26] highlights the mathematical difficulty of scenarios with FCFS $M/G/1$ servers, where it is inevitable to deal with choices depending on two parameters: mean and coefficient of variation of the service time distribution. [1] investigates a choice among $M/G/1$ servers focusing on the service rate and the coefficient of variation of the service time distribution. However, that analysis is valid only under the assumption that all servers share a common coefficient of variation. Another approach attempting, in various ways, to address server selection as characterized by two parameters is presented in [20], where heterogeneous costs and service times across different paths are studied. In there, the PoA is derived by applying simplifications or reducing the analysis to specific scenarios. A different approach that circumvents the use of two parameters in the service selection is presented in [21], with servers of type $M/G/1$ and $GI/GI/1$, but only under heavy traffic conditions that greatly simplifies the analysis.

Yet, we show that using multiple parameters is tractable. Indeed, we characterize each server with at least *two* independent parameters, i.e., delays in addition to service rates. Also, we generalize the delay function, as it can be defined by an arbitrary number of parameters, and admit, e.g., heterogeneous coefficients of variance, showing that obtaining exact solutions is still possible. In particular, we show that optimal solutions and NEs can be computed with exact algorithms in polynomial time.

Evaluations of the impact of the distance in cloud/edge scenarios generally pertain the problem of server placement [24]. More recently, with the evolution of multi-server architectures towards a horizontal edge-cloud continuum without hierarchy, the issue of choosing the servers from the individual user perspective has gained momentum [22]. However, most of the proposals advocate for heuristic low-complexity solutions or reinforcement learning strategies, which are found to be efficient. Our analysis shows that this does not happen by accident, but rather is grounded in the problem as characterized by useful structural properties [4] that we rigorously prove to hold even with the inclusion of the delay terms.

In conclusion, we present the first analysis that does not neglect the delay term to reach the servers—a practice that previous studies have regularly in-

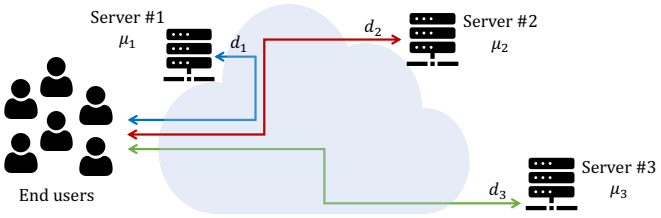

Fig. 1. Reference scenario: a group of users on a single network slice with computing resources accessible through the network in the edge-cloud continuum

troduced, to make analytical expressions tractable. Our results solve a problem that has been open for many years, and avoid erroneous solutions to the server selection, computing instead the real optimal choices (in selfish or global terms).

3 Analysis with Fixed Path Delay

Assume that a group of users is connected to a network with n servers, as shown in Fig. 1. Although the servers are accessible at different distances, they belong to the same network slice, created and deployed over the edge-cloud continuum, as typical of modern 3GPP networks [5]. Servers are sorted in increasing order according to the average delay to serve a job, in turn determined by the sum of path latency and average service time, excluding queuing delays, that is,

$$\forall i,j \in \mathcal{S} = \{1,\cdots,n\},\ i<j \implies d_i{+}1/\mu_i \leq d_j{+}1/\mu_j, \tag{1}$$

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 can make slow but close servers preferable to fast but far ones. Indeed, all previously published results only hold for the case $d_i{=}0$ and some of them can be extended to the case $d_i{=}d_j$, $\forall i,j$.

In the following, $\mathbf{p}^\star = \{p_i^\star\}_{i=1,\cdots,n}$ denotes a probability vector that allocates load to servers so as to minimize the average latency, and $\mathbf{p}^\dagger = \{p_i^\dagger\}_{i=1,\cdots,n}$ denotes the probability vector at the NE, i.e., where the users selfishly split their load with a stochastic strategy, to minimize their own expected latency.

3.1 Optimal load allocation problem

We aim to minimize the average system latency $U(\mathbf{p})$. If $\ell_i(x) > 0$ is the average latency of server i with capacity μ_i when it receives traffic with intensity $0{\leq}x{<}\mu_i$, and Λ is the aggregate offered traffic, the average latency is

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

The above latency has to be minimized subject to the following constraints:

$$p_i \geq 0, \quad \forall i; \qquad p_i\Lambda \leq \mu_i, \quad \forall i; \qquad \sum_i p_i = 1. \tag{3}$$

Function $\ell_i(x)$ accounts for path latency (d_i), average service capacity (μ_i), and can 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., $\ell_i(0) = d_i + 1/\mu_i$. Besides, in systems subject to congestion, $\ell_i(x)$ monotonically increases with x. 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, which makes the problem strictly convex. Thus, the solution is unique.

Lagrangian—with multipliers α_i, β_i, and γ—and KKT necessary conditions for the optimal solution $\mathbf{p}^\star$, are as follows:

$$\begin{aligned}
&\mathcal{L} = \sum_i p_i \ell_i(p_i\,\Lambda) - \sum_i \alpha_i\, p_i + \sum_i \beta_i\,(p_i\,\Lambda - \mu_i) + \gamma\left(1 - \sum_i p_i\right);\\
&\frac{\partial \mathcal{L}}{\partial p_i} = \ell_i(p_i\,\Lambda) + p_i\,\Lambda \ell_i'(p_i\,\Lambda) - \alpha_i + \Lambda\beta_i - \gamma = 0, \quad \forall i;\\
&\alpha_i\, p_i = 0, \quad \beta_i\,(p_i\Lambda - \mu_i) = 0, \qquad \forall i;\\
&\gamma\big(1 - \textstyle\sum_i p_i\big) = 0.
\end{aligned}$$

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. Multipliers α_i and β_i must be non-negative, while γ can take any real value (because it is associated to an equality constraint).

To identify which servers will receive traffic, consider the KKT conditions at $\mathbf{p}^\star$, which is a solution of the formulated problem. There are two cases:

Case $p_j^\star = 0$. If the optimal solution consists in assigning zero load to server j, then the KTT conditions imply that

$$\alpha_j \geq 0, \quad \beta_j = 0, \quad \ell_j(0) - \alpha_j - \gamma = 0, \quad \Rightarrow \ell_j(0) \geq \gamma, \tag{4}$$

i.e., servers that do not need to be active are those for which the latency computed at empty queue is not less than γ. However, with no queueing, a job is served in $\ell_j(0) = d_j + 1/\mu_j$, for any function ℓ_j. Therefore, if server j is not active, then server $i > j$, for which $d_i + 1/\mu_i \geq d_j + 1/\mu_j$, is also inactive. Being the set of servers ordered, there exists an integer $j^\star \leq n$ such that all servers from 1 to $j^\star$ receive some traffic, while the remaining ones are inactive.

Finding γ and $j^\star$ is part of the optimization. Moreover, their values are related, as it will emerge from the analysis of KKT conditions for $p_j^\star{>}0$.

Case $p_j^\star > 0$. For all nodes $j = 1, \cdots, j^\star$, which receive non-zero load in the optimal configuration, the following conditions must hold:

$$\alpha_j = \beta_j = 0, \qquad \gamma = \ell_j(p_j^\star\,\Lambda) + p_j^\star\,\Lambda\ell_j'(p_j^\star\,\Lambda). \tag{5}$$

Thus, in a sense, we are following a water-filling approach on the derivative of the weighted latency. Moreover, term γ is larger than the largest latency of any server, and thus we will refer to it as to the *augmented latency*. The latter must also be positive and, since $p_j^\star > 0$, must also be greater than any $\ell_j(0)$ for any active server, which leads to

$$\gamma > d_{j^\star} + 1/\mu_{j^\star}. \tag{6}$$

These results are coherent with (4), which defines when a server receives no load. They also prove γ to be a monotonically increasing—hence invertible—function $h_j(x) = \ell_j(x) + x\ell'_j(x)$, which has to be computed at $x = p_j^\star\, \Lambda$ at the optimum. If h_j^{-1} indicates the inverse function, we have

$$p_j^\star = h_j^{-1}(\gamma)/\Lambda. \tag{7}$$

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

$$1 = \sum_{j=1}^{j^\star} p_j^\star = \frac{1}{\Lambda} \sum_{j=1}^{j^\star} h_j^{-1}(\gamma). \tag{8}$$

Since the sum of increasing functions is also increasing, the RHS of (8) is invertible. In addition, the RHS must be between 0 and $\sum_{j=1}^{j^\star} \mu_j/\Lambda \geq 1$, so that all the offered traffic can be served. Thus, there exists a unique γ satisfying (8).

Because of conditions (4) computed on $j^\star{+}1$ and (6) for $j^\star$, γ must be found in a specific interval, i.e., $\gamma \in \left(d_{j^\star} + \frac{1}{\mu_{j^\star}}, d_{j^\star+1} + \frac{1}{\mu_{j^\star+1}}\right]$, where $d_{j^\star+1}{+}\frac{1}{\mu_{j^\star+1}}$ has to be taken as infinite if $j^\star{=}n$.

Determining $j^\star$ beforehand is key to compute the problem solution efficiently. Indeed, once $j^\star$ is known, by inverting normalization (8), one can compute γ, which in turn can be used in (7) to compute the optimal probabilities $p_j^\star$.

To find $j^\star$, we need to consider that probabilities $p_j^\star$ depend on Λ, although so far we have treated Λ as a constant. Notice that γ must be monotonically increasing in Λ, because so are all ℓ_j and the associated derivatives that define γ through (5). Thus, as traffic Λ increases, the augmented latency γ increases too, so that servers are progressively activated following the order of (1). Thus, when server j is activated at $\Lambda_j^{(\mathrm{Opt})}$, load $p_j^\star$ is still 0 and, according to (5), $\gamma = \ell_j(0) = d_j + \frac{1}{\mu_j}$. Hence, at $\Lambda_j^{(\mathrm{Opt})}$ we have:

$$p_i^\star\, \Lambda_j^{(\mathrm{Opt})} = h_i^{-1}\left(d_j + \frac{1}{\mu_j}\right), \quad \forall i \leq j, \tag{9}$$

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

$$\Lambda_j^{(\mathrm{Opt})} = \sum_{i=1}^{j-1} h_i^{-1}\left(d_j + \frac{1}{\mu_j}\right). \tag{10}$$

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

This optimization requires the inversion of a few monotonic functions, which admits closed form only in specific cases; in general, it can be done numerically with lightweight algorithms such as the dichotomous search. The following `MIND-IT(Opt)` (Minimum Delay Independent of Traffic and Service - Optimum) algorithm finds probability vector $\mathbf{p}^\star$ minimizing the average system latency.

`MIND-IT(Opt)`
`Input:` Sorted servers $j \in \{1, \cdots, n\}$, Λ.
`Step 1:` Compute activation thresholds $\Lambda_j^{(\mathrm{Opt})}$ with (10).
`Step 2:` Compute $j^\star$ by comparing Λ to the thresholds.
`Step 3:` Set $p_j^\star = 0, \forall j{>}j^\star$ and compute γ by inverting (8).
`Step 4:` Compute $p_j^\star$, $\forall j \leq j^\star$, with (7).
`Output:` $p_j^\star, j \in \{1, \cdots, n\}$.

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

Proof. The optimality of feasible $\mathbf{p}^\star$ follows from it satisfying all KKT conditions. Evaluating the activation thresholds requires computing a number of terms quadratic in n and inverting (8). The latter consists of 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 one, with complexity $\mathcal{O}(n \log r)$, where r is the target numerical resolution. The complexity of computing each of the n probabilities, with (7), is that of one inversion, i.e., $\mathcal{O}(\log r)$. Therefore, the complexity of `MIND-It(Opt)` is $\mathcal{O}(n^2 + n \log r)$, polynomial for any r.

3.2 The Nash equilibrium

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

With sorted servers, a user sends traffic to j only if the observed average latency is above the minimum possible latency at that server, $\ell_j(0)$. An NE exists and is unique due to $U(\mathbf{p})$ being strictly convex [1]. Such a NE can be approached through subsequent approximations, in a water-filling form [7]. At the NE, all users choose a strategy $\mathbf{p}_\mathbf{u}^\dagger = \mathbf{p}^\dagger$, which is also the same for all users due to symmetry, where the servers that receive traffic experience the same average latency τ, so that no user has an incentive to deviate from $\mathbf{p}^\dagger$:

$$\ell_j(p_j^\dagger \Lambda) = \tau, \qquad p_j^\dagger > 0, \qquad \forall j \leq j^\dagger, \tag{11}$$

where $j^\dagger$ is the number of used servers ($p_j^\dagger = 0$ 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 that, for a server j with $p_j^\dagger > 0$, the latency's lower bound is

$$\tau > \ell_j(0) = d_j + 1/\mu_j, \qquad \forall j \leq j^\dagger. \tag{12}$$

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

$$p_j^\dagger = \ell_j^{-1}(\tau)/\Lambda, \quad \forall j \le j^\dagger; \tag{13}$$

$$1 = \sum_{j=1}^{j^\dagger} \ell_j^{-1}(\tau)/\Lambda, \tag{14}$$

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 RHS of (14) is monotonic increasing and upper-bounded by $(1/\Lambda)\sum_{j=1}^{j^\dagger} \mu_j \ge 1$, as the traffic offered to each server cannot exceed the capacity (i.e., $\ell_j^{-1}(\tau) \le \mu_j$) and Λ cannot exceed the aggregate capacity of active servers. Notice that τ must be $d_{j^\dagger} + 1/\mu_{j^\dagger}$ when server $j^\dagger$ gets switched on, and as soon as $p_{j^\dagger}^\dagger$ becomes larger than zero, τ must be comprised in the interval

$$\tau \in \left(d_{j^\dagger} + 1/\mu_{j^\dagger}, d_{j^\dagger+1} + 1/\mu_{j^\dagger+1}\right), \quad \text{at } p_{j^\dagger}^\dagger > 0 \text{ and } p_j^\dagger = 0 \; \forall j > j^\dagger. \tag{15}$$

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

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

$$\Lambda_j^{(\mathrm{NE})} = \sum_{i=1}^{j-1} \ell_j^{-1}\left(d_i + 1/\mu_i\right). \tag{16}$$

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

`MIND-IT(NE)`
`Input:` Sorted servers $j \in \{1, \cdots, n\}$, Λ.
`Step 1:` Compute thresholds $\Lambda_j^{(\mathrm{NE})}$ with (16).
`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 (14).
`Step 4:` Compute $p_j^\dagger$, $\forall j \le j^\star$, with (13).
`Output:` $p_j^\dagger, j \in \{1, \cdots, n\}$.

Theorem 2. *The `MIND-IT(NE)` calculation algorithm is polynomial.*

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

3.3 Properties

Here we present a few interesting properties, which will help comparing NE and optimal working points as the offered traffic changes. Proofs are omitted due to lack of space, but can be found in [14] jointly with more properties.

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

$$\frac{d}{d\Lambda}U(\mathbf{p}^\star) = \frac{\gamma - U(\mathbf{p}^\star)}{\Lambda} = \sum_{j=1}^{j^\star} \left(p_j^\star\right)^2 \ell_j'(p_j^\star\,\Lambda) > 0. \tag{17}$$

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

Lemma 2. $\Lambda > 0 \implies \gamma > \tau$.

Lemma 3. $\Lambda_j^{(\mathrm{NEP})} \geq \Lambda_j^{(\mathrm{Opt})}, \forall j \in \{1, \cdots, n\}$.

Lemma 4. $\tau' \geq 0$ *and* $\tau'' \geq 0$ *at any traffic* $\Lambda \geq 0$.

3.4 Price of anarchy

The PoA in the studied system is the ratio between average latency at the NE and average latency at the optimum. It is a function of the offered traffic—denoted as $\eta(\Lambda)$—and can only assume values greater than or equal to 1:

$$\eta(\Lambda) = \frac{U(\mathbf{p}^\dagger)}{U(\mathbf{p}^\star)} = \frac{\tau}{U(\mathbf{p}^\star)} \geq 1. \tag{18}$$

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 also continuous and positive derivative because the condition used to compute γ at the optimum preserves the continuity of the derivative (cf. also Lemma 1). The numerator is non-decreasing and convex (cf. Lemma 4), but its derivative can be discontinuous at the activation points of servers at the NE, which occur after the activation of servers at the optimum (cf. Lemma 3). Therefore, consider the adjacent segments $[\Lambda_j^{(\mathrm{NE})}, \Lambda_{j+1}^{(\mathrm{NE})}]$. At $\Lambda_j^{(\mathrm{NE})}$, server j is activated and the NE assigns probabilities $\mathbf{p}^\dagger$ so that the curve of τ vs Λ be 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 servers observe a slowdown in the rate of increase of their latency. This corresponds to a sudden decrease of 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 NE, the latency at the optimum does not

observe such a discontinuity in the derivative. The result is that, for arrival rates slightly larger than $\Lambda_j^{(\mathrm{NE})}$, the PoA can decrease. However, the growth rate of τ must quickly increase again and faster than the growth rate of the latency at the optimum because the NE 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 activations at the NE must be convex. This behavior holds for all feasible load segments, including from $\Lambda_n^{(\mathrm{NE})}$ to $\sum_{j=1}^{n} \mu_j$.

If the above conjecture holds, as confirmed by our experiments, then searching for the worst-case PoA becomes simple.

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

Result 1 tells that finding the maximum 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(NE)` 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 optimum and NE.

4 Special case: M/M/1 queues

With **M/M/1 queues**, ℓ_i and h_i can be inverted in closed form as

$$\ell_i(x) = d_i + \frac{1}{\mu_i - x}; \qquad h_i(x) = d_i + \frac{\mu_i}{(\mu_i - x)^2}. \tag{19}$$

General case expressions for optimization simplify into:

$$\gamma = d_j + \frac{\mu_j}{\left(\mu_j - p_j^\star \Lambda\right)^2}, \quad p_j^\star > 0, \quad \forall j \le j^\star; \tag{20}$$

$$p_j^\star = \frac{1}{\Lambda}\left(\mu_j - \sqrt{\frac{\mu_j}{\gamma - d_j}}\right), \quad \forall j \le j^\star; \tag{21}$$

$$\frac{1}{\Lambda}\sum_{j=1}^{j^\star}\left(\mu_j - \sqrt{\frac{\mu_j}{\gamma - d_j}}\right) = 1. \tag{22}$$

The complexity of inverting (22) is $\mathcal{O}(\log r)$ instead of $\mathcal{O}(n \log r)$ observed in the general case, because the inversion can be done over the sum directly. However, the overall complexity of the exact optimization remains $\mathcal{O}(n^2 + n \log r)$.

The expressions needed to study the NE become:

$$\tau = d_j + \frac{1}{\mu_j - p_j^\dagger \Lambda}, \quad p_j^\dagger > 0, \quad \forall j \le j^\dagger; \tag{23}$$

$$p_j^\dagger = \frac{1}{\Lambda}\Big(\mu_j - \frac{1}{\tau - d_j}\Big), \quad \forall j \le j^\dagger; \tag{24}$$

$$\frac{1}{\Lambda}\sum_{j=1}^{j^\dagger}\left(\mu_j - \frac{1}{\tau - d_j}\right) = 1. \tag{25}$$

In the very special case of **M/M/1 queues with equal delays** $d_j{=}d^\star$, closed forms can be found from the above formulas, which generalizes the conclusions of [10] for the optimization and NE of systems with M/M/1 queues with $d^\star{=}0$. In particular, if $d_j = d^\star$ for all active servers, then we obtain:

$$p_i^\star = \frac{1}{\Lambda}\left(\mu_i - \sqrt{\mu_i}\frac{\sum_{j=1}^{j^\star}\mu_j - \Lambda}{\sum_{j=1}^{j^\star}\sqrt{\mu_j}}\right), \tag{26}$$

$$U(\mathbf{p}^\star) = d^\star + \frac{1}{\Lambda}\left(\frac{\left(\sum_{j=1}^{j^\star}\sqrt{\mu_j}\right)^2}{\sum_{j=1}^{j^\star}\mu_j - \Lambda} - j^\star\right). \tag{27}$$

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

$$\eta = \left(d^\star + \frac{j^\dagger}{\sum_{j=1}^{j^\dagger}\mu_j - \Lambda}\right) \Bigg/ \left(d^\star + \frac{1}{\Lambda}\left(\frac{\left(\sum_{j=1}^{j^\star}\sqrt{\mu_j}\right)^2}{\sum_{j=1}^{j^\star}\mu_j - \Lambda} - j^\star\right)\right). \tag{28}$$

We conclude that, with M/M/1 queues and homogeneous delays $d_j = d^\star$, all quantities of interest can be expressed in closed form. However, this is in general not true when fixed latency terms d_j are not homogeneous. Nonetheless, the PoA at the border of the stability region of the system can be always computed in closed form, as shown in the following theorem.

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

$$\lim_{\Lambda\to\sum_{j=1}^{n}\mu_j}\eta(\Lambda) = n\sum_{j=1}^{n}\mu_j \Bigg/ \left(\sum_{j=1}^{n}\sqrt{\mu_j}\right)^2. \tag{29}$$

Proof (Sketch). The average latency at the NE tends to diverge and can be computed in closed form by neglecting the fixed delay. By comparing $p_j^\star$ vs γ in near-saturation conditions, with $\gamma{\gg}d_j$, and summing all probabilities, we get the asymptotic expression for γ, hence obtain $p_j^\star$ and the average optimal latency.

5 Performance evaluation

Experimental platform — We set up a distributed measurement apparatus to obtain a ground truth and compare it with the predictions of our model. The apparatus enables the configuration of a group of servers (3 in our evaluations) with varying service capacities and diverse locations. This tool is coded in Golang [8], utilizing microservices for deployment, and employs QUIC as transport layer protocol [9]. The tool specifies three entities: clients, servers, and routing nodes.

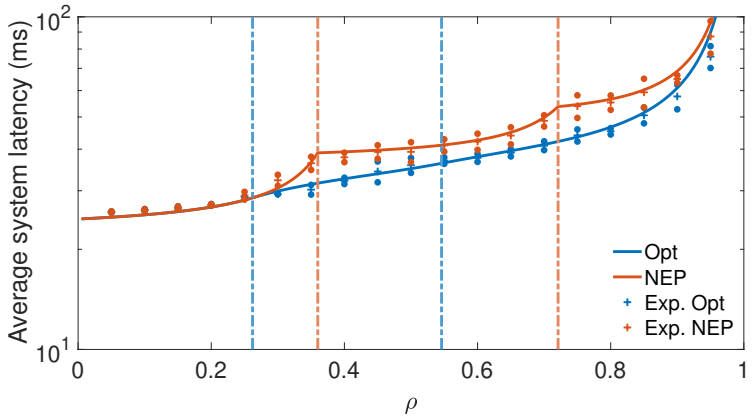

Fig. 2. Validation in Scenario 0

The server instantiates a configurable web processor that exposes one or multiple services with distinct computational demands. Client and server support a fine-tuning of applications and traffic shape characteristics. The routing element manages the traffic generated by other entities, directing it to its destination.

The tool collects and stores networking and routing events, including packet arrivals and departures to and from various elements, real-time monitoring of memory occupancy, and the count of available threads dedicated to handling incoming traffic. It also enables deploying application-specific measurements.

The experimental setting uses 3 servers deployed across Europe, each with deterministic service times, practically no bounds on available memory, and one working thread. The application requires clients to issue requests of 100 bytes to any available server; the server replies to each request by sending it back.

In our setting, several co-located clients issue packets to the three servers; the first two elements generate a traffic load equal to 5% of the overall system capacity—and this is the traffic being measured—while the last element provides background traffic. We measure the round trip time between each client and the three servers; then, we use it to compute the allocation of traffic to servers, and we feed this configuration to the routing element. The NE is found directly under the assumption of rational players, computing it beforehand.

Results and validation — We used a few significant scenarios to validate our analytical model and showcase the results derived in this paper. Due to space limitations, we only reports results for model validation and with simple queueing discipline, stressing the importance of taking into account fixed delays in the presence of servers deployed in the edge-cloud continuum.

Scenario 0—Validation. We start by validating our model via experiments in a real network context. We observed the two-way delay between users and servers with ping packets, and used the average RTT (round trip time) to run our analytical model. The characteristics of the available servers, in order of activation, are $d = [20, 34, 43.5]$ ms, and $\mu = [4.66, 5.00, 10.20]$ services/s.

Fig. 2 reports results generated by model and experiments (averages are reported as crosses, while 90% confidence intervals are delimited by dots). The figure also shows the activation thresholds of the servers, with vertical dotted lines (using the same color as the corresponding latency curves). Thresholds are computed analytically, based on the observed average two-way delay.

Analytical results show that differences between the optimum and the NE are generally small and can only be experienced when at least two servers are active. Experimental results match the analysis, which tells that considering a constant two-way delay instead of a stochastic model yields an affordable simplification.

Scenario 1—M/M/1 Edge & Cloud. We consider three M/M/1 servers with different capacity. One of the three servers is much farther apart from users than the other two servers, but it is faster. This represents a case in which two servers are within the edge area of the network and one is in the cloud. The characteristics of the available servers, in order of activation, are $d = [40, 30, 150]$ ms, and $\mu = [15, 9, 20]$ services/s. Latency and PoA as functions of the offered load in this scenario are shown in Fig. 3. As the second server gets activated at the optimum, the latency at the NE starts increasing faster than in the optimized system, and the PoA becomes larger than 1. However, the activation of the second server at the NE causes a temporary decrease in the PoA, after which it goes up quickly. The PoA curve is visibly piece-wise convex and the peak of the PoA is reached at the activation of the last server at the NE. The corresponding value is neither negligible nor very large (below 1.15). It is interesting that the PoA can be quite different at different loads, and a distributed implementation of the job allocation could work well at quite high loads.

Scenario 2—The importance of accounting for fixed delays. Fig. 4 reports results for a network configuration like the one of Scenario 1, although the optimization and the NE are computed under different assumptions and circumstances.

We first compare results discussed for Scenario 1 with the case in which fixed delays d_j are actually set to zero ("Without delays", in the figures). In this case, results are sensibly better than in the originally considered setup ("With delays"), at least in terms of latency, and only until the load becomes high. This means that fixed delays play an important role and neglecting them is not possible before the average sojourn time into servers becomes predominant.

Secondly, we consider a modified Scenario 1 in which fixed delays are equal for all servers (labeled as "Equal delays" in the figures). Such a delay is set to the average value observed in the original Scenario 1 (i.e., we set here d_j=73.3 ms for all servers). Latency curves in this case are smoother than for Scenario 1, although the difference with Scenario 1 and with the case without fixed delays vanishes as the load approaches 1. Hence, the heterogeneity of fixed delays plays a non-negligible role as well, at least for low-medium loads.

We eventually consider the case in which the existence of fixed delays is neglected. This case is labeled as "Ignoring delays" in the figures. We remark that fixed delays are present in this case, and therefore latency and PoA values reported in the figures do account for their presence, although neither optimization

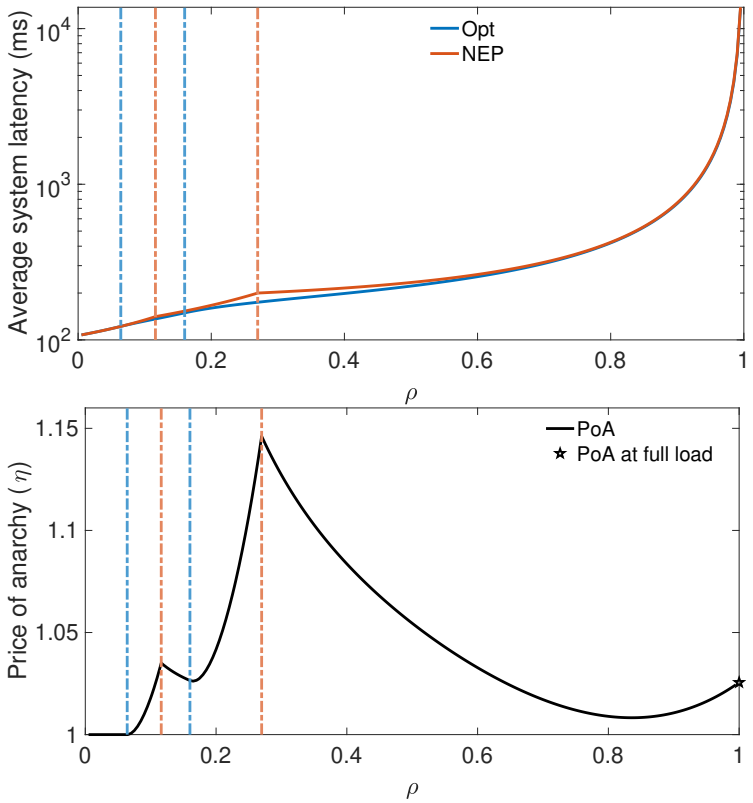

Fig. 3. Scenario 1. Top: latency, bottom PoA, vs offered load. The marked point indicates the PoA value computed with (29)

nor NE calculation considered them (as if they were set to zero although they are not). Notice that this approach reflects exactly what analytical state-of-the-art solutions could do in similar cases. Interestingly, obtained curves shows large errors with respect to the correct traffic allocation optimization or NE calculation, which remarks how accounting for the presence of fixed delays makes a huge difference.

Note that the dotted curve in the figure, obtained by ignoring the presence of fixed delays in optimization and NE calculation, is not even convex. This result does not contradict Conjecture 1 as the dotted curve is the result of an inaccurate mathematical approximation that leads optimizations and NE calculations astray—for the latter case, selfish users acting distributedly can easily see it. The dotted curve is intentionally included in the figures with the purpose to show that inaccurate assumptions might not only lead to wrong decisions, but also hide important properties.

As a final remark, Figure 4 shows that differences in terms of the maximum value for the PoA are not necessarily small and (i) when delays are uniform (and possibly tend to zero) the PoA tends to be smaller, whereas (ii) the shape of PoA

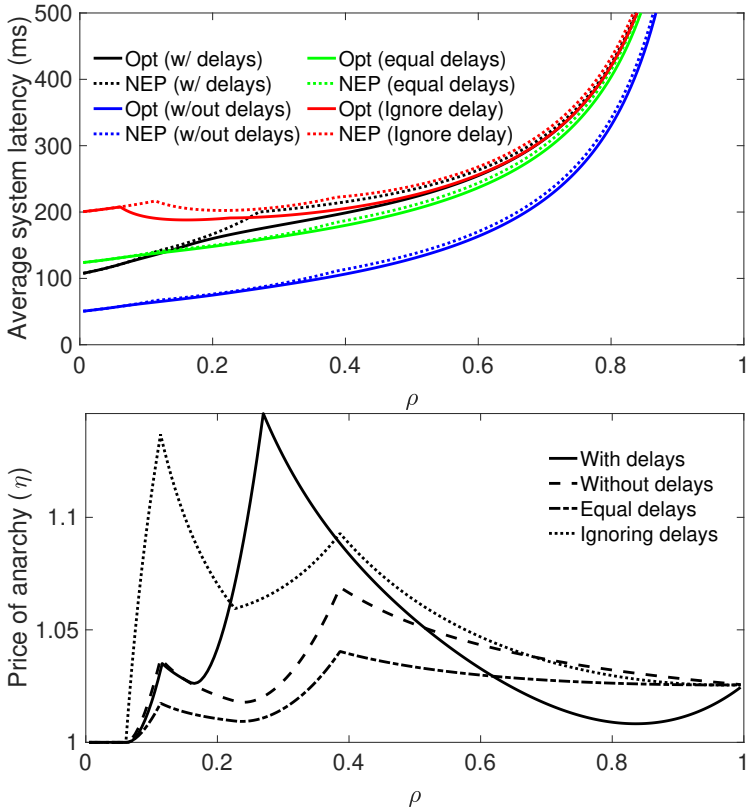

Fig. 4. Scenario 2. Left: latency, right PoA, vs offered load with variants on the evaluation and consideration of fixed delays

curves does not match, with peaks occurring at different loads in the different cases. The latter tells that a correct incorporation of fixed latency is paramount to design a system meant for distributed optimization, as it would be otherwise impossible to predict the load at which the PoA will be maximum—hence more critical—and its associated value.

6 Conclusions

The analysis and optimization of computing task allocation in the edge-cloud continuum requires attention to system delays, whose effect is traditionally neglected. We have shown that incorporating such delays in the analysis of the latency experienced by end-users complicates the optimization and the NE identification with respect to the simpler cases commonly studied. Yet, the optimal allocation and the NE can be characterized analytically, and computed with the algorithms derived in this paper. Our analysis is general, and only requires the sojourn time of a task to be an increasing and convex function of the server load, which is a common property of non-fully-deterministic systems.

Our findings were validated through a real deployment spanning over a multiparty laboratory across different countries. We showed that (*i*) optimal configurations and selfish NEs can exhibit a strong dependency on relative differences between servers in terms of capacity and distance from the user, and (*ii*) distributed and selfish optimizations incur limited costs, unless the system is driven into deep saturation and the service time variance becomes unrealistically high.

The next steps of our work will consider multiserver systems, and queues with limited buffer, where losses can occur.

Acknowledgements

This work has been supported by the Project AEON-CPS (TSI-063000-2021-38), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union NextGeneration-EU in the framework of the Spanish Recovery, Transformation and Resilience Plan, and the Italian National Recovery and Resilience Plan (NRRP), partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART").

References

1. Bell, C.E., Stidham, S.: Individual versus social optimization in the allocation of customers to alternative servers. Manag. Sc. **29**, 831–839 (1983)
2. Braess, D.: Uber ein Paradoxon aus der Verkehrsplanung. Unternehmensforschung **12** (1969)
3. Cheng, Z., Gao, Z., Liwang, M., Huang, L., Du, X., Guizani, M.: Intelligent task offloading and energy allocation in the uav-aided mobile edge-cloud continuum. IEEE Netw. **35**(5), 42–49 (2021)
4. Ding, Y., Li, K., Liu, C., Li, K.: A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing. IEEE Trans. Parallel Distrib. Syst. **33**(6), 1503–1519 (2021)
5. ETSI: TS 128 531 - V18.2.0 - 5G; Management and orchestration; Provisioning (Release 16). Technical specification, ETSI (2023)
6. Feldmann, R., Gairing, M., Lucking, T., Monien, B., Rode, M.: Selfish routing in non-cooperative networks: a survey. In: Proc. MFCS. p. 21–45. Springer (2003)
7. Garnaev, A., Trappe, W., Petropulu, A.: Equilibrium strategies for an OFDM network that might be under a jamming attack. In: Proc. IEEE CISS (2017)
8. Golang developers: Golang, https://go.dev
9. Hamilton, R., Iyengar, J., Swett, I., Wilk, A.: QUIC: A UDP-based secure and reliable transport for HTTP/2, tools.ietf.org/html/draft-tsvwg-quic-protocol-02
10. Haviv, M., Roughgarden, T.: The price of anarchy in an exponential multi-server. Op. Res. Lett. **35**(4), 421–426 (2007)
11. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. Comput. Sci. Rev. **3**(2), 65–69 (2009)
12. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: A survey. IEEE Commun. Surveys Tuts. **23**(4), 2131–2165 (2021)
13. Mancuso, V., Badia, L., Castagno, P., Sereno, M., Marsan, M.A.: Efficiency of distributed selection of edge or cloud servers under latency constraints. In: Proc. IEEE MedComNet. pp. 158–166 (2023)

14. Mancuso, V., Castagno, P., Badia, L., Sereno, M., Ajmone Marsan, M.: Optimal Allocation of Tasks and Price of Anarchy of Distributed Optimization in Networked Computing Facilities. Tech. rep., arXiv:2404.05543 [cs.GT] (2024), https://arxiv.org/abs/2404.05543
15. Milojicic, D.: The edge-to-cloud continuum. Computer **53**(11), 16–25 (2020)
16. Orda, A., Rom, R., Shimkin, N.: Competitive routing in multiuser communication networks. IEEE/ACM Trans. Netw. **1**(5), 510–521 (1993)
17. Pigou, A.C.: The Economics of Welfare. Macmillan, London (1920)
18. Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. Econometrica: Journal of the Econometric Society pp. 520–534 (1965)
19. Roughgarden, T.: Routing and the Price of Anarchy. The MIT Press: Cambridge, MA, USA (2005)
20. Stidham, S.: Optimal Design of Queueing Systems. Chapman & Hall/CRC (2009)
21. Stidham, S.: The Price of Anarchy for a Network of Queues in Heavy Traffic, pp. 91–121. Springer US, Boston, MA (2014)
22. Thai, M.T., Lin, Y.D., Lai, Y.C., Chien, H.T.: Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. IEEE Trans. Netw. Service Manag. **17**(1), 227–238 (2020)
23. Wang, C.X., You, X., et al.: On the road to 6G: Visions, requirements, key technologies and testbeds. IEEE Commun. Surveys Tuts. **25**(2), 905–974 (2023)
24. Wang, S., Zhao, Y., Xu, J., Yuan, J., Hsu, C.H.: Edge server placement in mobile edge computing. J. Parall. Distrib. Comput. **127**, 160–168 (2019)
25. Wang, T., Bauer, K., Forero, C., Goldberg, I.: Congestion-aware path selection for Tor. In: Proc. Financial Crypt. Data Sec. Conf. pp. 98–113. Springer (2012)
26. Wu, T., Starobinski, D.: A comparative analysis of server selection in content replication networks. IEEE/ACM Trans. Netw. **16**(6), 1461–1474 (2008)
27. Yu, W., Cioffi, J.M.: Constant-power waterfilling: performance bound and low-complexity implementation. IEEE Trans. Commun. **54**(1), 23–28 (2006)
28. Zhang, J., Letaief, K.B.: Mobile edge intelligence and computing for the Internet of vehicles. Proc. IEEE **108**(2), 246–261 (2019)