Efficiency of Distributed Selection of Edge or Cloud Servers under Latency Constraints

Vincenzo Mancuso*, Leonardo Badia†, Paolo Castagno‡, Matteo Sereno‡,◇, Marco Ajmone Marsan*
* IMDEA Networks Institute, Madrid, Spain
† Dept. of Information Engineering, University of Padova, Padova, Italy
‡ Dipartimento di Informatica, Università di Torino, Turin, Italy
◇Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Italy
email: vincenzo.mancuso@imdea.org, badia@dei.unipd.it, paolo.castagno@unito.it, matteo.sereno@unito.it, ajmone@polito.it

***Abstract*—We consider a set of network users (nodes) that generate latency-constrained service requests requiring the execution of computing tasks on servers located either in the cloud or at the network edge. We explore the efficiency of a distributed server selection strategically performed by individual nodes. In an earlier analysis, we argued for a stateless centralized allocation based on a probabilistic selection between edge and cloud servers. In that proposal, the optimal share of edge and cloud tasks was computed according to static network characteristics, with no knowledge of the actual network state. In this new study, we perform an analysis based on game theory, where we compare the globally optimal allocation performed at a central level against a distributed server selection driven by the selfish objectives of individual nodes. The inefficiency of the selfish allocation can be computed as the price of anarchy, which is shown to be very small, thus justifying a distributed strategic implementation of stateless policies. This insight is precious for designing algorithms for server selection and quantitatively proves the efficiency of distributed selfish approaches.**

***Index Terms*—Edge computing, Radio access network, Game theory, Distributed policies, Performance evaluation.**

I. Introduction

Future mobile communication networks require ubiquitous availability of services based on pervasive storage and computing [1] provided by either resource-abundant servers physically located in the network core, or less powerful peripheral servers offering the advantage of proximity to the end user.

A fundamental distinction is implied in many studies, and also reflected in our investigation, between cloud and edge servers [2]–[4]. Under an extreme simplification, a cloud server offers generally larger latency, but also higher capacity than a server located at the edge of the network. Since the ultimate performance experienced by a service request strongly depends on the congestion encountered at the chosen server, under working network conditions there is no dominant choice between the two server types. Rather, directions of individual service requests are tightly knit with one another and they should be harmonized so as to avoid overloads.

In the present contribution, we consider a network scenario where user service instances must be completed within a deadline, and their associated computing requirements can be satisfied either at the edge or in the cloud [5]. The network control can exploit different criteria to allocate service requests to the more convenient type of server, possibly depending on the current network conditions.

In [6], we discussed several algorithms, with the objective of investigating how the information available about the network state affects the overall performance and can be captured through different parameters that lead to optimize network management. We performed a comparison between *stateless* and *stateful* policies [7] used to route incoming requests, to either the cloud or the edge servers. A stateless policy would correspond to a choice based on network parameters (such as the server capacity and the overall service arrival rate) that are relatively stationary and easy to estimate, whereas a stateful policy exploits the instantaneous network conditions. A key result that goes in favor of a low-complexity implementation of the selection policy is that the performance improvements brought by stateful policies are minimal and are also prone to errors in the parameter estimation that can sometimes lead to worse performance with respect to stateless policies [6].

Thus, it might be more convenient to apply a simple stateless policy, like the one called `RANDALPH` (randomized alpha) in [6], which translates in the random assignment of a given fraction α of requests to the edge server, while the remaining share $1-\alpha$ is executed in the cloud. The value of α is computed in a centralized fashion, based on stateless network parameters, which is proven to be overall efficient in practical contexts.

In this paper, we push this further by analyzing whether stateless approaches for service request allocation can be distributed, still achieving efficient control, without awareness of the network state that would be expensive to acquire. To do so, we use the instruments of game theory [8], [9] to solve a distributed selection of a local parameter α_i for each request i, chosen by a strategic agent with only local information driven towards an individualistic objective. This results in a game theoretic implementation of the randomized alpha policy, therefore called `GANDALPH` (game theoretic `RANDALPH`).

We perform a quantitative comparison of the performance of the centralized policy `RANDALPH` against the distributed policy `GANDALPH`, in particular by considering the globally best assignment of the former and the Nash equilibrium (NE) achieved by the latter, and to compute the Price of Anarchy (PoA) [10]. While providing analytical justifications about why the PoA is expected to be contained, we present realistic evaluations that show that it practically falls within ranges of less than 10%, thereby supporting a fully distributed and scalable implementation of the selection policies.

The rest of this paper is organized as follows. Section II addresses the related work. Section III describes our scenario and how to evaluate a centralized stateless allocation of

jobs to edge and cloud servers. We present the distributed allocation modeled as a static game of complete information in Section IV, where we discuss the resulting NE and its properties, which explains why the resulting solution is near-optimal. Section V presents numerical results comparing the centralized and distributed allocations, and conclusions are drawn in Section VI.

II. Related Work

The problem of cloud vs. edge server selection can be equivalently referred to as *computation offloading* or also *request routing*. In [2], [11], a taxonomy of different approaches to this problem is presented, according to which a centralized offloading scheme makes decisions about directing traffic to either edge or cloud through a single agent that may or may not use information on the status of the servers, according to which the policy is called either *stateful* or *stateless*. In [6] we showed that ideal stateful policies, with access to instantaneous information on either of the available servers or both of them, can outperform stateless policies. However, stateful policies are prone to errors, and small uncertainty on the status of the servers dramatically reduces their performance, well below the level that can be achieved by stateless and simple policies. Indeed, the advantage of stateless policies is that they are not affected by state estimate errors by definition. In that work, we only considered centralized policies, whereas here we discuss distributed implementations with their specific challenges.

Making request routing a distributed and selfish approach has been studied in various works, e.g., [7], [12]–[16]. In general, selfish routing leads to performance degradation, and factors like network topology [15] and load [16] play an important role in performance degradation. However, the authors of [17] show via analysis and experiments that the performance loss is minimized if selfish players have little context information, which means that conveying too much path-load context information to service customers is counterproductive when they can make decisions on their own. This justifies why here we focus on stateless selfish routing strategies and do not investigate more complex operational scenarios, although it is known that providing appropriately "curated" and "persuasive" data to routing deciders can improve performance [18], [19].

The optimization of offloading has been subject of many other investigations. For example, [20] proposes a scheduling algorithm whose objective is to minimize the task execution time. In other works, offloading is optimized together with other aspects, such as power allocation [3], [21]. We can say that our approach is related to the research line that compares global (centralized) with individual (decentralized) optimizations [22]. This problem has been addressed in many contexts such as communication services, queueing systems, transport optimization, and more (a non-exhaustive list of contributions includes [23]–[28]). In all these works, and also in our contribution, the comparison between these two approaches is carried out by using the concept of PoA proposed by [29] seen as a metric to quantify how much does system performance suffer from the lack of regulation.

III. Preliminaries

The structure of the system considered in our investigation is as follows. We assume that mobile users (UEs) of a network slice are connected through a random access channel (RACH) to a base station (BS) that employs two separate network processors for uplink and downlink messages. The RACH is assumed to use dedicated resources, so it is seen as isolated from other network slices on the same BS. The BS is, in turn, attached to a backhaul (BH), where an edge server is directly connected and can be used for computing tasks requested by the UEs, albeit with limited capacity. Alternatively, task can be sent from the BH through the transport network and the Internet to a physically distant but more powerful cloud server [4], [7]. Fig. 1 illustrates the reference scenario used in this work, with the backhaul represented as a simple logical switch although it could be an optical ring. Transport and core of the cellular network, jointly with the Internet, are symbolically represented as a transit cloud, which may include several high-speed routing devices. Uplink and downlink messages are in different colors because they carry different information.

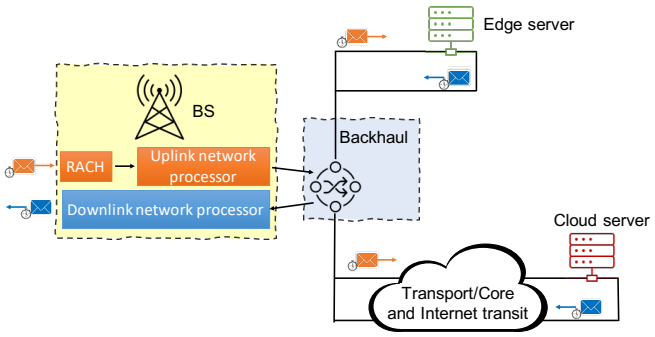

Fig. 1. Reference scenario

For simplicity, only one server of each kind (cloud or edge) is considered, but the analysis can be extended to multiple servers of the same type. Also, our model allows for inserting background traffic whenever needed. This would be necessary whenever a quantitative comparison with practical implementations is performed because the presence of background traffic is unavoidable in any real-world scenario, such as a testbed, and sometimes its impact is significant [30]. However, in the present evaluations, the background traffic does not affect the discussion that we bring forward related to the distributed implementation of the server selection policies.

UEs are assumed to generate service requests with rate λ_u. If these requests are directed to a server whose capacity is saturated, then the request is discarded, and we consider this to be a *loss* event. Moreover, we impose a delay constraint on the execution of the request so that a *failure* can occur both if the request is discarded and if a non-discarded request violates the timeout condition (i.e., if the result of the computation associated with the request does not reach the UE before the timeout expiration). This means that failures happen with probability P_{fail} that is greater than the loss probability P_{loss} as failures are a broader condition. We also distinguish between $\mathsf{P}_{\text{fail}}^{(\text{E})}$ and $\mathsf{P}_{\text{fail}}^{(\text{C})}$ (or analogously $\mathsf{P}_{\text{loss}}^{(\text{E})}$ and $\mathsf{P}_{\text{loss}}^{(\text{C})}$) as the conditional probability of failures (or losses) that have been directed to the edge or to the cloud server, respectively.

The foundation of the analysis is to characterize all the system components as work-conserving FIFO queues and therefore evaluate the experienced latency as the convolution of the delay terms at each step. If we represent all the delay terms of these individual components through the Laplace-

Stjeltjes transform (LST) of their probability distribution function (pdf), the resulting overall delay has an LST obtained by a chain multiplication of all terms.

Thus, if we assume that each request of UE i is sent with probabilities α_i or $1-\alpha_i$ to the edge or cloud server, respectively, the pdf of the overall delay of a request of UE i, conditioned to it being successful and not lost due to buffer overflow, has an LST $\widehat{f}_T(s)$ that can be written as

$$\begin{aligned}\widehat{f}_T(s) =& \widehat{f}_{\mathrm{R}}(s)\,\widehat{f}_{\mathrm{L_u}}(s)\,\widehat{f}_{\mathrm{L_d}}(s)\,\widehat{f}_{\mathrm{B_u}}(s)\,\widehat{f}_{\mathrm{B_d}}(s) \\ &\cdot \Big(\alpha_i\big(1-\mathsf{P}^{(\mathrm{E})}_{\mathrm{loss}}\big)\widehat{f}_{\mathrm{E}}(s) \\ &\quad + (1-\alpha_i)\big(1-\mathsf{P}^{(\mathrm{C})}_{\mathrm{loss}}\big)\widehat{f}_{\mathrm{T_u}}(s)\,\widehat{f}_{\mathrm{T_d}}(s)\,\widehat{f}_{\mathrm{C}}(s)\Big) \quad (1)\end{aligned}$$

where $\widehat{f}(s)$ denotes an LST of the pdf of a delay term, with subscripts referring to the RACH (R), the link between the BS and the network processor ($\mathrm{L_u}$ and $\mathrm{L_d}$ for uplink and downlink, respectively), the BH connection ($\mathrm{B_u}$ and $\mathrm{B_d}$ for uplink and downlink, respectively), the transport network ($\mathrm{T_u}$ and $\mathrm{T_d}$ for uplink and downlink, respectively), and the time spent in the edge or in the cloud server (E and C, respectively).

The failure probability $\mathsf{P}_{\mathrm{fail}}$ of UE i is the sum of the loss probability observed as a weighted average over edge and cloud servers, i.e.,

$$\mathsf{P}_{\mathrm{loss}} = \alpha_i \mathsf{P}^{(\mathrm{E})}_{\mathrm{loss}} + (1-\alpha_i)\mathsf{P}^{(\mathrm{C})}_{\mathrm{loss}}, \quad (2)$$

and the probability that the service exceeds the timeout:

$$\begin{aligned}\mathsf{P}_{\mathrm{fail}} &= \mathsf{P}_{\mathrm{loss}} + \big(1-\mathsf{P}_{\mathrm{loss}}\big)\big(1-F_T(T_O)\big) \quad (3)\\ &= 1 - F_T(T_O)\big(1-\mathsf{P}_{\mathrm{loss}}\big)\end{aligned}$$

where T_O is the timeout value and $F_T(t)$ is the cumulative distribution function of the latency computed as the inverse LST of $\widehat{f}_T(s)/s$ via numerical techniques.[1] Notice that, as described in [6], loss at edge and cloud, and timeout probabilities, depend on the average of α_i across all UEs.

Now, we sketch how these components can be modeled, with a more detailed analysis that can be found in [6]. The RACH model is taken from [30] and considers that user requests experience a delay due to the transmission over the channel, possibly encountering collisions and subsequent backoffs. We take a sufficiently high number of allowed retransmission attempts to prevent the RACH from losing service requests by itself. However, losses can occur due to the capacity of the servers being saturated, as previously discussed. Thus, from [6] we get the following approximate expression for the LST of the sojourn time in the RACH:

$$\widehat{f}_T(s) = \frac{1-e^{-s(T_x+W_x)}}{s(T_x+W_x)} \sum_{i=1}^{k_x} \frac{1-e^{-i}}{e^{i(i-1)/2}} \left(\frac{e^{-sT_x}}{1+\bar{\tau}_B}\right) \quad (4)$$

where T_x, W_x, k_x, and $\bar{\tau}_B$ are RACH parameters corresponding to the maximum time to reply to a RACH request, the maximum time to establish a connection after a RACH exchange, the maximum number of transmission attempts, and the average backoff time in case of collisions. In (4), we consider a traffic-independent expression for the probability that the RACH request is successful in i attempts, which holds true if RACH losses are negligible and we adopt power ramping over multiple transmission attempts.

From the implementation standpoint, the BH consists of individual links and dedicated resources interconnecting the BS, the edge server, and the transport network that bridges to the cloud. The BH can be considered reliable, i.e., not introducing any loss, and with high capacity. Both the edge and the cloud servers are assumed to run on virtual machines (VMs), whose numbers are n_{E} and n_{C}, respectively. Service requests coming from UEs can be allocated to one VM equivalently located in either server, after being queued in a buffer with finite capacity. The maximum number of requests that can be queued at the edge or cloud server is k_{E} and k_{C}, respectively. Hence, a loss event happens if the number of allocated requests exceeds the buffer capacity.

We model the network processor as an M/D/1-PS queue, because time-frequency resources of a base station are shared among all active transmissions, which can occur in parallel. All the intermediate queues (backhaul and transport networks) are more simplistically modeled as M/M/1 queues because of the serial nature of transmission over many wired link technologies [11]. For all these systems, we just repeat this model for both uplink and downlink directions. Finally, the edge and cloud servers are represented as M/M/n_{E}/k_{E} and M/M/n_{C}/k_{C} queues, respectively. The multi-server nature of edge and cloud queues reflects the fact that they are essentially computing stations with multiple processors available for service [5]. This promptly gives us expressions for the LST terms in (1). Additionally, the loss probabilities $P^{(\mathrm{E})}_{\mathrm{loss}}$ and $P^{(\mathrm{C})}_{\mathrm{loss}}$ are computed from the probabilities that the buffer at the queues is full, that is, the queue is occupied by k_{E} or k_{C} requests, respectively.

Note that we model the (uplink and downlink) network processor delays as deterministic because we consider the case of a lightly loaded BS slice. On the contrary, we model computing times at both the edge and cloud servers with exponential distributions because of their intrinsic variability and because of our focus on the latency introduced by those computing elements.

IV. Game theoretic model

The idea of RANDALPH is to set the same $\alpha_i = \alpha$ for all service requests. This value is computed in a centralized way as a global optimum so that all service requests are randomly directed to either the edge or the cloud server with respective probabilities α and $1-\alpha$, without constant monitoring of the network state, which makes it suitable for a simpler implementation, whose efficiency is still comparable to more complex stateful policies [6].

However, in light of the drawbacks of a centralized computation, we want to explore a game theoretic approach to determine whether the α–values can be efficiently computed through a distributed evaluation by individual UEs seen as selfish (i.e., strategic) agents. This results in the game theoretic procedure that we named GANDALPH. The differences with the original centralized approach are subtle but essential. Firstly, we consider an individual value α_i for each service request i. Since we consider continuous α-values, it does not really matter if i identifies the atomic service request or the UEs as this can be framed as either a fine-grained choice of individual tasks or a probabilistic assignment of all the tasks of the same user.

[1] The division by s in the LST domain corresponds to an integration in the probability domain, so that while by inverting $\widehat{f}_T(s)$ one would obtain a probability distribution function, by inverting $\widehat{f}_T(s)/s$ will obtain the corresponding cumulative distribution function.

Moreover, we do not assume any communication exchange between the agents, so they all act independently. In the jargon of game theory, we consider a *static game of complete information* [8], [24], where all the UEs are equally aware of the network parameters but cannot communicate their choices. This sets a difference with other game theoretic approaches based on auctions or bargaining [27], [31], in that our simpler implementation does not require any subsequent interaction. Finally, the choice of each agent is selfish, i.e., just driven toward the minimization of the failure probability P_{fail} experienced by that player.

Notably, under this approach all users will still choose the same α for symmetry reasons. However, the rationale of RANDALPH and GANDALPH is different. In the former, α is a value chosen to be the same by a central optimizer so as to achieve the best performance. In the latter, each strategic player chooses α_i so as to optimize a *selfish* goal while at the same time being aware that all other players will do the same [10]. This results in a local optimum from the individual player's perspective, and NEs are found as the points without incentives for unilateral deviation towards another α_i (while the other α_j, $j \neq i$, are kept as they are).

The analysis summarized in the previous section strongly motivates this approach to be sensible. In practice, the failure probability P_{fail} that a single player can expect to get depends on α_i through a dependence that, avoiding the analytical intricacies, can be summarized as (2), which uses the total probability on the conditions of choosing the edge or the cloud server. For a selfish player, the best choice of α_i happens where the first derivative of P_{fail} is zero, but it is immediate to see that

$$\frac{\partial \mathsf{P}_{\text{fail}}}{\partial \alpha_i} = \mathsf{P}_{\text{fail}}^{(\text{E})} - \mathsf{P}_{\text{fail}}^{(\text{C})} + \alpha_i \frac{\partial \mathsf{P}_{\text{fail}}^{(\text{E})}}{\partial \alpha_i} + (1-\alpha_i)\frac{\partial \mathsf{P}_{\text{fail}}^{(\text{C})}}{\partial \alpha_i}. \quad (5)$$

Therefore, equalizing the failure probability of edge and cloud, e.g., with a load balancing approach, is not necessarily enough to reach a NE. However, if the failure probabilities of edge and cloud can be equalized in a point where both servers are far from saturation, then that point would be near the NE, because the partial derivatives are close to zero. Hence, the analysis reveals that the NE would lead the system to operate almost like in a centralized load balancing scenario when the system load is low. Under those circumstances, load balancing is indeed near-optimal and very robust [6]. Instead, the operating point would progressively deviate from a balanced assignment as the load increases and the partial derivative terms in (5) become predominant. This must not surprise because, at high load, a load balance policy cannot work well as it would fairly lead all jobs to fail.

In general, the partial derivative of the failure probability must be positive for the edge at any α_i, while at the cloud it must be negative. Thus, the difference $\mathsf{P}_{\text{fail}}^{(\text{E})} - \mathsf{P}_{\text{fail}}^{(\text{C})}$ is a monotonic increasing function of α_i, ranging from a negative value $-\mathsf{P}_{\text{fail}}^{(\text{C})}|_{\alpha_i=0}$ to a positive value $\mathsf{P}_{\text{fail}}^{(\text{E})}|_{\alpha_i=1}$. The weighted sum of the partial derivatives in (5) has a similar behavior, but it goes from a positive to a negative value instead. These considerations support the uniqueness of the NE as a point where the failure probabilities at the cloud and the edge must be different, unless edge and cloud servers have the same characteristics. Besides, the role of the partial derivatives is important to determine which server (edge or cloud) must experience a higher loss rate at the NE.

A key observation from the results of [6] is that P_{fail} is relatively flat around its minimum in practical situations. This implies that: (i) a local search from an individual standpoint is likely to achieve near-optimal values of α_i; (ii) even if α_i is not exactly chosen by each individual user as the optimal value of α for RANDALPH, the resulting P_{fail} is still likely to be near-optimal.

Moreover, we are interested in obtaining a small average failure probability, otherwise performance cannot be good and the system operation becomes pointless. Thus, at least one server must be far from saturation, and if one server approaches saturation, it must receive just a little portion of traffic. Hence, all terms of (5) are small or slowly changing with α_i, which explains why P_{fail} is flat around its minimum. In particular, the last term describes the influence of a single request on the failures in the cloud server, which again is sensible to assume to be minimal. The same can hold for the third term but, while we can argue that the edge server can be more sensitive, we remark at the same time that if this is the case, it is also likely that α_i is small, thereby causing the term to be close to 0.

As argued above, the NE of GANDALPH is necessarily unique because of the monotonic trend that the failure probability of any selfish user must have if it deviates from the behavior of the other UEs – in particular, it grows if α_i goes to either 0 or 1, meaning that only either the cloud or the edge server is used. To quantitatively juxtapose the approaches, we can compare the two values of α under a centralized or distributed management, i.e., the optimal value chosen by RANDALPH and the NE of GANDALPH. A further comparison can quantify the impact that a possible difference of these α values has on the failure probability. Thus, we compute the PoA [10], [29] as $\text{PoA} = \mathsf{P}_{\text{fail}}^{\text{Nash}} / \mathsf{P}_{\text{fail}}^{\text{Coordinated}}$ with a clear meaning of the superscripts.

V. Evaluation

We designed a set of experiments to compare coordinated and selfish routing decision strategies in the reference scenario considered in this paper. In particular, we analyze and compare performance figures in terms of failure probability P_{fail}. The difference between RANDALPH and GANDALPH will be expressed in terms of PoA. We will also show how the two different approaches result in routing strategies that can diverge substantially.

A. Coordinated optimum and NE search

The optimal configuration for RANDALPH is the value of edge routing probability α, commonly adopted by all users, that minimizes the failure probability P_{fail}. Since this optimization problem is not convex in general, as we will show with an example at the beginning of the numerical evaluation subsection, we resort to a discretized search. In practice, we run a brute force search with resolution 10^{-3} on the value of α. The search requires a few hours of computation of a dedicated 3 GHz processor. We do not optimize the search of a coordinated optimum, because the object of the work is not optimization per se, rather the evaluation of the performance and the PoA of a distributed implementation of the server selection [22].

For what concerns the NE, using a brute force search on condition (5) is impractical, since the computation of partial derivatives is prone to numerical errors and would

in any case involve the computation of several values of failure probability for each candidate value of edge routing probability, so as to be able to estimate the partial derivative function. Instead, we resort to design a search algorithm which is similar to the well known binary search. To explain how our algorithm works, first of all, consider that P_{fail} observed by a selfish user has a single minimum as her routing α_i changes, i.e., with respect to the variations of edge routing probability of the selfish user (while the rest of users do not change their strategy). To see that, consider the only three possible cases that can occur when the traffic of the selfish user is routed in full to the cloud, i.e., the failure probability at the edge, $\mathsf{P}^{(\text{E})}_{\text{fail}}|_{\alpha_i=1}$ is smaller, equal, or larger than the failure probability at the cloud, $\mathsf{P}^{(\text{C})}_{\text{fail}}|_{\alpha_i=1}$. In the first case, offloading some traffic from cloud to edge is beneficial, although beyond some point the offload can become counterproductive, because $\mathsf{P}^{(\text{E})}_{\text{fail}}$ can only increase while $\mathsf{P}^{(\text{C})}_{\text{fail}}$ can only decrease. The last case is similar, but this time the offload cannot help and the minimum is observed at $\alpha_i = 1$. If instead the failure probabilities are identical, we would need to compare the partial derivatives of the conditional failure probabilities, to see which failure probability changes faster, and would reach the same conclusions as in the other two cases. In all cases, the value of P_{fail} observed by a selfish user has a single absolute minimum value for $\alpha_i \in [0, 1]$.

From the above considerations, we infer that a selfish user would only deviate her strategy toward a given direction, and in so doing, she would greedily find her best routing probability. Moreover, as users have the same characteristics and are rational, in a distributed implementation scenario they would all have the same incentive to move like the selfish users we have taken as reference in the discussion so far. This means that all users would move toward the same direction and iteratively converge to a NE point. In particular, when we have identified the direction that a selfish user would take, we can assume that the entire set of users will move in the same direction.

We therefore implement the following search for the NE configuration. We start by considering a candidate interval for α and take the central value of the interval as the current routing probability of all users. The initial interval can simply be the entire space of strategies, i.e., $[0, 1]$. We then split the interval into three equally sized adjacent sub-intervals and compute the average value of the failure probability of a selfish user under the hypothesis that she can decide to move toward a point in any of the three intervals. We only take a few (3 to 5) evenly spaced samples per sub-interval, to compute averages and identify the smallest of them. This shows in which direction a selfish user would move if starting from the center of the interval. Since, as previously noted, the rest of users would follow the target selfish user, we can next repeat the procedure by considering a half-sized search interval that corresponds to either (a) the right half of the original interval if the selfish user had an incentive to increase α_i, (b) the left half is the incentive was toward decreasing α_i, or (c) the central segment otherwise. Using three possible partially overlapped intervals rather than two separate intervals makes the search robust to numerical errors which might appear when dealing with failure probability values of the order of 10^{-6} or less, and adjusting the routing probability by 10^{-3} or smaller steps. Those errors might otherwise cause the search to remain stuck in the wrong interval, which is relevant for cases in which the failure probability is relatively flat and can be small as required by many commercial applications. The procedure continues until the search interval size becomes smaller than a threshold (10^{-4}, in the numerical results shown in what follows). The middle point of the last identified interval is considered as the NE point, and the associated failure probability is computed.

TABLE I
PARAMETERS USED IN THE NUMERICAL EVALUATION

Parameter	Notation	Default value
Number of UEs	n_u	50
Service request rate per UE	λ_u	60 s^{-1}
Number of servers at edge	n_{E}	1
Buffer size at edge	k_{E}	10 requests
Number of servers at cloud	n_{C}	10
Buffer size at cloud	k_{C}	50 requests
Average request service time	μ^{-1}	5 ms
Round trip time from UE to edge		26.955 ms
Round trip time from UE to cloud		52.498 ms
Number of RACH preambles		54
Max number of RACH retransmissions	k_x	10
RACH retransmission timeout		10 ms
Uplink packet size		2000 b
Downlink packet size		4000 b
Uplink radio slice capacity		10 Mb/s
Downlink radio slice capacity		25 Mb/s
Uplink BH slice capacity		20 Mb/s
Downlink BH slice capacity		35 Mb/s
Core network slice capacity		100 Mb/s
Service timeout	T_O	100 ms

B. Evaluation scenario

To illustrate the behavior of GANDALPH, we evaluate its performance in the concrete scenario described next, which allows us to evaluate the impact of several parameters like the system load and the granularity at which UEs contribute to the system load, the distance of the cloud service and its capacity. We consider n_u UEs, each issuing λ_u requests per second (req/s), on average. In the plots, we use the value of the system load ρ, which is the workload of the system, calculated as the ratio of the aggregate offered traffic $n_u\lambda_u$ divided by the total service capacity of the system $(n_{\text{C}} + n_{\text{E}})\mu$, which in turn depends on the number of VMs in the edge and cloud servers (n_{E} and n_{C}, respectively), and their capacity μ.

All UEs see the same distance to an edge server, since users are connected to the same BS, hence access the same backhaul. The edge server runs a single server, modeled as an $M/M/1/k_{\text{E}}$ queue with a finite buffer space (k_{E}=10 requests – one in service and up to 9 waiting – in the numerical evaluations) with an exponential service at rate μ=200 requests per second. In other words, a request takes on average 5 ms to be served, thus representing short jobs typical of mobile applications that require network assistance for parsing the local context and making informed decisions, e.g., in assisted driving applications, steering of UAV fleets, and so on [32].

The round-trip-time (RTT) between UEs and the edge server is the same as between UEs and BH, which we set to 24.875 ms, plus an almost negligible extra delay in accessing the edge server from the BH (we use 0.08 ms as the time needed to cross the BH) plus 2 ms to account for internal edge data center latency. The resulting RTT is 26.955 ms. These RTT values are the ones observed in a testbed built to evaluate RANDALPH in [6]. The cloud is

an $M/M/n_C/k_C$ server with n_C=10 servers (each akin to the edge server) and k_C=50 maximum requests by default (including requests under service), although we also explore other values in specific experiments shown later. The cloud is reachable through the same BH, and the RTT between the BH and the cloud is set to 24.543 ms unless otherwise specified (again, we use the values observed in [6]). We also add 1 ms to account for internal delay in the cloud data center. Therefore, the total RTT between UEs and cloud is $26.955 + 24.543 + 1 = 52.498$ ms, without accounting for queueing and processing at the cloud.

All requests go through the RACH procedure for which they use 54 orthogonal preambles with a RACH opportunity every millisecond. The maximum number of RACH transmission attempts is k_x=10, enough to guarantee full reliability of the channel, since users adopt a power ramping algorithm to progressively increase the transmission power of their RACH preambles after each failure. Retries are spaced according to random backoff times with average duration 10 ms. With the adopted configuration, the probability that the RACH will cause a request loss is below $3 \cdot 10^{-18}$ and can be therefore neglected.

After success on the RACH, requests are sent in packets of 2000 bits and, as discussed in Section III, served by a network processor at the BS, modeled as an FCFS queue, after which they move to another queue representing a BH from which requests are dispatched to either the edge server or to another queue representing the core network segment between the BH and the cloud server. The mentioned queues have infinite buffer space, and the RACH does not cause losses, so that all requests eventually reach either the edge or the cloud server. Downlink transmissions from the edge or cloud server follow the respective inverse path, and are sent in individual packets of 4000 bits. The capacity of the RAN slice radio link is 10 Mb/s in uplink and 25 Mb/s in downlink, representing a BS slice dedicated to the considered service. The BH capacity is 20 and 35 Mb/s in uplink and downlink, respectively. These values consider a service that accesses only a slice of the backhaul resources. The core network slice capacity is 100 Mb/s in both uplink and downlink.

With these system parameters, the performance bottleneck is at the edge and cloud sites, while network elements only cause random delays. This implies that system performance is determined by edge/cloud losses and service timeouts. The value of the latter is set to T_O=100 ms, in line with the requirements of real-time services for autonomous driving, online gaming, and augmented reality, just to mention a few use cases [32], [33].

C. Numerical evaluation

We evaluate the NE of GANDALPH as a function of the system load ρ, the number of cloud servers n_C and the round-trip-time distance of the cloud from UEs, as well as the number of UEs in the system. We compare the NE to the optimum α computed by a centralized strategy that minimizes the failure probability. The NE is numerically found as the value of a common edge selection probability α_i=α_{Nash} for which the individual failure probability does not improve by selfishly deviating from it, according to the procedure described in Section V-A. Numerical results produced by using Matlab compute the loss probability of edge and cloud servers and the LST of the overall RTT delay of served

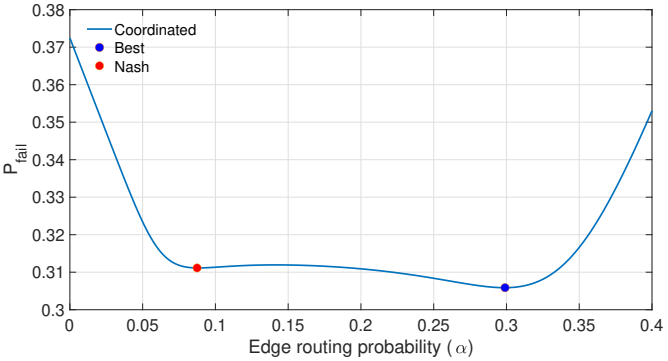

Fig. 2. Example of non-convex dependency of failure probability on edge probability routing. The "Coordinated" curve represents the failure probability vs. the edge routing probability achieved with RANDALPH. "Best" and "Nash" mark the optimal coordinated strategy and the NE found with GANDALPH, respectively. All network parameters are according to Table I.

requests resulting from the load partition between edge and cloud imposed by the chosen α_i, so as to be used in (3). Since with a centralized approach or at equilibrium all users adopt the same value of α, and because we only consider GANDALPH at the NE, in what follows we drop the index i from the notation.

Table I summarizes the default parameters used in the numerical evaluation.

We start by showing that the optimization of RANDALPH is not a convex problem and that multiple local minima can exist. Fig. 2 shows the case with default parameters. The figure zooms into the interval in which the optimum can be found, which is marked with a blue dot on the curve of P_{fail} vs α. he interval shown is not small, and, as can be seen, the failure probability is somehow flat over a large portion of the interval, which makes the search tedious. Two local minima are clearly visible, and the NE point (marked with a red dot) is close to one of them, although unfortunately not the one giving the absolute minimum. Differences in failure probabilities at the two minima and the NE are however quite small. Here, to clearly show non-convexity, we consider a case with very high load. Indeed, the total offered traffic is $n_u\,\lambda_u$=3000 req/s, while the aggregate service capacity of edge and cloud servers is $(n_C{+}n_E)/0.005$=2200 req/s, i.e., ρ=1.36. This explains why the failure probability observed in the figure is high. In particular, since at least 3000−2200=800 req/s cannot be served (without considering timeouts), the bare minimum loss to observe in the system is 800/3000=0.267. Values reported in the figure, just above 0.3, are therefore not surprising. Next, we consider a more extensive range of loads that implies also more reasonable failure probabilities.

Fig. 3 compares the values of α in a scenario with all default parameters except λ_u, which varies from 440 req/s (ρ=0.2) to 2640 req/s (ρ=1.2). Results were obtained through the centralized solution (i.e., RANDALPH) or the NE where the edge routing probability is chosen by each UE (i.e., GANDALPH). The two curves have a similar trend, with just a slight divergence for values of the system load ρ higher than 1, which have limited relevance.

For the same scenario of Fig. 3, Fig. 4 compares instead the resulting value of P_{fail}, which is actually more interesting than the sheer value of α. We can see that the two selection policies achieve very close results. The corresponding PoA (i.e., the ratio of the two curves) is shown in Fig. 5, and it is found to be extremely limited, always below a decrease of

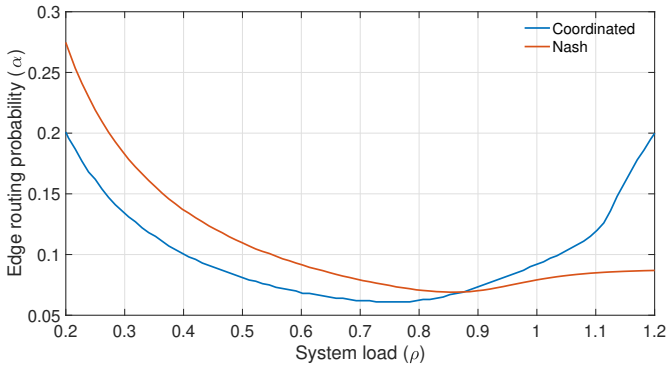

Fig. 3. Routing strategies of optimized RANDALPH (Coordinated) and GANDALPH (Nash) with the parameter values in Table I and variable system load (obtained by varying λ_u only).

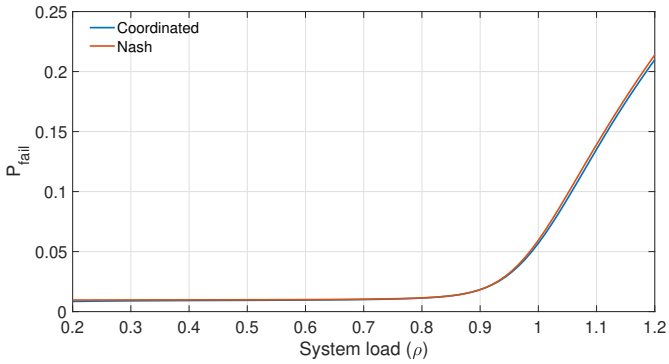

Fig. 4. Failure probability achieved with optimized RANDALPH (Coordinated) and GANDALPH (Nash) with the parameter values in Table I and variable system load (obtained by varying λ_u only).

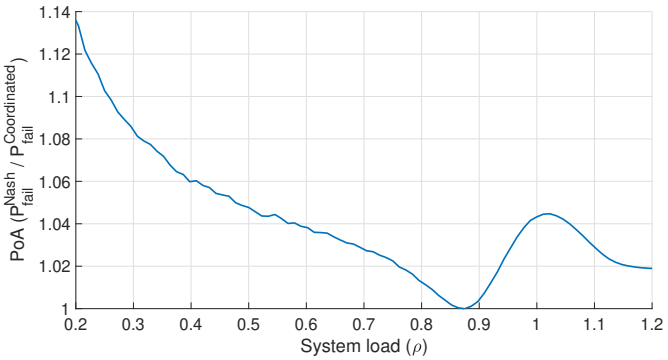

Fig. 5. Price of anarchy of GANDALPH over the best allocation achieved with RANDALPH with the parameter values in Table I and variable system load (obtained by varying λ_u only).

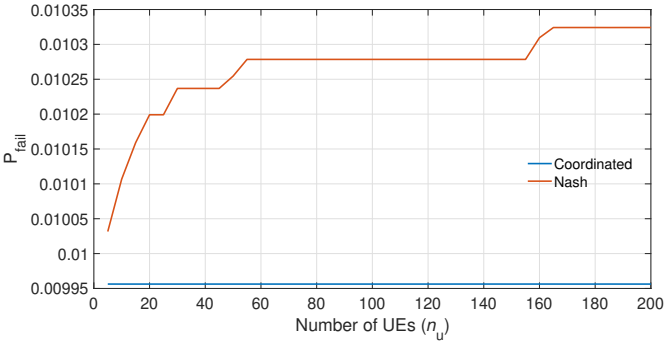

Fig. 6. Failure probability with optimized RANDALPH (Coordinated) vs. GANDALPH (Nash) with a variable number of UEs at fixed aggregate traffic of $n_u\lambda_u$=1500 req/s, and the default parameter values in Table I (ρ=0.682).

efficiency of 10% for $\rho > 0.25$. The fact that the value is slightly higher for low values of ρ is not very relevant, as in this case P_{fail} is very close to zero. More interesting is the fact that the PoA attains its lowest values for a system load value close to 90%, which is a more realistic operation point.

Fig. 6 investigates the role of the granularity of a single UE. In particular, the figure shows results for a fixed total arrival rate of $n_u\lambda_u$=1500 req/s, i.e., a medium-high system load (ρ=0.682), as the number of UEs that generate the load increases and hence the importance of each user on the system economy progressively vanishes. The rest of parameters take their default values. As can be seen in the figure, the "size" of the UE (i.e., λ_u=1500/n_u) has little impact on performance, which empirically confirms that using α as the probability to route a single service request of a UE does not matter, especially when the number of UEs is high. To explain the result, note that only the aggregate traffic matters for RANDALPH, whereas, the efficiency of GANDALPH slightly decreases when more UEs are considered (each with a smaller size so that the aggregate traffic is the same). This follows from the principle known as the tragedy of the commons [10], i.e., the system efficiency decreases when the role of the individual in the community is less impactful, as selfish behaviors are encouraged. In this case, however, the efficiency loss is just marginal (note the values on the vertical axis).

Figs. 7 and 8 show the impact of the distance to the cloud, whose increasing value deteriorates performance, because it causes more failures. Here, we use n_u=60 and λ_u=250 req/s, and variable RTT between the cloud and the BH, while the rest of parameters take their default values. Notice that the system load in those figures is the same as in Fig. 6.

As illustrated in Fig. 7, the selfish allocation of GANDALPH pushes up the edge routing probability α, i.e., the probability that the edge server be selected, up to 100%. This value is reached when requests sent to the cloud are often violating the timeout, which occurs almost certainly as soon as the edge-cloud RTT approaches 73 ms, the RTT between UE and edge being just below 27 ms, and the timeout 100 ms. The distributed assignment GANDALPH is not particularly worse than the original centralized RANDALPH, once again keeping the loss of efficiency within 10%, as reported in Fig. 8. That figure also shows that the PoA diminishes at very high delay, which is not surprising since when the RTT is too high, no strategy can lead to good performance since almost all requests fail, either because they are routed to the cloud and violate the timeout constraint, as done by the coordinated approach, or because of the overflow of the edge queue to which they are routed by the selfish approach.

Finally, the role of the cloud capacity is explored in Figs. 9–12. In all figures, the capacity of the cloud is changed by varying the number of servers (n_C), but the first two plots consider a fixed offered traffic (n_u=60, λ_u=250 req/s, while ρ changes with n_C), whereas the second pair of figures considers a fixed number of UEs and system load (n_u=60, ρ=0.682, while λ_u changes with n_C). The number of servers used at the cloud is indicated as the number of VMs allocated to the service. In these experiments, the buffer space at the cloud also scales with the number of VMs, with a ratio 5:1, i.e., k_C=5 n_C. The rest of parameters are as described in Table I. The figures show that (i) edge routing probabilities obtained with centralized and distributed approaches are not distant (cf. Figs. 9 and 11), and (ii) the corresponding failure probabilities are even closer (cf. Figs. 10 and 12). In particular, Figs. 10 and 12 show that when the cloud

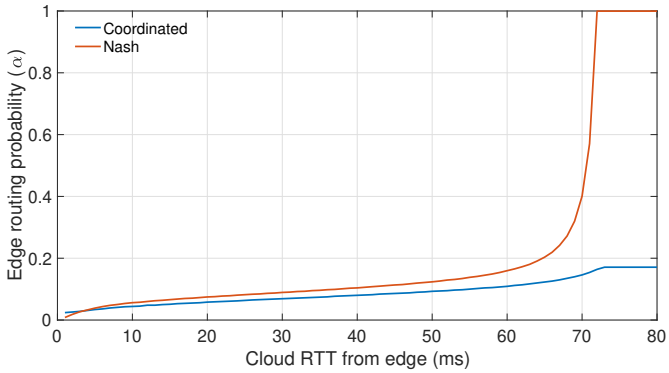

Fig. 7. Routing strategies of optimized centralized RANDALPH (Coordinated) and GANDALPH (Nash) vs the distance of the cloud from the edge, with n_u=60 UEs, λ_u=250 req/s (ρ=0.682) and other default parameters.

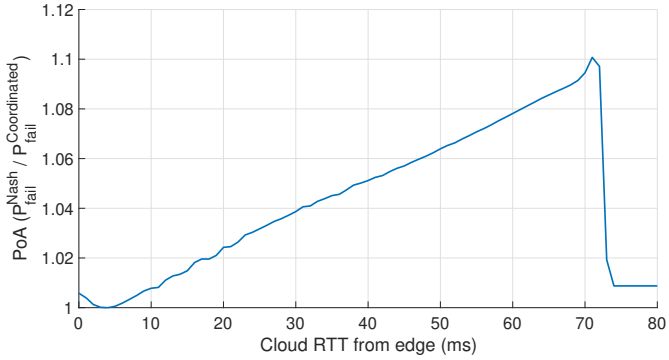

Fig. 8. PoA vs. the distance of the cloud, with n_u=60 UEs, λ_u=250 req/s (ρ=0.682) and other default parameters.

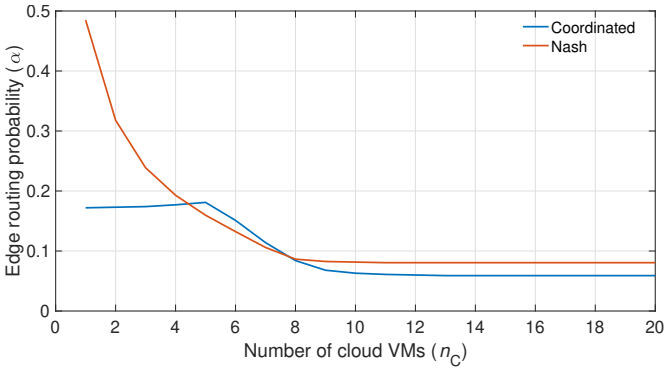

Fig. 9. Best strategy of optimized RANDALPH (Coordinated) and GANDALPH (Nash) with variable cloud capacity (i.e., variable n_C, the number of virtual machines (VMs)), with fixed aggregate traffic of 1500 req/s generated by n_u=60 UEs at λ_u=250 req/s. The buffer space of the cloud is k_C=5 n_C. Other parameters are as in Table I. Network capacity is less than offered traffic, hence $\rho > 1$, as long as the number of VMs is below 7.

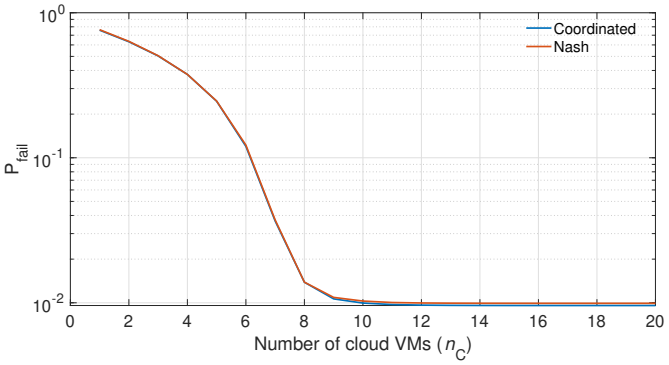

Fig. 10. Failure probability of optimized RANDALPH (Coordinated) and GANDALPH (Nash) for the same scenario as for Fig. 9.

contains fewer servers, the performance of both GANDALPH and RANDALPH deteriorates, and both the coordinated and distributed approaches achieve the same level of quality. They also dictate a more frequent selection of the edge server, save for GANDALPH under fixed aggregate traffic, where a plateau is reached (see Fig. 9). However, the resulting differences in terms of failure probability (Figs. 10 and 12) are minimal.

With fixed request arrival rate ($n_u\,\lambda_u$=1500 req/s) and variable cloud capacity (Figs. 9 and 10), the load ρ is below 1 with 7 or more VMs, because each VM contributes with a capacity of 200 services per second (and the edge has one VM). This explains the different behaviors of the curves before and after the point at 7 VMs. The behavior of the algorithms with only one VM at the cloud is interesting. In that case, the capacities of edge and cloud are the same, and the network is largely overloaded (1500 req/s with a total capacity of 400 services per second). The edge and cloud servers are in this case almost equivalent, since whenever a service request reaches a server, it has a high chance of being lost due to a full buffer, but if the request is accepted, it is very likely to be the last in the buffer and, thus, have in front 9 services (one in progress and 8 in the buffer) at the edge, and 4 services (one in progress and 3 in the buffer) at the cloud. On average this implies 45 ms waiting delay at the edge and 20 at the cloud, plus 5 ms for the request service, plus the RTT from/to the UE. This means 76.955 ms at the edge server ($24.875+0.08+2+45+5$) and 75.498 ms at the cloud server ($24.875+0.08+24.543+1+20+5$). Hence, the cloud server is slightly better, and we see in Fig. 9 that its choice probability is slightly higher than 0.5 with the selfish approach. Instead, the best coordinated strategy consists in sending only a small part of the traffic to the edge, and let the rest be lost. By pursuing the more appealing solution, GANDALPH incurs higher losses, a few percents more than RANDALPH, although this effect is not well visible in Fig. 10 due to the adopted log scale.

Conversely, with fixed system load and variable offered traffic (Figs. 11 and 12), the differences between GANDALPH and RANDALPH are less important. In this case, at the NE point, the edge routing probability is always a bit higher than at the best coordinated operation point found with RANDALPH. This happens because, being $\rho < 1$ and the edge not saturated, any selfish UE sees an incentive in offloading some traffic from cloud to edge, thus sparing some RTT and, in turn, reducing the timeout probability.

VI. Conclusions

We presented a game theoretic analysis of a randomized policy for edge/cloud server selection to satisfy latency-constrained computing-based service requests. Our objective is to quantify the efficiency of a distributed implementation of the policy as compared to a centralized optimization of the server selection probabilities. Our results are extremely encouraging, since they show that a selfish allocation by strategic agents, through an algorithm called GANDALPH, behaves very similar to the centralized optimal policy RANDALPH, with values of α that are close, at least when the system is not overloaded, and resulting performance metrics that are even closer, in all cases, for the two approaches.

These results are quite a relevant finding, since they prove the effectiveness of a fully distributed implementation of

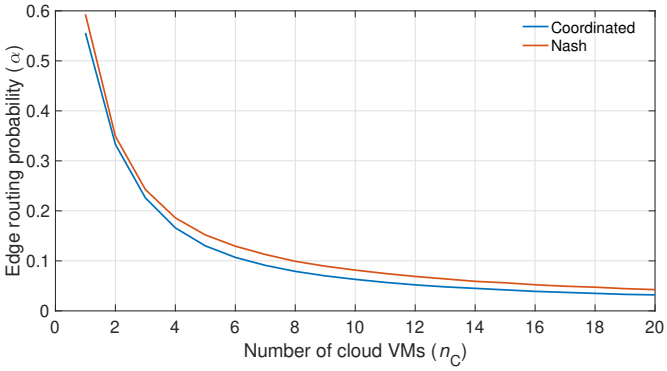

Fig. 11. Best strategy of optimized RANDALPH (Coordinated) and GANDALPH (Nash) with variable cloud capacity (i.e., variable n_C, the number of virtual machines (VMs)), with fixed system load ρ=0.682 generated by n_u=60 UEs and variable λ_u. The buffer space of the cloud is k_C=5 n_C. Other parameters are as in Table I.

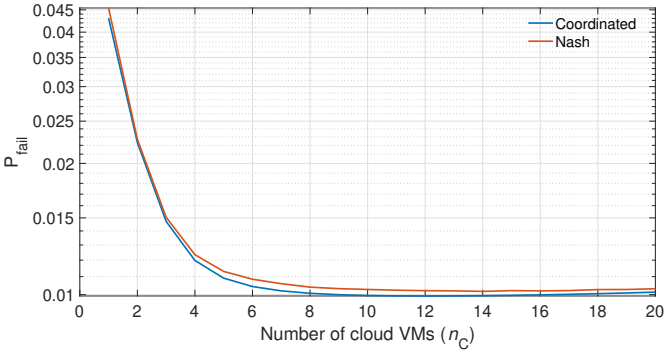

Fig. 12. Failure probability of optimized RANDALPH (Coordinated) and GANDALPH (Nash) for the same scenario as for Fig. 11.

server selection policies based on global parameters, without the need for constant network monitoring, thus achieving an implementation which can be at the same time simple and efficient, as claimed by the authors of [31].

Future work may include validation of these findings in realistic testbeds and the development of a platform implementation, based on network slicing principles as in [30].

Acknowledgments

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 by the RESTART Program, financed by the Italian government with the resources of the Italian Recovery, Transformation and Resilience Plan – Mission 4, Component 2, Investment 1.3, theme 14 "Telecommunications of the future."

References

[1] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, 2017.

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

[3] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, 2017.

[4] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, 2017.

[5] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, 2018.

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

[7] G. Panek, I. Fajjari, H. Tarasiuk, A. Bousselmi, and T. Toukabri, "Application relocation in an edge-enabled 5G system: Use cases, architecture, and challenges," *IEEE Commun. Mag.*, vol. 60, no. 8, pp. 28–34, 2022.

[8] G. Quer, F. Librino, L. Canzian, L. Badia, and M. Zorzi, "Inter-network cooperation exploiting game theory and Bayesian networks," *IEEE Trans. Commun.*, vol. 61, no. 10, pp. 4310–4321, 2013.

[9] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 260–288, 2018.

[10] L. Prospero, R. Costa, and L. Badia, "Resource sharing in the Internet of Things and selfish behaviors of the agents," *IEEE Trans. Circuits Syst. II*, vol. 68, no. 12, pp. 3488–3492, Dec. 2021.

[11] F. Sufyan and A. Banerjee, "Computation offloading for distributed mobile edge computing network: A multiobjective approach," *IEEE Access*, vol. 8, pp. 149 915–149 930, 2020.

[12] 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.

[13] W. Xu and J. Rexford, "MIRO: Multi-path interdomain routing," in *Proc. ACM SIGCOMM*, 2006, p. 171–182.

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

[15] F. Benita, V. Bilò, B. Monnot, G. Piliouras, and C. Vinci, "Data-driven models of selfish routing: why price of anarchy does depend on network topology," in *Proc. WINE*. Springer, 2020, pp. 252–265.

[16] R. Colini-Baldeschi, R. Cominetti, P. Mertikopoulos, and M. Scarsini, "When is selfish routing bad? The price of anarchy in light and heavy traffic," *Op. Res.*, vol. 68, no. 2, pp. 411–434, 2020.

[17] S. Scherrer, A. Perrig, and S. Schmid, "The value of information in selfish routing," in *Proc. SIROCCO*, 2020, p. 366–384.

[18] Y. Zeng, Q.-C. He, and X. Cai, "Targeted Bayesian persuasion in a basic selfish routing game," in *Proc. INFORMS-CSS*. Springer, 2022, pp. 47–56.

[19] P. N. Brown, "Providing slowdown information to improve selfish routing," in *Proc. GameNets*. Springer, 2023, pp. 328–338.

[20] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, 2016, pp. 1451–1455.

[21] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4411–4422, 2018.

[22] C. H. Bell and S. Stidham, "Individual versus social optimization in the allocation of customers to alternative servers," *Manag. Sc.*, vol. 29, p. 831–839, 1983.

[23] H. Kameda, J. Li, C. Kim, and Y. Zhang, *Overall Optimal Load Balancing vs. Individually Optimal Load Balancing*. London: Springer London, 1997, pp. 35–97.

[24] M. Haviv and T. Roughgarden, "The price of anarchy in an exponential multi-server," *Oper. Res. Lett.*, vol. 35, no. 4, pp. 421–426, 2007.

[25] E. Altman, U. Ayesta, and B. Prabhu, "Load balancing in processor sharing systems," *Telecommun. Syst.*, vol. 47, pp. 35–48, 2011.

[26] 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.

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

[28] T. Roughgarden and E. Tardos, "How bad is selfish routing?" *J. ACM*, vol. 49, no. 2, p. 236–259, mar 2002.

[29] C. H. Papadimitriou, "Algorithms, games, and the Internet," in *Automata, Languages and Programming*, F. Orejas, P. G. Spirakis, and J. van Leeuwen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[30] P. Castagno, V. Mancuso, M. Sereno, and M. Ajmone Marsan, "A simple model of MTC flows applied to smart factories," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2906–2923, 2020.

[31] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 1702–1716, 2019.

[32] D. Callegaro and M. Levorato, "Optimal edge computing for infrastructure-assisted UAV systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1782–1792, 2021.

[33] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, 2018.