

A game-theoretic approach to computation offloading in mobile cloud computing

Valeria Cardellini · Vittoria De Nitto Personé ·

Valerio Di Valerio · Francisco Facchini ·

Vincenzo Grassi · Francesco Lo Presti ·

Veronica Piccialli

Received: date / Accepted: date

Abstract We consider a three-tier architecture for mobile and pervasive computing scenarios, consisting of a *local* tier of mobile nodes, a *middle* tier (*cloudlets*) of nearby computing nodes, typically located at the mobile nodes access points but characterized by a limited amount of resources, and a *remote* tier of distant cloud servers, which have practically infinite resources. This architecture has been proposed to get the benefits of computation offloading from mobile nodes to external servers while limiting the use of distant servers whose higher latency could negatively impact the user experience. For this architecture, we consider a usage scenario where no central authority exists and multiple non-cooperative mobile users share the limited computing resources of a close-by cloudlet and can selfishly decide to send their computations to any of the three tiers. We define a model to capture the users interaction and to investigate the effects of computation offloading on the users' perceived performance. We formulate the problem as a generalized Nash equilibrium problem and show existence of an equilibrium. We present a distributed algorithm for the computation of an equilibrium which is tailored to the problem structure and is based on an in-depth analysis of the underlying equilibrium problem. Through numerical examples, we illustrate its behavior and the characteristics of the achieved equilibria.

Keywords Mobile cloud computing · Generalized Nash equilibrium problem · Distributed algorithm

Mathematics Subject Classification (2000) 90C33 · 90C30 · 68M20

V. Cardellini, V. De Nitto Personé, V. Di Valerio, V. Grassi, F. Lo Presti and V. Piccialli
Department of Civil Engineering and Computer Science Engineering, University of Roma "Tor Vergata",
Italy. E-mail: {cardellini, denitto, di.valerio, vgrassi, lopresti, piccialli}@disp.uniroma2.it

F. Facchini
Department of Computer, Control, and Management Engineering, University of Rome La Sapienza, Via
Ariosto 25, 00185 Roma, Italy. E-mail: facchini@dis.uniroma1.it

1 Introduction

Mobile devices (e.g. smartphones and tablets) are more and more becoming the hub around which much of the computing and communication demand of users is centered, thus posing new, heavy challenges. Indeed, in spite of the continuous technological improvements, the computation capabilities of mobile devices still are (and likely will remain) limited with respect to their “fixed” counterparts (e.g. desktop computers and data center servers): thus execution time of an application on a mobile node is (possibly much) greater than on a fixed one. In addition, mobile nodes are battery powered; hence energy consumption is a key issue to be accounted for. To overcome these potential limitations, it has been suggested to offload code execution from the mobile node to *external* machines [30]. This strategy has many potential advantages: (i) reduced application execution time; (ii) reduced battery consumption; and (iii) the possibility to execute applications whose resource demand could exceed the capabilities of mobile nodes.

There are several proposals in the literature (see [2, 33] for a comprehensive survey) which rely on cloud computing infrastructures for computation offloading in mobile scenarios [4]. Cloud computing delivers the vision of computing as a utility (such as water, electricity, gas, and telephony) and provides “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released” [25]. However, the use of traditional cloud infrastructures in a mobile environment can introduce significant network delays which, as reported in the literature, adversely affect the user experience and outweigh the potential benefits of this solution [8, 9, 16]. To overcome this problem, it has been proposed to use close-by servers (referred to as *cloudlets*), typically located at the wireless access points (APs) where mobile nodes connect to, so that they are at just “one hop” distance from the mobile node [31].

AP-located cloudlets cannot reasonably be expected to provide the “unlimited” amount of resources typically provided by a distant cloud server. Indeed, economic reasons (and physical constraints) limit the amount of resources that can be allocated to each cloudlet [31]. Hence, while a cloud server guarantees a good isolation among different users that offload their computations to it (i.e. users do not compete for the cloud resources), this does not hold in a cloudlet. As the load increases, resources contention and sharing can cause delays and performance degradation which might result in higher and higher response times, which in turn can offset the benefits of offloading computation to the cloudlet.

As a consequence, while the analysis of whether and to what extent it is convenient to offload computation to a distant cloud server can be carried out in a “single user” scenario, thanks to the isolation among concurrent users, the same analysis in a cloudlet-based architecture requires to take into consideration the dynamics of the interaction among the different users and the possible presence of regulatory policies for the access to the shared cloudlet resources.

Most of the papers that investigate the effectiveness of computation offloading consider single user mobile computing scenarios (e.g., [1, 7, 9, 15, 21, 23, 36]), thus implicitly assuming a perfect isolation in case of concurrent users. We are aware

of only few papers where interactions among different mobile users on a resource-limited cloud are taken into account [28, 29, 37, 38], see next session for a more detailed discussion.

In this paper, we will consider the general case of an architecture intended to support computation offloading for mobile nodes, where both a *middle* tier consisting of nearby resource-limited cloudlets and a *remote* tier consisting of resourceful distant cloud servers are available, as depicted in Figure 1 (this architecture is referred to as a *hybrid* mobile cloud architecture in [2]).

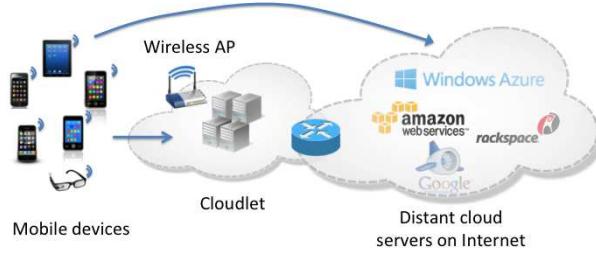


Fig. 1 Three-tier architecture for mobile cloud computing.

For such an architecture both a *managed* and an *unmanaged* usage scenario can be envisioned. The former scenario typically corresponds to the case where a wireless service provider (WSP) deploys its own cloudlet infrastructure at its own APs, to be used by its mobile subscribers. Hence, the WSP can be expected to centrally regulate the access to the cloudlet, with the goal of offering a good service experience to its subscribers and of fulfilling its own utility goals. The latter scenario corresponds instead to the case where cloudlet-augmented WiFi hot spots are deployed by public authorities or private entrepreneurs at facilities like airports, train stations, public buildings, cafes, etc., for the benefit of their citizens or customers. This (future) scenario is an extension of the current one where free-access WiFi hot spots are deployed just as an additional service for citizens, or as a way for attracting more customers, on a simple best effort basis and without any attempt of regulating their use. Analogously to this current scenario, in the unmanaged scenario we envision cloudlet-augmented WiFi hot spots are offered on a best effort basis, as their management is not likely to be part of the core business of the authorities that deployed them, and mobile users autonomously decide whether or not to take advantage of their presence, according to their own goals.

The managed scenario gives rise to a hard optimization problem for the handling of the cloudlet resources for which only centralized heuristics have been proposed, see [28, 29, 37, 38]. On the other hand, the unmanaged scenario is more challenging and has not been addressed in the literature. In this paper we focus on the latter unmanaged scenario, with the goal of investigating whether and under what conditions it gives rise to a convenient offloading strategy. To this end, we analyze the interaction among mobile users in a game-theoretic setting, assuming that the users indepen-

dently determine their offloading strategies according to a rational behavior. Within this framework, the contributions of this work are as follows:

- to the best of our knowledge, this is the first work where computation offloading is analyzed for a general multi-user “three-tier” mobile cloud computing scenario, with no central authority managing the access to the two external cloud tiers;
- by using queueing theory, we model such a scenario as a non-cooperative game among selfish users, where the users interaction can be formulated as a Generalized Nash Equilibrium Problem (GNEP) [11]. The queueing model underlying the GNEP formulation allows us to capture in a reasonably simple way the effects of computation offloading on the users’ perceived performance;
- we introduce a *distributed* algorithm for the computation of an equilibrium which is tailored to the model architecture. This algorithm, on the one hand is based on an in-depth analysis of the underlying equilibrium problem and, on the other hand, exploits and adapts some very recent game-theoretic advancements. The overall result is a model where each user can determine his/her own computation offloading strategy automatically on the basis of easily collected information;
- we report computational experiments demonstrating the effectiveness of the proposed algorithm and illustrating the characteristics of the achieved solution.

The remainder of the paper is organized as follows. Section 2 presents related works and motivates the mobile computing scenario we consider. In Section 3, we describe this scenario and state the problem we intend to tackle, while in Section 4 we propose the game-theoretic model that is used in the rest of the paper. In Section 5 we analyze the properties of the game, and show the existence of an equilibrium. In Section 6 we provide a distributed algorithm for the achievement of an equilibrium, and discuss issues related to its implementation. Section 7 presents a set of experiments illustrating the behavior of the solution method and assessing the characteristics of the achieved equilibria. Finally, Section 8 outlines future work.

2 Related Work

Several architectural proposals aimed at supporting the implementation of computation offloading (or “cyber foraging” [30]) in mobile cloud computing (MCC) scenarios have appeared in the recent past. Some comprehensive surveys have been recently published [2, 14, 33, 34], but other papers have appeared and continue to appear on this subject [1, 15, 21, 27, 29, 38]. The proposed architectures for MCC mainly differ in: (1) the granularity of workload offloaded to external (cloud) nodes, spanning for example entire virtual machines, application components or single application functions; (2) the methodologies adopted to determine what parts of the application can be potentially offloaded, including manual or automatic partitioning methodologies; (3) and whether application partitioning is determined statically before the application starts its execution, or dynamically at runtime, with the possibility of changing the partitioning during the application execution.

The exploitation of external nodes “close” to mobile devices has been suggested to alleviate the latency problem caused by the interaction with distant cloud servers

located in the Internet. Close nodes could be peer mobile nodes [18, 22, 35] or the wireless access points (APs) where mobile devices connect to, suitably augmented with some computational and storage capacity [16, 31]. A prototype implementation of this kind of augmented AP has been recently announced by IBM and Nokia Siemens Networks¹.

Mostly related to the work presented in this paper are the methodologies aimed at determining which offloadable tasks of a mobile application should be actually shifted from the mobile device to external nodes with the goal of improving the application performance and the user experience. We can broadly classify the existing proposals according to a *single user* vs. a *multiple users* scenario. In the single user scenario, a single mobile node is considered, without taking into account possible interference with other mobile nodes. On the other hand, in the multiple users scenario the offloading decisions take into account that multiple users compete for computational external resources that may be scarce.

Most of the offloading methodologies proposed up to now (e.g., [1, 7, 9, 15, 36]) focus on the single user scenario and address the issue by representing the application as a weighted graph/tree and applying a graph partitioning algorithm, whose complexity depends on the granularity of the offloading. The optimal solution is determined through an Integer Linear Programming (ILP) formulation while fulfilling some given objectives (e.g., application delay, energy saving, communication cost). However, since graph partitioning is an NP-complete problem, heuristics have been proposed to find efficiently approximate solutions so as to be able to deal also with large graphs. Solutions based on graph partitioning have been also investigated in pre-cloud mobile scenarios, e.g. [26].

To the best of our knowledge, only few works have addressed the multiple users scenario [28, 29, 37, 38]. Yang et al. [37] propose a cooperative centralized optimization problem with the goal of partitioning the computations from multiple users among their mobile devices and a given set of limited cloud resources, to minimize the average application delay for all the users. The proposed problem is formulated using Mixed Integer Linear Programming (MILP). In [38], the same authors study the partitioning problem for mobile data stream applications and consider multiple users that share the wireless network bandwidth as well as computational cloud resources with the goal of maximizing the throughput of the data stream application. The problem is addressed by means of a genetic algorithm. Differently from us, they consider only a two-tier architecture, composed of mobile devices and distant cloud servers, and their algorithm runs on the cloud side.

Similarly to our envisaged scenario, Rahimi et al. [28, 29] consider a three-tier architecture for MCC with multiple users, where local cloud resource are limited; in addition, in [29] they also take into consideration user mobility information. They formulate the tiered cloud resource allocation as an optimization problem and propose for its solution a greedy heuristic based on a simulated annealing approach. Their heuristic runs on a centralized entity (i.e. a middleware broker) that has to be contacted by the mobile nodes.

¹ www-03.ibm.com/press/us/en/pressrelease/40490.wss

Differently from all the works in the multiple users scenario described above, we consider a scenario where, as motivated in the introduction, no central authority exists that manages the access to the shared resources. In this scenario, multiple users decide selfishly whether and where to offload their computations, and we analyze their non-cooperative behavior in a game theoretic setting. Furthermore, we propose a distributed algorithm that can be executed on each mobile node and is appealing for a real-world implementation.

Game theory provides a suite of conceptual and procedural tools to capture the interaction among selfish players and has been successfully applied to typical ICT problems, like resource allocation, routing, pricing, and load balancing. In particular, it has been largely applied in mobile and wireless scenarios in the context of competition in the access to a shared communication medium [24].

3 System Model

We consider a mobile computing scenario as depicted in Figure 1, where a set of mobile nodes share a wireless access point (AP) to connect to the Internet. Different applications are executed on the mobile nodes, each consisting of one or more tasks. To overcome their limited computational capabilities and save battery power, mobile nodes can offload (part of) their computational load to an infrastructure consisting of: (i) a cloudlet, i.e. a node with computational capabilities deployed at the AP (and hence at “one hop” distance from the mobile nodes); (ii) a conventional remote cloud accessible over the Internet.

In the following, without lack of generality, we will refer to a *task* as the unit of computation. At the coarsest level, a task can correspond to an entire application, while at the finest level it can correspond to a function, e.g., an image compression, or even a simpler operation. It is worth observing that, in general, not all application components can be offloaded, as some component always needs to be executed locally, e.g., a task associated to an application user interface. The nearby cloudlet and the distant cloud are characterized by the same execution environment. In other words, it is functionally equivalent to offload a task to the cloudlet or to the cloud. Apart from this, a cloudlet differs substantially from a conventional cloud in that it is characterized by a limited amount of resources, while the conventional cloud is assumed to have a seemingly unlimited capacity.

As motivated in the introduction, we focus on an “unmanaged” scenario, where users autonomously decide whether or not to take advantage of nearby cloudlets, rather than some remote cloud. Due to its limited resources, we assume that a cloudlet offers a *best effort* service model whereby quality of service guarantees, admission control strategies, and resource allocation policies are not supported. At low load the mobile users can take full advantage of the computational capability offered by the cloudlet; as the load increases, resources contention and sharing can cause delays and performance degradation, thus making more appealing the execution on a distant server. Users exploit the available resources in a “selfish” way, according to their own utility goals. Each time an offloadable task is to be executed, a decision must be taken on whether it is more convenient for the user to execute it locally, on the

one-hop cloudlet, or on the more resourceful and distant cloud server. The former alternative causes the consumption of mobile node local resources only (including battery power). The latter two alternatives allow to save local resources and also to have the task executed on possibly faster devices. These potential advantages must however be weighted against possible interaction delays, and also power consumption due to the additional communications. If the mobile node decides to offload the task, the code and/or data are transferred for remote execution. Upon completion, a message with the computation results is returned to the mobile device.

As shown in Figure 2, we model our system as a queueing network. Queueing theory is widely used in the analysis of resource contention in computing and communication systems [20], and seems to be a natural candidate to capture the main features of our system.

The mobile device and the cloudlet are represented as queueing nodes to capture the resources contention on these two systems. The cloudlet is modeled as a set of n servers since we expect a cloudlet to be a “data center in a box” and therefore to comprise possibly multiple servers machines, with multiple processors/cores [31]. The cloud, on the other hand, given its virtually infinite capacity can be regarded as an infinite server, with no contention among different users. Finally, we model both the wireless access network and the Internet as simple delay centers to capture the average network delay experienced by the user when a task is remotely executed. In conclusion, the mobile device is modeled as an $M/G/1/PS$ queue; the cloudlet is modeled as a set of $n M/G/1/PS$ homogeneous servers with a front-end dispatcher that uniformly splits the arrival stream among the servers (this latter architecture has been proved to be an effective and popular solution for load sharing in multiserver systems and is largely used by commercial products, e.g. see [3]); the wireless access network, the Internet, and the remote cloud are modeled as an $M/G/\infty$ queue, i.e. as delay centers.

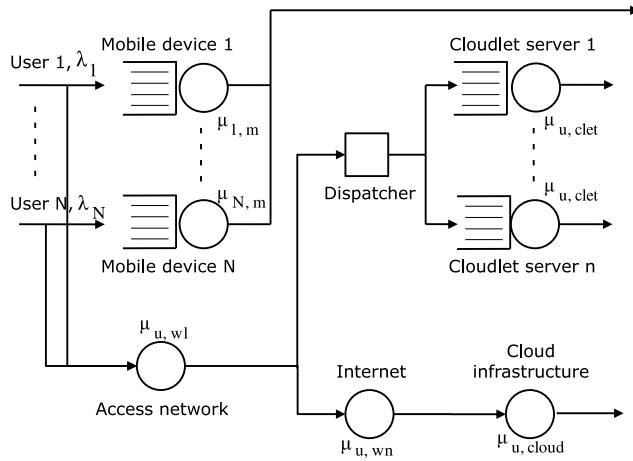


Fig. 2 System model

For each user, we assume that the (offloadable) tasks to be executed by a mobile device are generated according to a Poisson process with rate λ_u , $u = 1, \dots, N$. We denote by $\frac{1}{\mu_{u,m}}$, $\frac{1}{\mu_{u,clet}}$ and $\frac{1}{\mu_{u,cloud}}$ the expected execution time of user u tasks on the mobile device, the cloudlet and the cloud, respectively. We denote by $\frac{1}{\mu_{u,wl}}$ and $\frac{1}{\mu_{u,wn}}$ the expected time to transfer data/code for remote execution over the wireless access network and the Internet, respectively. We assume the latter two quantities include the time for the return message to be delivered to the mobile node (in other words, they represent the round trip times). Finally, we denote by $P_{u,m}$ and $P_{u,t}$ the power consumed by the mobile device to execute locally the task and to transmit the task code/data for remote execution, respectively.

We remark that our application model is somewhat different from those considered in the literature discussed in the previous section, as we do not consider possible dependencies among tasks belonging to the same application. The papers [1, 7, 9, 15, 36] model these dependencies as a graph and analyze how to partition the application tasks on the mobile and cloud resources. Our higher level model allows us to capture the tasks contention on the shared resources, which is our focus, and to overcome at the same time the difficulties caused by the combinatorial aspects in the above mentioned papers.

4 Generalized Nash Equilibrium Formulation

In this section we formulate the mobile computation offloading problem as a Generalized Nash Equilibrium Problem [11, 13]. The goal of each user (actually of the mobile node) is to determine whether and where to offload a task based on the impact this has on his/her usage experience, expressed through suitable Quality of Service (QoS) measures. We call this decision the user *strategy* and model it by associating to each user u a triple $x_u = \{x_{u,m}, x_{u,clet}, x_{u,cloud}\}$, $\sum_{i \in I} x_{u,i} = 1$, where $I = \{m, clet, cloud\}$, which represents the fraction of user tasks that is executed locally ($x_{u,m}$), offloaded to the cloudlet ($x_{u,clet}$), or to the cloud ($x_{u,cloud}$).

Given that power consumption and application performance are the most important quality factors in a mobile scenario, see e.g. [9, 1, 15, 7], we consider them as the QoS measures driving each user strategy. In particular, we assume that the user wants to optimize the observed performance while limiting the power consumption. Without lack of generality, we consider as the user performance measure the expected number of user tasks in the system, i.e. the expected number of tasks launched but not yet completed. From the mobile user point of view, this corresponds to the average execution time of the number of tasks launched in a time unit. We observe that this is a quite general approach, which accounts for different levels of detail/granularity. As an example, consider the case of an application which is executed once per second and whose execution requires ten modules to be run. We can consider as a task either the application or the invoked modules and with the proposed performance measure we obtain exactly the same expression.

Let us denote by $R_{u,m}$, $R_{u,clet}$ and $R_{u,cloud}$ the mean response time when a task is executed locally, offloaded to the cloudlet or to the cloud, respectively. It is easy to verify that:

$$R_{u,m} = \frac{1}{\mu_{u,m}} - \frac{x_{u,m}\lambda_u}{1 - \frac{x_{u,m}\lambda_u}{\mu_{u,m}}}, \quad R_{u,clet} = \frac{1}{\mu_{u,wl}} + \frac{1}{1 - \frac{1}{n} \sum_v \frac{x_{v,clet}\lambda_v}{\mu_{v,clet}}} \quad (1)$$

$$R_{u,cloud} = \frac{1}{\mu_{u,wl}} + \frac{1}{\mu_{u,wn}} + \frac{1}{\mu_{u,cloud}}. \quad (2)$$

The user objective is thus to minimize $\lambda_u R_u(x_u, x_{-u})$ (that by Little's first law represents the number of tasks in the system) within a given energy budget. Here $R_u(x_u, x_{-u})$ denotes user u mean task response time and x_{-u} denotes the strategies of all users except user u . User u mean task response time is defined as follows:

$$R_u(x_u, x_{-u}) = x_{u,m}R_{u,m} + x_{u,clet}R_{u,clet} + x_{u,cloud}R_{u,cloud}. \quad (3)$$

Note that the user mean response time depends not only on the user u strategy x_u , but also on the strategies of the other users. This dependency is due to the users indirectly affecting each other when they offload tasks to the cloudlet, since the cloudlet mean response time is function of the cloudlet load $\sum_v \frac{x_{v,clet}\lambda_v}{\mu_{v,clet}}$ to which each user contributes.

Each user u , in order to compute the optimal strategy, needs to solve the following optimization problem:

$$\min \lambda_u R_u(x_u, x_{-u}) \quad (4)$$

$$\text{subject to: } \frac{1}{n} \sum_v \frac{x_{v,clet}\lambda_v}{\mu_{v,clet}} \leq U_{max} \quad (5)$$

$$\frac{x_{u,m}\lambda_u}{\mu_{u,m}} P_{u,m} + \frac{(x_{u,clet} + x_{u,cloud})\lambda_u}{\mu_{u,wl}} P_{u,t} \leq P_{u,max} \quad (6)$$

$$x_{u,clet} + x_{u,cloud} \leq \chi \quad (7)$$

$$\sum_{i \in I} x_{u,i} = 1 \quad (8)$$

$$x_{u,m}, x_{u,clet}, x_{u,cloud} \geq 0. \quad (9)$$

Constraint (5) models the cloudlet utilization which we assume should not exceed a given threshold U_{max} (in practice, this corresponds to giving an upper bound to the cloudlet response time). Observe that this constraint involves the decision variables of all the users. Constraint (6) ensures that the user energy consumption is lower than a threshold $P_{u,max}$. Constraint (7) takes into account that, in general, only a fraction χ , $0 < \chi \leq 1$, of the tasks can be offloaded. Finally, the simple and natural constraints (8) and (9) ensure that the considered fractions are greater than or equal to zero and that their sum is one.

In this setting, the users decisions are mutually dependent and the proposed model is a GNEP. GNEPs differ from classical Nash Equilibrium Problems (NEP) in that,

while in a NEP only the players' objective functions depend on the other players strategies, in a GNEP both the objective functions and the strategy sets depend on the other players strategies. In our problem, the dependence of each player strategy set on the other players strategies is represented by the constraint (5), which includes all the users decision variables $x_{u,clet}$. More specifically, since the players all share a common (linear) constraint, this game is known as jointly convex game [10].

5 Properties of the GNEP Formulation

In this section we show that the game (4)-(9) can actually be solved by finding a solution to a suitable *Variational Inequality* (to be defined later on), for which we can then derive a distributed algorithm. First, in Section 5.1, crucial to our approach, we establish that the function associated to the *Variational Inequality* is, under appropriate, reasonable conditions, *monotone*. Then, in Section 5.2 we transform the original GNEP in an equivalent *extended* game, the equilibrium point of which can be computed in a distributed way [32], as detailed in Section 6.

5.1 Existence and Monotonicity Properties of the GNEP

We recall that each user $u = 1, \dots, N$ controls three variables: $x_u = (x_{u,m}, x_{u,clet}, x_{u,cloud})$. For sake of simplicity, we set:

$$\alpha_u = \frac{\lambda_u}{\mu_{u,m}}, \quad \beta_u = \frac{\lambda_u}{\mu_{u,wl}}, \quad \delta_u = \frac{\lambda_u}{\mu_{u,clet}}, \quad \gamma_u = \lambda_u \left(\frac{1}{\mu_{u,wl}} + \frac{1}{\mu_{u,wn}} + \frac{1}{\mu_{u,cloud}} \right).$$

Using this notation we can rewrite problem (4)-(9) as

$$\min \quad \lambda_u R_u(x_u, x_{-u}) \tag{10}$$

$$\text{subject to} \quad \frac{1}{n} \sum_v \delta_v x_{v,clet} \leq U_{\max} \tag{11}$$

$$\alpha_u P_{u,m} x_{u,m} + \beta_u P_{u,t} (x_{u,clet} + x_{u,cloud}) \leq P_{u,\max} \tag{12}$$

$$x_{u,clet} + x_{u,cloud} \leq \chi \tag{13}$$

$$x_{u,m} + x_{u,clet} + x_{u,cloud} = 1 \tag{14}$$

$$x_{u,m}, x_{u,clet}, x_{u,cloud} \geq 0, \tag{15}$$

where

$$\lambda_u R_u(x_u, x_{-u}) = \frac{\alpha_u x_{u,m}}{1 - \alpha_u x_{u,m}} + \beta_u x_{u,clet} + \gamma_u x_{u,cloud} + \frac{\delta_u x_{u,clet}}{1 - \frac{1}{n} \sum_v \delta_v x_{v,clet}}.$$

In order to analyze the game we make the following basic assumption:

Assumption A U_{\max} as well as all α_u and δ_u , $u = 1, \dots, N$, are positive and smaller than 1.

Under this standard assumption, it is easy to check that each user's problem is convex for given values of the other users' variables. By the results in [10], we know

that we can recover a solution of this jointly convex game (known as *variational solution* or *normalized solution*) by solving a suitable Variational Inequality: VI (K, F) ² [12]. In order to define this VI which permits to compute a solution of our GNEP we therefore have to specify the set K and the function F . We do this next, following [10]. To define K we first define the sets

$$\tilde{K}_u := \{x_u \in \mathbb{R}_+^3 : \sum_{i \in I} x_{u,i} = 1, x_{u,clet} + x_{u,cloud} \leq \chi, \\ \alpha_u P_{u,m} x_{u,m} + \beta_u P_{u,t} (x_{u,clet} + x_{u,cloud}) \leq P_{u,\max}\},$$

which are nothing else but the feasible set of user u with the joint constraint neglected. The “contribution” of the joint constraint is taken into account by the set

$$\Omega := \{x \in \mathbb{R}^{3N} : \frac{1}{n} \sum_u \delta_u x_{u,clet} \leq U_{\max}\}.$$

The set K in the definition of our VI is then given by $K := (\prod_{u=1}^N \tilde{K}_u) \cap \Omega$. It remains now to define the function F . This is just the vector obtained by “stacking” the partial gradients of each user, where the gradients are taken only with respect to the users’ own variables:

$$\nabla_{x_u} \lambda_u R_u = \begin{pmatrix} \frac{\alpha_u}{(1 - \alpha_u x_{u,m})^2} \\ \beta_u + \delta_u \frac{1 - \frac{1}{n} \sum_{v \neq u} \delta_v x_{v,clet}}{(1 - \frac{1}{n} \sum_v \delta_v x_{v,clet})^2} \\ \gamma_u \end{pmatrix}, \quad F = \begin{pmatrix} \frac{\alpha_1}{(1 - \alpha_1 x_{1,m})^2} \\ \beta_1 + \delta_1 \frac{1 - \frac{1}{n} \sum_{v \neq 1} \delta_v x_{v,clet}}{(1 - \frac{1}{n} \sum_v \delta_v x_{v,clet})^2} \\ \gamma_1 \\ \vdots \\ \frac{\alpha_N}{(1 - \alpha_N x_{N,m})^2} \\ \beta_N + \delta_N \frac{1 - \frac{1}{n} \sum_{v \neq N} \delta_v x_{v,clet}}{(1 - \frac{1}{n} \sum_v \delta_v x_{v,clet})^2} \\ \gamma_N \end{pmatrix}.$$

Existence of a solution to a general GNEP is usually not easy to show. However, in our case we are dealing with a jointly convex GNEP with compact feasible set and it is well-known [10], but can also easily be seen directly, that a solution to the GNEP (10)-(15) exists.

Proposition 1 *Supposing that Assumption A holds, the GNEP (10)-(15) has at least one solution.*

Proof. As already observed, under Assumption A any solution of the VI (K, F) is a solution of the GNEP (10)-(15), see [10]. But F is continuous on K and K is obviously compact. Therefore by [12, Corollary 2.2.5] VI (K, F) has a solution and, as a consequence, also the original GNEP (10)-(15) has a solution. \square

² The VI (K, F) , where $K \subseteq \mathbb{R}^n$ is a closed convex set and $F : K \rightarrow \mathbb{R}^n$ is a continuous function, is the problem of finding a point $\bar{x} \in K$, such that $F(\bar{x})^T (x - \bar{x}) \geq 0$, for all $x \in K$.

Note that in general the GNEP (10)-(15) could have infinite solutions; our aim is to compute a variational solution by a distributed algorithm (see comments later on the significance of this particular solution). To this end a key role is played by the *monotonicity* of F ³. The easiest way to check the monotonicity of a differentiable F is to check that the Jacobian of F , JF , is positive semidefinite on K [12].

The Jacobian of F has the following structure:

$$JF(x) = \begin{pmatrix} A_1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & B_1 & 0 & 0 & B_{12} & 0 & \dots & 0 & B_{1N} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \vdots & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & A_N & 0 & 0 \\ 0 & B_{N1} & 0 & 0 & B_{N2} & 0 & \dots & 0 & B_N & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix} \quad (16)$$

where

$$\begin{aligned} A_u &= \frac{2\alpha_u^2}{(1 - \alpha_{ux_{u,m}})^3}, \quad B_u = \frac{2}{n}\delta_u^2 \frac{1 - \frac{1}{n}\sum_{v \neq u} \delta_v x_{v,clet}}{(1 - \frac{1}{n}\sum_t \delta_t x_{t,clet})^3}, \\ B_{uv} &= \frac{1}{n}\delta_v \delta_u \frac{1 - \frac{1}{n}\sum_v \delta_v x_{v,clet} + \frac{2}{n}\delta_u x_{u,clet}}{(1 - \frac{1}{n}\sum_t \delta_t x_{t,clet})^3}. \end{aligned}$$

Theorem 1 Assume that

$$\delta_{\max} \leq \frac{2n}{N\chi}(1 - U_{\max}), \quad (17)$$

where $\delta_{\max} = \max_{u=1,\dots,N} \delta_u$, then F is monotone.

Proof. Reordering the variables, $JF(x)$ can be rewritten in the following form:

$$JF(x) = \begin{pmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad A = \text{diag}(A_u)_{u=1}^N, \quad B = \begin{pmatrix} B_1 & B_{12} & \dots & B_{1N} \\ B_{21} & B_2 & \dots & B_{2N} \\ \vdots & & & \vdots \\ B_{N1} & B_{N2} & \dots & B_N \end{pmatrix}.$$

Since A is positive definite by Assumption A, checking the monotonicity reduces to checking that the matrix B is positive semidefinite. In order to check the semidefiniteness of B we check the semidefiniteness of its symmetric part $B^s := \frac{1}{2}(B^T + B)$. Set

$$D := 1 - \frac{1}{n} \sum_t \delta_t x_{t,clet},$$

the diagonal elements B_u^s can be rewritten as

$$B_u^s = \frac{\delta_u^2}{nD^2} \left(2 + 2 \frac{\delta_u x_{u,clet}}{nD} \right)$$

³ We recall that F is monotone on K if $(F(y) - F(x))^T (y - x) \geq 0, \forall y, x \in K$.

while the off-diagonal elements are

$$B_{uv}^s = B_{vu}^s = \frac{\delta_v \delta_u}{nD^2} \left(1 + \frac{\delta_v x_{v,clet} + \delta_u x_{u,clet}}{nD} \right).$$

Let δ denote the vector $\delta := (\delta_1 \dots \delta_N)^T$. It is easily seen that the matrix B^s can be rewritten as

$$\frac{1}{nD^2} \left(\delta \delta^T \circ \left(I + E + \frac{2}{nD} \begin{pmatrix} \delta_1 x_{1,clet} & \frac{\delta_1 x_{1,clet} + \delta_2 x_{2,clet}}{2} & \dots & \frac{\delta_1 x_{1,clet} + \delta_N x_{N,clet}}{2} \\ \ddots & & & \\ \frac{\delta_N x_{N,clet} + \delta_1 x_{1,clet}}{2} & \frac{\delta_N x_{N,clet} + \delta_2 x_{2,clet}}{2} & \dots & \delta_N x_{N,clet} \end{pmatrix} \right) \right) \quad (18)$$

where the symbol \circ denotes the Hadamard product of two matrices, i.e. the matrix having as elements $(A \circ B)_{ij} = A_{ij}B_{ij}$, and E is the matrix with all entries equal to 1.

Set $x_{clet}^\delta := (\delta_1 x_{1,clet} \dots \delta_N x_{N,clet})^T$, and let $e \in \mathbb{R}^N$ be the vector of all ones, then, noting that

$$\begin{pmatrix} \delta_1 x_{1,clet} & \frac{\delta_1 x_{1,clet} + \delta_2 x_{2,clet}}{2} & \dots & \frac{\delta_1 x_{1,clet} + \delta_N x_{N,clet}}{2} \\ \ddots & & & \\ \frac{\delta_N x_{N,clet} + \delta_1 x_{1,clet}}{2} & \frac{\delta_N x_{N,clet} + \delta_2 x_{2,clet}}{2} & \dots & \delta_N x_{N,clet} \end{pmatrix} = \frac{1}{2} \left(x_{clet}^\delta e^T + e(x_{clet}^\delta)^T \right),$$

the matrix B^s is given by

$$B^s = \frac{1}{nD^2} \left(\delta \delta^T \circ \left(I + E + \frac{1}{nD} (x_{clet}^\delta e^T + e(x_{clet}^\delta)^T) \right) \right). \quad (19)$$

The Schur product theorem (see [17, Theorem 7.5.3]) states that the Hadamard product of two positive semidefinite matrices is positive semidefinite. Therefore, since the matrix $\delta \delta^T$ is obviously positive semidefinite, in order to show the positive semidefiniteness of B^s it is enough to show that the matrix

$$\left(I + E + \frac{1}{nD} (x_{clet}^\delta e^T + e(x_{clet}^\delta)^T) \right)$$

be positive semidefinite. Neglecting the contribution of the positive semidefinite matrix E , this reduces to proving that the minimum eigenvalue of the matrix $\frac{1}{nD} (x_{clet}^\delta e^T + e(x_{clet}^\delta)^T)$ is greater or equal to -1 . It is known, see [5, Fact 4.9.16], that the matrix $x_{clet}^\delta e^T + e(x_{clet}^\delta)^T$ has a characteristic polynomial given by

$$\eta^{N-2} \left(\eta^2 - 2(e^T x_{clet}^\delta) \eta + (e^T x_{clet}^\delta)^2 - N \|x_{clet}^\delta\|^2 \right). \quad (20)$$

From this we see that the matrix $x_{clet}^\delta e^T + e(x_{clet}^\delta)^T$ has $(N-2)$ zero eigenvalue, a non negative eigenvalue and a non positive eigenvalue. These two latter eigenvalues are given respectively by

$$\eta_+ = \frac{e^T x_{clet}^\delta + \sqrt{N} \|x_{clet}^\delta\|}{2}, \quad \eta_- = \frac{e^T x_{clet}^\delta - \sqrt{N} \|x_{clet}^\delta\|}{2}.$$

We then get that a sufficient condition for the positive semidefiniteness of B^s is

$$\frac{1}{nD} \left(\frac{\sqrt{N}\|x_{clet}^\delta\| - e^T x_{clet}^\delta}{2} \right) \leq 1. \quad (21)$$

But recalling that $e^T x_{clet}^\delta \geq \|x_{clet}^\delta\|$ since $x_{clet}^\delta \geq 0$, that on the feasible region $\|x_{clet}^\delta\|$ is at most $\sqrt{N}\delta_{\max}\chi$ (see (13)) and that $D \geq 1 - U_{\max}$ by (11), we see that

$$\frac{1}{nD} \left(\frac{\sqrt{N}\|x_{clet}^\delta\| - e^T x_{clet}^\delta}{2} \right) \leq \frac{1}{2nD} \left(\delta_{\max}\chi\sqrt{N}(\sqrt{N}-1) \right) \leq \frac{\delta_{\max}}{1-U_{\max}} \frac{N\chi}{2n}. \quad (22)$$

Therefore, (21) is certainly satisfied if (17) holds. \square

Remark 1 The previous theorem hinges on condition (17) which guarantees the key property of F being monotone. It is then important to get a good understanding of its meaning. However, before looking at this issue, we stress that condition (17) is just a *sufficient* condition for the monotonicity of F . Indeed, a look at the proof of Theorem 1 shows that condition (17) derives from a series of majorizations based on worst case scenarios; therefore, in practice we can expect monotonicity of F even when (17) is not “violated too much”. This is confirmed by the numerical results in Section 7, that show that condition (17) is not critical from the practical point of view. Condition (17) essentially says that monotonicity of F is guaranteed if the cloudlet is not overloaded. In fact, condition (17) requires that the maximum traffic intensity δ_{\max} of the users (on the cloudlets) be lower than a certain threshold value. For a given number N of users, this threshold increases when the number n of cloudlet servers increases or when either or both U_{\max} and χ decrease. Therefore monotonicity can always be achieved by deploying more cloudlets or by imposing in the protocol, i.e. in the constraints (11) and (13), suitably small values of U_{\max} and χ .

5.2 The Extended Game

Centralized algorithms for the computation of an equilibrium could now be easily derived by solving the VI (K, F) defined above. In fact, assuming monotonicity of F , there are plenty of *centralized* algorithms available, see [12]. However, in order to develop a *distributed* algorithm, we can not act directly on the original GNEP (10)-(15) or on its equivalent VI reformulation. Roughly speaking, the reason is that distributed algorithms require that the feasible sets (of the game or of the VI) are the Cartesian product of lower dimensional sets, a condition that in our case is not satisfied due to the shared constraint (5). However, as we show next, we are able to reformulate the GNEP (10)-(15) into another game with no coupling constraints through a simple, but non trivial transformation which, essentially, was first hinted at in [13]. It will turn out this new game inherits the monotonicity properties of the original game so that, as we will see in the next section, under the conditions of Theorem 1, we will be able to develop distributed algorithms for the computation of a variational solution of the GNEP (10)-(15).

To achieve the decoupling of the users' feasible sets, we consider an extended game, with one extra "player". In this extended game the first N users control x_u and have the problem

$$\min_{x_u \in \tilde{K}_u} \lambda_u R_u(x_u, x_{-u}) + \rho \left(\frac{\delta_u}{n} x_{u,clet} \right)$$

while the $(N+1)-th$ player controls the variable $\rho \in \mathbb{R}$ and solves the problem

$$\max_{\rho \geq 0} \rho \left(\frac{1}{n} \sum_u \delta_u x_{u,clet} - U_{\max} \right).$$

We call this game the *extended game*. Note that this extended game is a standard Nash equilibrium problem since there is no coupling in the constraints. The first N users are the "original" users. Their problems have been modified in two ways: (a) the joint constraint has been eliminated and (b) in the objective function a term has been added to make up for this omission. The $(N+1)$ -th user is a sort of cloudlet manager and controls the variable ρ which can be seen as the cloudlet "price". Note that the additional term in the objective function of the other users is then nothing else but the "cost" of using the cloudlet. More precisely, it can be shown that ρ will just turn out to be the Lagrange multiplier of the shared constraint (11). It is a classical result [12, Proposition 1.4.2] that our game is equivalent to the VI (F_e, K_e) , where

$$F_e(x, \rho) = \begin{pmatrix} 0 \\ \rho(\delta_1/n) \\ 0 \\ \vdots \\ 0 \\ \rho(\delta_N/n) \\ 0 \end{pmatrix}, \quad K_e = (\prod_{u=1}^N \tilde{K}_u) \times \mathbb{R}_+,$$

$$-\frac{1}{n} \sum_u \delta_u x_{u,clet} + U_{\max}$$

The following result is key to our developments and relates GNEP (10)-(15) to the extended game. Note that in the theorem below, when we say that the game is monotone, we obviously mean that its VI reformulation is so, in other words, that the function F_e is monotone.

Theorem 2 A point \bar{x} is a variational solution of the original game (10)-(15) if and only if a $\bar{\rho}$ exists such that $(\bar{x}, \bar{\rho})$ is a Nash equilibrium of the extended game. Furthermore, if the original game is monotone, then also the extended game is monotone.

Proof. The first assertion is just a verification which can be carried out comparing the Karush-Kuhn-Tucker conditions of the VI (K, F) and of the extended game. Note that since all constraints involved in both problems are affine, the Karush-Kuhn-Tucker conditions surely hold at a solution. The second assertion of the theorem can

be checked writing down the Jacobian of F_e :

$$JF_e(x, \rho) = \begin{pmatrix} & & & & \begin{matrix} 0 \\ \delta_1/n \\ 0 \\ \vdots \\ 0 \\ \delta_N/n \\ 0 \end{matrix} \\ & JF(x) & & & \\ & & & & \\ \hline & & & & \\ 0 - \delta_1/n & 0 & \cdots & 0 - \delta_N/n & 0 \end{pmatrix}.$$

This is a block skew-symmetric matrix, and $JF_e(x, \rho)$ is monotone if and only if $JF(x)$ is monotone. \square

The bottom line of this section is: we can compute a (variational) solution of the GNEP (10)-(15) by finding a solution of the standard extended game. This latter game is monotone if and only if the original GNEP is monotone and, in particular, if the conditions of Theorem 1 are met. On the basis of these results, in the next section we will show how to apply some very recent algorithmic developments in order to design distributed algorithms for the solution of the extended game.

6 Distributed Solution

In this section we consider the problem of computing an equilibrium of the GNEP (10)-(15) by a distributed algorithm. To achieve our goal we will combine in a suitable way classical results about proximal regularization, see e.g. [12, Chapter 12], with some very recent, advanced distributed methods proposed in [32]. In doing so, we take great care to make appropriate choices so that the resulting solution method is not only mathematically sound, but also well suited to the characteristics of our model, in terms of information exchange and computational burden of the users, so as to be amenable to practical use.

Our approach to the solution of (10)-(15) is to solve the VI (F_e, K_e) in a distributed way. To this end, one key requirement is that F_e be strongly monotone⁴. However, it can easily be observed that, because of the 0 in the lower-right corner of JF_e (see the proof of Theorem 2) F_e can never be strongly monotone, even if F is so. To circumvent this difficulty, we *regularize* the VI (F_e, K_e) and use a *proximal-point* method [12, Chapter 12]. This results in the following scheme, where α is an arbitrary positive constant.

⁴ We recall that F_e is strongly monotone on K_e if $(F_e(y) - F_e(x))^T(y - x) \geq m\|y - x\|^2$, $\forall y, x \in K_e$ for some fixed, positive m . Note that every strongly monotone function is monotone but the vice versa does not necessarily hold. If F_e is continuously differentiable it is known that F_e is strongly monotone on K_e if and only if $JF_e(x, \rho) - mI$ is positive semidefinite for all points in K_e .

Algorithm 1: Proximal-point algorithm for the solution of VI (F_e, K_e)

-
- (S.0) : Choose $(x^0, \rho^0) \in K_e$ and set $k = 0$.
 (S.1) : If (x^k, ρ^k) is a solution of VI (K_e, F_e) stop.
 (S.2) : Compute the new iteration (x^{k+1}, ρ^{k+1}) as the unique solution of the strongly monotone VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$.
 (S.3) : Set $k \leftarrow k + 1$ and go to Step 1.
-

It is known [12, Chapter 12] that the above scheme converges to a solution of the VI (K_e, F_e) , i.e. to a (variational) solution of the GNEP (10)-(15). The key point in developing a (totally asynchronous) distributed solution method is therefore the development of a (totally asynchronous) distributed solution method for the VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$. To this end we may consider the distributed Algorithm 2. Note that the algorithm we present is *synchronous*. We do so for simplicity of presentation only. *Totally asynchronous* (in the sense of [6]) versions can easily be envisaged and all the considerations we made in this section readily extend to the asynchronous case.

Algorithm 2: Parallel distributed algorithm for the solution of VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$

-
- (S.0) : Choose $(x^0, \rho^0) \in K_e$ and set $i = 0$.
 (S.1) : If (x^i, ρ^i) is a sufficiently accurate solution of VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$ stop.
 (S.2) : For $u = 1, \dots, N$ set x_u^{i+1} to be the unique solution of the strongly convex optimization problem

$$\min_{x_u} \lambda_u R_u(x_u, x_{-u}^i) + \rho^i \frac{\delta_u}{n} x_{u,clet} + \alpha \|x_u - x_u^k\|^2$$

subject to $x_u \in \tilde{K}_u$

- (S.3) : Take

$$\rho^{i+1} = \max\{0, \rho^k + \frac{1}{2\alpha} \left(\frac{1}{n} \sum_u \delta_u(x_{u,clet}^i) - U_{\max} \right)\}.$$

- Set $i \leftarrow i + 1$ and go to Step 1.
-

The overall scheme resulting by the combination of the *outer* Algorithm 1 and of the *inner* Algorithm 2 is depicted in Figure 3, where the information flows are also represented.

Below we discuss in more detail some important issues.

- The inner Algorithm 2 stops in Step 1 when a “sufficiently accurate solution of VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$ ” has been reached. In practice this is not a problem at all, as usually very few inner iterations are needed to reach a very accurate solution of VI $(K_e, F_e + \alpha(\cdot - (x^k, \rho^k)))$, since as the outer iterations progress and (x^k, ρ^k) converges, we are solving a sequence of outer problems that are more and

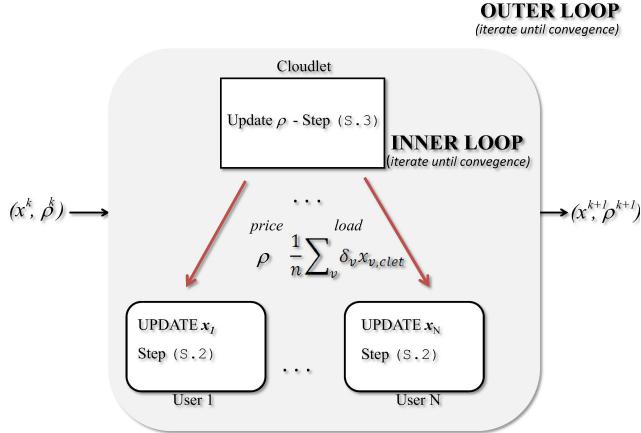


Fig. 3 Distributed algorithm

more similar. This is confirmed in our numerical experiments in Section 7. More details on this theoretical point can be found in [32].

- The problems solved by each user at Step 2 can be rather easily interpreted. The objective function includes two additional terms with respect to the original game. The first term, $\rho \left(\frac{\delta_u}{n} x_{u,cllet} \right)$, is a cost associated to the use of the cloudlet with a price of ρ . In other words we penalize the shared constraint (11) and “put it in the objective function” in order to decouple the feasible sets of the users. The second term, $\alpha \|x_u - x_u^k\|^2$, is a classical regularization term that is needed to guarantee strong convexity of the objective function.
- The problems solved by each user at Step 2 are three variables strongly convex problems with linear constraints and can be solved extremely efficiently and very fast by any commercial optimization software.
- The updating of the “price” ρ requires the cloudlet to monitor the system load (the term $\frac{1}{n} \sum_u \delta_u x_{u,cllet}^i$). The system load along with the price ρ are then sent by the cloudlet to the users which require this information to solve their optimization problem. We observe that the system load can be easily measured at the cloudlet side, and the cloudlet can be easily instrumented to transmit this information to the users exploiting its resources, so the distributed algorithm is amenable to a real-world implementation.
- We remark once more that our algorithm computes a variational solution of the GNEP (10)-(15), that is, one of the possibly infinite number of equilibria of the game. The variational solution is characterized by the fact that the multipliers of the shared constraint (11) are the same for all users (see [10]). This solution is particularly appealing from a practical point of view since it can be interpreted as a fairness condition for it implies that the “cost” of use of the cloudlet (the multiplier) is the same for all users.

7 Experimental Results

In this section we investigate through numerical experiments the behavior of the proposed computation offloading strategy. First, in Section 7.1, we compute the system equilibria under different scenarios and study how the users' tasks are dispatched among the mobile device, the cloudlet, and the remote cloud infrastructure. Then in Section 7.2 we compare the proposed non-cooperative strategy with a cooperative one. Our aim is to understand how the performance degrades due to the selfish behavior of the users.

For the analysis, we implemented in MATLAB the distributed algorithm in Section 6, setting the parameter α to 0.1. The algorithm stops when the norm of the difference of two consecutive iterations is less than 10^{-4} .

7.1 Non-cooperative Strategy Analysis

We consider a homogeneous scenario where the users profile is characterized by the same set of parameters. If not stated otherwise, as basic setting we consider $n = 2$ cloudlet servers, $\lambda_u = 0.25$ task/s, $1/\mu_{u,m} = 0.5$ s, $\mu_{u,cler} = 5\mu_{u,m}$, $\mu_{u,cloud} = 10\mu_{u,m}$, $1/\mu_{u,wl} = 0.1$ s, $1/\mu_{u,wn} = 0.4$ s and $U_{max} = 0.7$. The execution time parameters are consistent with those experimentally measured in [21, 7, 19]. We also set $\chi = 1$, i.e. all tasks can be offloaded to the cloud. Moreover, unless otherwise noted, we do not consider the power consumption constraint, i.e. we set $P_{u,max} = \infty$.

In Figure 4, we show the results of four set of experiments to investigate the behavior of the non-cooperative strategy as number of users, number of cloudlet servers, task execution time, and maximum power consumption increase. Note that, since we consider a homogeneous scenario, the user's strategies coincide. Hence, we only need to show the strategy of one user.

In the first set of experiments, we study the computation offloading strategy as the number of users (N) increases from 20 to 70. From Figure 4(a) we can observe that until the cloudlet is not overloaded, the users take fully advantage of its computational resources to execute their tasks ($x_{u,cler} = 1$). As the number of users grows, the cloudlet utilization increases. Eventually, when the utilization hits the threshold U_{max} , which occurs when $N = 56$, the cloudlet cannot serve all the tasks; as N increases further, a larger fraction of tasks is executed on the mobile nodes themselves. It is worth observing that, nevertheless, the tasks are not dispatched to the remote cloud due to the high delays which offset the faster computational speed. Figure 5(a) shows the number of user tasks in the system (i.e. the objective function value) for the first set of experiments. As we can expect, it increases with the number of users in the cloudlet, because the resource contention increases and a fraction of the tasks must be even executed on the slowest mobile devices.

In the second set of experiments, we study the behavior of the proposed strategy as the number of cloudlet servers increases from $n = 2$ to $n = 10$. We set the number of users to $N = 15$ and increase the task execution time, setting it to $1/\mu_{u,m} = 2.2$ s (so that the local execution on the mobile device is not suitable). The results are shown in Figure 4(b). As expected, increasing the computational power of the cloudlet allows

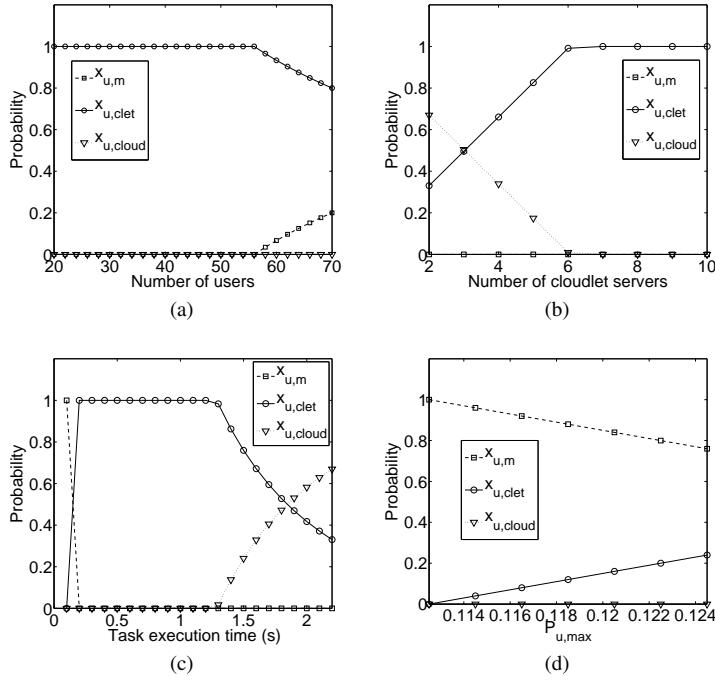


Fig. 4 User strategies for different system parameters.

for a larger fraction of tasks to be executed on it, which results, as shown in Figure 5(b), in a reduction of the number of user tasks in the system, also due to the faster network connection to the cloudlet.

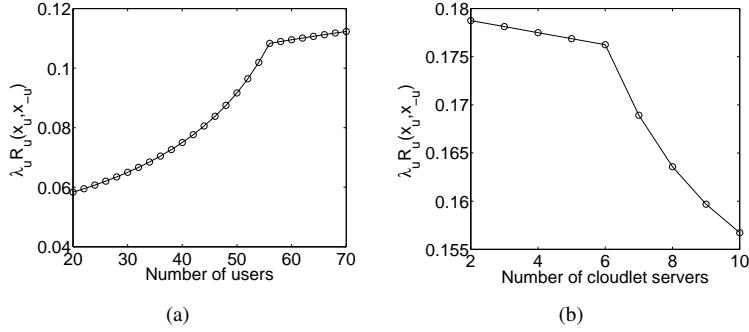


Fig. 5 Number of user tasks in the system

In the third set of experiments, we study the computation offloading strategy as the task execution time on the mobile node, $1/\mu_{u,m}$, ranges from 0.1 s up to 2.2 s

($\mu_{u,clet}$ and $\mu_{u,cloud}$ are scaled accordingly). We fix the number of users $N = 15$ as in the previous experiments and set the number of cloudlet servers to $n = 2$. The results are shown in Figure 4(c). As we can see, at low-medium load the users take fully advantage of the cloudlet resources ($x_{u,clet} = 1$), except when the task execution time is very small ($1/\mu_{u,m} = 0.1$ s), in which case it is more convenient to execute the task locally on the mobile device. In particular, $x_{u,clet}$ remains equal to 1 until $1/\mu_{u,m} \leq 1.3$ s, corresponding to a cloudlet utilization of about 0.49. From this point onwards, an ever growing fraction of tasks is offloaded to the remote cloud, because in these experiments the cloudlet is overloaded by the larger task execution time. Hence, when the cloudlet is overloaded, it is more profitable to dispatch a fraction of tasks to the remote cloud rather than to the mobile device, because the delay introduced by the wireless network and the Internet is compensated by the faster execution on the remote cloud.

We now analyze the impact of the constraint on the power consumption, which has been neglected in the previous experiments where we set $P_{u,max} = \infty$. Following [23], we set $P_{u,m} = 0.9$ W, $P_{u,t} = 1.3$ W, and we study how the offloading strategy changes as $P_{u,max}$ increases from 0.112 W to 0.125 W. We also increase the transfer time over the wireless network, setting $1/\mu_{u,wl} = 0.5$ s (for example, we can suppose that the access network is congested), while keeping $1/\mu_{u,m} = 0.5$ s, so that the power consumption to transmit the task weighs more than the power consumed to execute the task locally. The results are shown in Figure 4(d). As we can see, when $P_{u,max} \cong 0.124$ W, the users' strategy is to offload to the cloudlet more than 20% of the computation. Indeed, the high transmission time is compensated by the cloudlet faster response time and the power constraint is still satisfied. However, as the power constraint becomes more stringent, the user strategy is to reduce progressively the fraction of offloaded computation because offloading consumes too much energy due to the high transfer time over the wireless network.

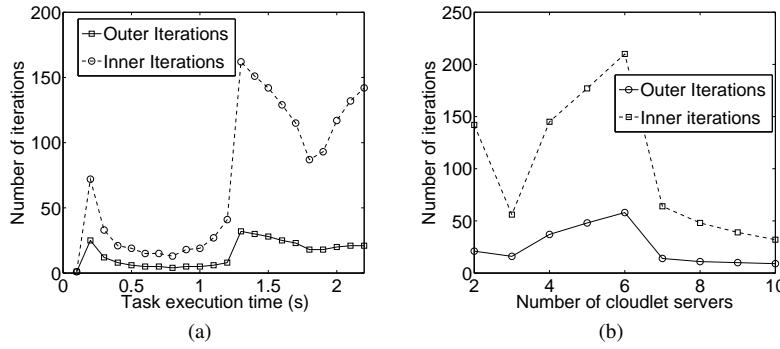


Fig. 6 Inner and outer number of iterations of the distributed algorithm

We now turn our attention to the convergence speed of the proposed distributed algorithm. In our experiments we set $(x^0, \rho^0) = (x^k, \rho^k)$ each time Algorithm 2 is executed (step S.0). Furthermore, the values of x and ρ in the first outer loop are taken

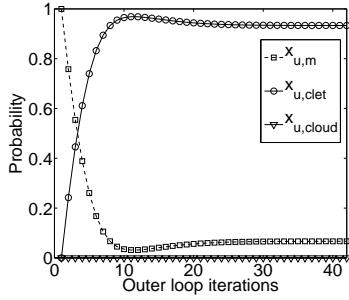


Fig. 7 Intermediate algorithm solution strategies

equal to $\{x_{u,m}, x_{u,clet}, x_{u,cloud}\} = \{1, 0, 0\} \forall u$ and 0 respectively. Figure 6 shows the number of iterations needed to compute the equilibrium policies. For space limits, we show only the results for the second and third sets of experiments; however, similar conclusions hold for the other experiments. If we compare Figures 6(a) and 6(b) with Figures 4(c) and 4(b), we can see that whenever the cloudlet is overloaded and the strategy requires to distribute the tasks between the remote cloud or the mobile device, the number of required inner iterations, i.e. the number of times step S.2 of Algorithm 2 is executed, grows up to 200. However, the number of iterations can be decreased up to one third by using as initial state of Algorithm 1 the previously computed equilibrium. This could be a promising solution to speed up the algorithm convergence in a real environment, where we could expect that the users gradually join and leave the system. Furthermore, intermediate solutions that progressively approximate the new system equilibrium can also be used as they are computed, rather than waiting the algorithm to stop. For example, Figure 7 shows the intermediate outer loop solutions, i.e. the x_u^k values (this figure refers to the same setting of the first set of experiments when the number of cloudlet users is equal to 60). As we can see, from 10 outer iterations onwards, we already have a good approximation of the system equilibrium.

Finally, observe that in our experiments we never had problems due to assumption (17) in Theorem 1 not being satisfied. Indeed, our experiments showed that such assumption is not critical from a practical point of view. Nevertheless, the parameter α should be carefully tuned depending on the cloudlet load to ensure the algorithm convergence. As a rule of thumb, the higher the expected cloudlet load, the larger α should be in order to ensure convergence. In our experiments, we used $\alpha = 0.1$ to accommodate the higher loads (even though a smaller α would have ensured faster convergence at lower loads).

7.2 Comparison with the Cooperative Strategy

We now compare the proposed non-cooperative strategy with a cooperative one in order to investigate the performance degradation caused by the selfish users behavior. In a cooperative scenario we determine the *social optimum*, that is the solution

which maximizes the sum of the users objective functions, i.e. $\sum_{v=1}^N \lambda_v R_v$, subject to the union of all the user constraints. Under Assumption A the corresponding problem is a convex optimization problem with linear constraints. We studied the cooperative solution with the same set of parameters used in the first set of experiments as the number of users varies. Figure 8(a) shows the user optimal strategy. As we can

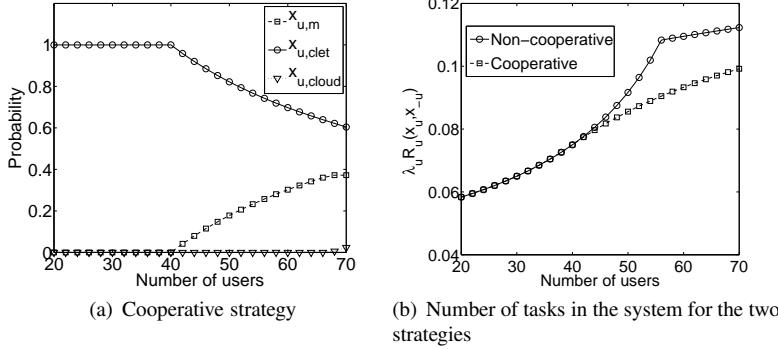


Fig. 8 Comparison of the proposed non-cooperative strategy with the cooperative one

see, differently from the non-cooperative strategy, the users switch earlier their computation to the mobile devices ($N = 40$ against $N = 56$) because they are not acting selfishly. Indeed, we can expect that the behavior of a selfish user is to offload as much as possible its computation to the cloud, regardless of what the others do. However, by doing so users experience a worse performance as the overall load increases, as shown in Figure 8(b). This is the so called “price of anarchy”. Note also that, under light load, the two strategies achieve the same results, because the cloudlet capacity can accommodate all the tasks.

8 Conclusions

We have considered the problem of computation offloading in a mobile cloud computing scenario, motivated by the increasing interest in this architectural paradigm. In particular, as suggested by recent literature on this topic, we have considered a three-tier architecture where mobile nodes have the choice of offloading their computation to a nearby resource-constrained cloudlet or to a distant tier of resourceful cloud servers. While previous works have either dealt with single-user scenarios without considering the overall system or at most with centralized global approaches to tackle the interactions among different mobile users on a resource-limited cloud, in this paper we have focused on a non-cooperative usage scenario where individual users try to take advantage selfishly of the available resources.

We have adopted a game theoretic approach to investigate the dynamics of user interactions, modeling the offloading strategy of mobile users as a Generalized Nash Equilibrium Problem. We have shown existence of an equilibrium and have provided

a distributed algorithm to compute an equilibrium strategy for each user. Through a set of numerical experiments we have illustrated the properties of the equilibrium that can be achieved, by comparing also it with results obtained by a centralized optimal management of the cloudlet resources. The proposed distributed algorithm has a solid theoretical foundation and is appealing for a real-world implementation, since it requires only a limited amount of information that can be easily obtained.

A number of topics may be explored in future research, including a multi-class task model for a finer modeling of the tasks launched by each user, as well as a monetary cost model to use the cloud servers. Furthermore, while we have proved the existence of a solution for the variational inequality, a further step with some practical relevance is the selection of the given variational solution if more than one exists. Besides working on these methodological extensions, we also plan to implement the distributed algorithm in a system prototype, to validate the results in a real environment.

References

1. Ermyas Abebe and Caspar Ryan. Adaptive application offloading using distributed abstract class graphs in mobile environments. *J. Syst. Softw.*, 85(12):2755–2769, 2012.
2. Saeid Abolfazli, Zohreh Sanaei, Ejaz Ahmed, Abdullah Gani, and Rajkumar Buyya. Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. *IEEE Communications Surveys Tutorials*, 2013. To appear.
3. Eitan Altman, Urtzi Ayesta, and Balakrishna Prabhu. Load balancing in processor sharing systems. In *Proc. of 3rd Int'l Conf. on Performance Evaluation Methodologies and Tools*, ValueTools '08, 2008.
4. Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan. Advancing the state of mobile cloud computing. In *Proc. of MCS 2012*, pages 21–28. ACM, 2012.
5. Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, And Formulas With Application To Linear Systems Theory*. Princeton University Press, 2005.
6. Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
7. Byung-Gon Chun, Sungewan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proc. of EuroSys 2011*, pages 301–314, 2011.
8. Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users. In *Proc. of 2012 IEEE Int'l Conf. on Pervasive Computing and Communications*, PerCom '12, pages 122–127, March 2012.
9. Eduardo Cuervo, Aruna Balasubramanian, Dae-Ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. MAUI: making smartphones last longer with code offload. In *Proc. of MobiSys 2010*, pages 49–62. ACM, 2010.
10. Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. On generalized Nash games and variational inequalities. *Operations Research Letters*, 35(2):159–164, 2007.
11. Francisco Facchinei and Christian Kanzow. Generalized Nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, 2010.
12. Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional Variational Inequalities and Complementarity Problems*, volume 1,2. Springer Verlag, 2003.
13. Francisco Facchinei and Jong-Shi Pang. Nash equilibria: the variational approach. In Daniel P. Palomar and Yonina C. Eldar, editors, *Convex Optimization in Signal Processing and Communications*, pages 443–493. Cambridge Books, 2009.
14. Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Gener. Comput. Syst.*, 29(1):84–106, 2013.
15. Ioana Giurgiu, Oriana Riva, and Gustavo Alonso. Dynamic software deployment from clouds to mobile devices. In *Proc. of Middleware 2012*, pages 394–414. Springer-Verlag, 2012.

16. Kiryong Ha, Padmanabhan Pillai, Grace A. Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *Proc. of IEEE Int'l Conf. on Cloud Engineering*, IC2E '13, pages 166–176, 2013.
17. Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
18. Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *Proc. of 1st ACM Workshop on Mobile Cloud Computing & Services*, MCS '10, 2010.
19. Shigeru Imai and Carlos A. Varela. Light-weight adaptive task offloading from smartphones to nearby computational resources. In *Proc. of 2011 ACM Symposium on Research in Applied Computation*, pages 146–152, 2011.
20. Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.
21. Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proc. of IEEE INFOCOM 2012*, pages 945–953, March 2012.
22. Sokol Kosta, Vasile Perta, Julinda Stefa, Han Hui, and Alessandro Mei. Clone2Clone (C2C): Peer-to-peer networking of smartphones on the cloud. In *Proc. of 5th USENIX Workshop on Hot topics in Cloud Computing*, HotCloud '13, June 2013.
23. Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *IEEE Computer*, 43(4):51–56, April 2010.
24. Allen B. MacKenzie and Luiz A. DaSilva. Game theory for wireless engineers. *Synthesis Lectures on Communications*, 1(1):1–86, 2006.
25. Peter Mell and Timothy Grance. The NIST definition of cloud computing. *NIST Special Publication 800-145*, September 2011.
26. Shumao Ou, Kun Yang, and Jie Zhang. An effective offloading middleware for pervasive services on mobile devices. *Pervasive Mob. Comput.*, 3(4):362–385, August 2007.
27. Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter Rentfrow. METIS: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *Proc. of 2013 IEEE Int'l Conf. on Pervasive Computing and Communications*, PerCom '13, 2013.
28. Reza Rahimi, Nalini Venkatasubramanian, Sharad Mehrotra, and Athanasios Vasilakos. MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proc. of IEEE 5th Int'l Conf. on Utility and Cloud Computing*, UCC '12, pages 83–90, November 2012.
29. Reza Rahimi, Nalini Venkatasubramanian, and Athanasios Vasilakos. MuSIC: On mobility-aware optimal service allocation in mobile cloud computing. In *Proc. of IEEE 6th Int'l Conf. on Cloud Computing*, Cloud '13, July 2013.
30. Mahadev Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001.
31. Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.
32. Gesualdo Scutari, Francisco Facchinei, Jong-Shi Pang, and Daniel P. Palomar. Real and complex monotone communication games. *Arxiv.org, Dec. 2012, available at*, <http://arxiv.org/abs/1212.6235>, 2012.
33. Mohsen Sharifi, Somayeh Kafaie, and Omid Kashefi. A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys & Tutorials*, 14(4):1232–1243, 2012.
34. Muhammad Shiraz, Abdullah Gani, Rashid Khokhar, and Rajkumar Buyya. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys Tutorials*, 2013. To appear.
35. Narseo Vallina-Rodriguez and Jon Crowcroft. ErdOS: achieving energy savings in mobile OS. In *Proc. of 6th Int'l Workshop on MobiArch*, MobiArch '11, pages 37–42. ACM, 2011.
36. Tim Verbelen, Tim Stevens, Filip De Turck, and Bart Dhoedt. Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. *Future Gener. Comput. Syst.*, 29(2), February 2013.
37. Lei Yang, Jannong Cao, and Hui Cheng. Resource constrained multi-user computation partitioning for interactive mobile cloud applications. Technical report, Dept. of Computing, Hong Kong Polytechnic Univ., August 2012.
38. Lei Yang, Jannong Cao, Yin Yuan, Tao Li, Andy Han, and Alvin Chan. A framework for partitioning and execution of data stream applications in mobile cloud computing. *Sigmetrics Perform. Eval. Rev.*, 40(4):23–32, April 2013.