Joint Optimization of Radio Resources and Code Partitioning in Mobile Cloud Computing

Paolo Di Lorenzo, Member, IEEE, Sergio Barbarossa, Fellow, IEEE, and Stefania Sardellitti, Member, IEEE,

Department of Information, Electronics, and Telecommunications (DIET), Sapienza University of Rome, Via Eudossiana 18, 00184 Rome, Italy.
E-mail: {dilorenzo, sergio}@infocom.uniroma1.it, stefania.sardellitti@uniroma1.it.

Abstract

The aim of this paper is to propose a computation offloading strategy, to be used in mobile cloud computing, in order to minimize the energy expenditure at the mobile handset necessary to run an application under a latency constraint. We exploit the concept of call graph, which models a generic computer program as a set of procedures related to each other through a weighted directed graph. Our goal is to derive the partition of the call graph establishing which procedures are to be executed locally or remotely. The main novelty of our work is that the optimal partition is obtained jointly with the selection of the transmit power and constellation size, in order to minimize the energy consumption at the mobile handset, under a latency constraint taking into account transmit time, packet drops, and execution time. We consider both a single channel and a multi-channel transmission strategy, thus proving that a globally optimal solution can be achieved in both cases with affordable complexity. The theoretical findings are corroborated by numerical results and are aimed to show under what conditions, in terms of call graph topology, communication strategy, and computation parameters, the proposed offloading strategy can provide a significant performance gain.

Index Terms

Mobile cloud computing, small cells, computation offloading, call graph, energy minimization, delay constraint.

Submitted to IEEE Transactions on Wireless Communications. The research leading to these results has received funding from the Seventh Framework Programme (FP7/2007-2013) under grant agreement Nr. 318784 STP TROPIC.
I. INTRODUCTION

Computation offloading has attracted a lot of research efforts as a way to augment the capabilities of resource-constrained and energy-hungry mobile handsets by migrating computation to more resourceful servers. Offloading is useful either to enable smartphones to run more and more sophisticated applications, while meeting strict delay constraints, or to prolong the battery lifetime by limiting the energy spent at the mobile handset to run a given application, still under application-dependent delay constraints. In the current mobile handset market, user demand is increasing for several categories of energy-hungry applications. Video games are becoming more and more popular and they have large energy requests. Mobile users are also increasingly watching and uploading streaming video from YouTube [1]-[3]. Furthermore, mobile devices are increasingly equipped with new sensors that produce continuous streams of data about the environment. New applications relying on continuous processing of the collected data are emerging, such as car navigation systems, pedometers, and location-based social networking, just to name a few. All these applications are severely energy demanding for today’s mobile handsets. On the other hand, the trend in battery technology makes it unlikely that the energy problem will disappear in the near future. Indeed, recent researches on the adoption of new battery technologies, e.g., new chemicals, are unlikely to significantly improve battery lifetime [3]-[4]. This explains why mobile computation offloading is actually a promising idea to overcome the limitations of the mobile handsets.

The idea of offloading computations is not new. A recent broad recent survey on computation offloading for mobile systems is [5]. A great impulse to computation offloading has come through the introduction of cloud computing [6]. One of the key features of cloud computing is virtualization, i.e. the capability to run multiple operating systems and multiple applications over the same machine (or set of machines), while guaranteeing isolation and protection of the programs and their data. Through virtualization, the number of virtual machines can scale on demand, thus improving the system computational efficiency. Providing wireless access to the cloud has enormous potentials to augment the capabilities of mobile handsets, through what is known as mobile cloud computing [21]. One of the limitations of today mobile cloud computing is the latency experienced in the propagation of information through a wide area network (WAN). As pointed out in [18], [36] [27], humans are acutely sensitive to delay and jitter, and it is very difficult to control these parameters at WAN scale: As latency increases, interactive response suffers. Furthermore, mobile access through macro base stations (MBS) might require large power consumptions. To overcome this limitation, the authors of [36] introduced the concept of cloudlet, i.e. the possibility for the mobile handsets to access nearby static resourceful computers, linked to a
distant cloud through high speed wired connections. In [36], cloudlets are imagined to be deployed much like Wi-Fi access points today. Comparisons of energy expenditure for computation offloading using 3G and Wi-fi systems are given in [18] and [27], where it is shown how the benefits of offloading are most impressive when the remote server is nearby, such as on the same LAN as the Wi-Fi access point, rather than using 3G technology. Having cloud functionalities on the Wi-Fi would certainly improve accessibility with low latencies. However, Wi-Fi operates under a different communication standard with respect to cellular networks. This induces some extra difficulty in terms of vertical handover and lack of Quality of Service (QoS) guarantee. In practice, it is clearly better to have a single stand-alone technology enabling mobile access to the cloud from mobile phones without switching communication standard. Within the current European Project named TROPIC, we have recently proposed an alternative approach merging the concepts of small cell networks with cloud computing [37] into the so called femto-cloud computing. The idea is to endow small cell base stations, also known as Small Cell e-Node B (SCeNB), with enhanced cloud functionalities to make them useful to provide mobile users with proximity access to the cloud. The novel base stations are called Small Cell cloud-enhanced e-Node B (SCceNB). Exploiting the deployment of LTE SCceNB’s, the femto-cloud architecture offers proximity access with easier vertical hand-over and better QoS with respect to Wi-Fi access.

Our goal in this paper is to propose a computation offloading method operating jointly over the communication and computation aspects. The scope is to devise an offloading strategy that minimizes the energy consumption at the mobile handset under a latency constraint. Communicating under a latency constraint is a classical problem that received considerable attention, see, e.g., [28]-[35]. At the same time, computation offloading has been studied in several works, as e.g., [7]-[27]. The offloading decisions are usually made by analyzing parameters including bandwidths, server speeds, available memory, server loads, and the amounts of data exchanged between servers and mobile systems. The solutions include code partitioning programs [10], [16], [18], [25], [7], [20], [11], [8], [12], and predicting parametric variations in application behavior and execution environment [13]-[14]. Offloading typically requires code partitioning, aimed to decide which parts of the code should run locally and which parts should be offloaded, depending on contextual parameters, such as computational intensity of each module, size of the program state to be exchanged to transfer the execution from one site to the other, battery level, delay constraints, channel state, and so on. Program partitioning may be static or dynamic. Static partitioning has the advantage of requiring a low overhead during execution, but it works well only if the parameters related to the offloading decisions are accurately known in advance or predicted. In contrast, enforcing a dynamic decision mechanism makes possible to adapt the method to different operating conditions and
to cope with different degrees of uncertainties. Given the variability of the wireless channel, dynamic partitioning seems more appropriate, but it has an associated higher signaling overhead, which must be taken under control. A dynamic approach to computation offloading consists in establishing a rule to decide which part of the computations may be advantageously offloaded, depending on channel conditions, server state, delay constraints, and so on. In particular, reference [18] proposes MAUI, a system that enables fine-grained energy-aware code offloading to the fixed infrastructure. MAUI decides at runtime which program modules should be executed remotely, through an optimization engine that achieves the best energy saving possible under the mobile device’s current connectivity constrains. References [17], [19] propose CloneCloud, which is a flexible application partitioner that enables unmodified mobile applications running in an application-level virtual machine to seamlessly offload part of their execution from mobile devices onto device clones operating in a computational cloud. A recent study on both mobile computation offloading and mobile software/data backups in real-life scenarios is [27], where it is illustrated a precise evaluation of the feasibility and costs of computation offloading in terms of bandwidth and energy consumption on real devices.

In this work we propose a novel approach that optimizes code partitioning and radio resource allocation jointly. To find the optimal code partitioning, we exploit the concept of call graph of a program [38]-[41], modeling the relations between procedures of a computer program through a directed acyclic graph. Our objective is to find the code partition and the radio resource allocation (power and constellation size) that minimize the energy consumption at the mobile side, under a latency constraint that incorporates communication and computation times. We consider both a single channel and a multi-channel transmission strategies. We provide theoretical results proving the existence of a unique solution of the resource allocation problem. The theoretical findings are then corroborated with numerical results, illustrating under what kind of call graph’s topologies, radio channel conditions, and communication strategies, the proposed method can provide a significant energy saving at the mobile handset.

The paper is organized as follows. Section II describes the problem formulation, starting from the definition of call graphs, and illustrating then the proposed solution based on the joint power/code optimization for delay constrained energy minimization, both in the single and multiple channel case. Section III shows some simulation results aimed to confirm our theoretical findings and to quantify the energy savings that the proposed optimization method achieves. In particular, the performance of the method are strongly application dependent, i.e., the structure of the call graph deeply affects the final offloading decision. Furthermore, we also quantify the effect of the wireless channel between MUE and SCceNB on the final offloading decision, thus showing how the performance are affected by the degrees
of freedom of the channel. Finally, Section IV draws some conclusions.

II. PROBLEM FORMULATION

In this section, we start by introducing the notion of call graph of a program, which will be instrumental to formulate our joint optimization of radio and computation resources. The representation of a program through its call graph is instrumental to design an offloading strategy that decides to offload only some modules of the program, depending on the computation requirements of each module, the size of the program state that needs to be exchanged to transfer the execution from one site to the other, and the channel conditions. Then, we formulate the optimization problem as the minimization of the overall energy spent to run a program, under latency and power constraints imposed by the application and the radio interface, respectively. We consider first the case of a single communication channel between MUE and SCceNB. Then, we generalize the approach to the multicarrier case.

A. Call Graph

A call graph is a useful representation that models the relations between procedures of a computer program into the form of a directed graph \( G = (V, E) \). The call graph represents the call stack as the program executes. Each vertex \( v \in V \) represents a procedure in the call stack, and each directed edge \( e = (u, v) \) represents an invocation of procedure \( v \) from procedure \( u \). The call graph contains all the relationships among the procedures in a program and, in general, it includes auxiliary information concerning for instance the number of instructions within each procedure and the global data shared among procedures. For non recursive languages with reasonable assumptions on the program structure \([38]\), the call graph is a directed, acyclic graph.

In our computation offloading framework, we label each vertex \( v \in V \) with the energy \( E^l_v \) it takes to execute the procedure locally, and with the overall number of instructions \( w_v \) (CPU cycles), which the procedure is composed of. Then, we denote by \( T^l_v \) and \( T^r_v \) the time it takes to execute the program module locally or remotely, respectively. They are directly related to \( w_v \) through the following expressions

\[
T^l_v = \frac{w_v}{f_l}, \quad \text{and} \quad T^r_v = \frac{w_v}{f_s},
\]

where \( f_l \) and \( f_s \) are the CPU clock speeds (cycles/seconds) at the MUE and at the server, respectively. At the same time, each edge \( e = (u, v) \) is characterized by a label describing the number of bits \( N_{u,v} \) representing the size of the program state that needs to be exchanged to transfer the execution from node \( u \) to node \( v \). In general, some procedures cannot be offloaded, like, e.g., the program modules controlling
the user interface or the interaction with I/O devices. Examples of these categories include codes that determine the location of a smartphone by reading from the GPS device, or code that uses a network connection to perform an e-commerce transaction, e.g., to purchase an item from an online store. The set of procedures that should be executed locally, at the mobile site, is denoted by $V_l$. An example of call graph of a generic application, composed of $V = 4$ nodes and $E = 3$ edges, is shown in Fig. 1. The call graph has a tree structure, where node 1 is the root node.

**B. Joint Optimization of Radio Parameters and Code Partitioning in the Single Channel Case**

The goal of this section is to formulate an optimization problem aimed at determining which modules of an application’s call graph should be executed locally and which ones should be executed remotely. Intuitively speaking, the modules more amenable for offloading are the ones requiring intensive computations and limited exchange of data to transfer the execution from one site to the other. Our goal now is to make this intuition the result of an optimization procedure. To this end, we formulate the offloading decision problem jointly with the selection of the transmit power and the constellation size used for transmitting the program state necessary to transfer the execution from the mobile handset to the cloud or viceversa. The objective is to minimize the energy consumption at the mobile site, under a power budget constraint at the mobile handset, and a latency constraint taking into account the time to transfer the execution and the time necessary to execute the module itself. This constraint is what couples the computation and communication aspects of the problem. We assume that the set of instructions to be executed is available at both MUE and SCceNB. If not, they are supposed to be downloaded by the server through a high speed wired link.
Let us indicate with $I_v$ the indicator variable, which is equal to one, if the procedure $v$ of the call graph is executed remotely, or zero, if it is executed locally. We also denote by $p_{u,v}$ the power spent to transmit the program state between the procedures $u$ and $v$. The set of all powers is collected in the vector $p = \{p_{u,v}\}_{(u,v) \in E} \in \mathbb{R}^{\text{card}(E)}$, where $\text{card}(E)$ is the cardinality of set $E$. To decide which modules of the call graph should be executed remotely, we need to solve the following optimization problem:

\[ \begin{align*}
\text{min} & \quad \sum_{v \in V} (1 - I_v) \cdot E_v^l + \sum_{(u,v) \in E} f_{u,v} (I_u, I_v, p_{u,v}) \\
\text{s.t.} & \quad \sum_{v \in V} \left( (1 - I_v)T_v^l + I_vT_v^r \right) + \sum_{(u,v) \in E} g_{u,v} (I_u, I_v, p_{u,v}) \leq L \\
& \quad I_v \in \{0, 1\}, \quad I_v = 0, \quad \forall v \in V_l, \\
& \quad 0 < p_{u,v} \leq P_T \quad \forall (u, v) \in E,
\end{align*} \]

where $I = \{I_v\}_{v \in V}$. The objective function in (2) represents the total energy spent by the MUE for executing the application. In particular, the first term in (2) is the sum (over all the vertices of the call graph) of the energies spent for executing the procedures locally, whereas the second term is the sum (over the edges of the call graph) of the energies spent to transfer the execution from the MUE to the SCceNB. The function $f_{u,v} (I_u, I_v, p_{u,v})$ in (2), reported in Table I (a), describes the energy spent by the MUE when procedure $u$ calls procedure $v$ and its dependency on the indices $u$ and $v$. Note that an energy cost occurs only if the two procedures $u$ and $v$ are executed at different locations, i.e. $I_u \neq I_v$.

More specifically, if $I_u = 0$ and $I_v = 1$, the energy cost is equal to the energy $J_{u,v} (p_{u,v})$ needed to transmit the program state $N_{u,v}$ from the MUE to the SCceNB, whereas, if $I_u = 1$ and $I_v = 0$, the cost is equal to the energy $\varepsilon_{u,v}$ needed by the MUE to decode the $N_{u,v}$ bits of the program state transmitted back by the SCceNB. The cost $\varepsilon_{u,v}$ is not a function of the MUE’s transmitted power and it depends only on the size of the program state $N_{u,v}$. The constraint in (3) is a latency constraint and it contains three terms: the first term is the time needed to compute the procedures locally, the second is the time needed to compute the procedures remotely, and the third term is the delay resulting from transferring the program state from one site (e.g., the MUE) to the other (e.g., the cloud). The constant $L$ represents the maximum latency dictated from the application. The delay function $g_{u,v} (I_u, I_v, p_{u,v})$ in (3), reported in Table I (b), represents the delay associated to letting procedure $u$ call procedure $v$. From Table I (b), we note that no delay occurs if the two procedures $u$ and $v$ are both executed in the same location, i.e. $I_u = I_v$. Furthermore, if $I_u = 0$ and $I_v = 1$, the delay is equal to the time $D_{u,v} (p_{u,v})$ needed to transmit the program state $N_{u,v}$ from the MUE to the SCceNB, whereas, if $I_u = 1$ and $I_v = 0$, the delay
is equal to the time \( \gamma_{u,v} \) needed by the MUE to get the \( N_{u,v} \) status bits back from the SCceNB. The delay function \( D_{u,v}(p_{u,v}) \) in Table I will be made explicit later on. The term \( \gamma_{u,v} \) is not a function of the MUE’s transmitted power and it depends only on the size of the program state \( N_{u,v} \) to be transferred. The constraint in (4) specifies that the variables \( I_v \) are binary and that for all procedures contained in the set \( V_l \), which is the set of procedures that are to be executed locally, \( I_v = 0 \). The last constraint, in (5), is a power budget constraint on the maximum transmit power \( P_T \).

**Remark 1:** The problem formulation in (2)-(5) is an extension of the MAUI strategy from [18], where it was proposed an integer linear program aimed at finding the optimal program partitioning strategy that minimizes the MUE’s energy consumption, subject to a latency constraint. The difference with respect to our work is that in [18], the quantities \( D_{u,v}(p_{u,v}) \) and \( J_{u,v}(p_{u,v}) \) were assumed to be constant, whereas in our case those quantities are optimized jointly with the program partition, acting on the transmit power and on the constellation size.

It is now time to express the quantities \( D_{u,v}(p_{u,v}) \) and \( J_{u,v}(p_{u,v}) \) as a function of channel state and transmit power. We assume an adaptive modulation scheme that selects the QAM constellation size as a function of channel conditions and computational requirements. As a consequence, the minimum time \( D_{u,v}(p_{u,v}) \) necessary to transmit \( N_{u,v} \) bits of duration \( T_b \) over an additive white Gaussian noise (AWGN) channel is

\[
D_{u,v}(p_{u,v}) = \frac{N_{u,v}T_b}{\log_2 \left( 1 + \frac{p_{u,v}|h|^2}{\Gamma(\text{BER})d^\beta N_0} \right)},
\]

where the denominator represents the number of bits/symbol, \( p_{u,v} \) is the transmit power, \( d \) is the distance between MUE and SCceNB, \( \beta \) is the path loss exponent, \( N_0 \) denotes the noise power and \( h \) is the
channel fading coefficient (normalized to distance); \( \Gamma(BER) = \frac{2\log(5 \cdot BER)}{3} \) represents the so called SNR margin\(^1\), introduced to meet the desired target bit error rate (BER) with a QAM constellation. Note that the gap factor \( \Gamma(BER) \) is valid under the assumption \( \Gamma(BER) > 0 \), i.e. \( BER < 1/5 \).

In practice, the time necessary to let the server receive the correct input bits is often greater than \( (6) \) because of retransmission of erroneous packets. Let us denote by \( S \) the random variable indicating the number of retransmissions over the wireless link and with \( P_e \) the corresponding packet error rate. Assuming independent errors over different packets, the probability of transmitting a packet \( s \) times is given by \( P\{S=s\} = P_e^{s-1}(1 - P_e) \). The expected number of transmissions is then

\[
\bar{S} = \sum_{s=1}^{\infty} s \cdot P_e^{s-1}(1 - P_e) = \frac{1}{1 - P_e}. \quad (7)
\]

The relation between BER and \( P_e \) is dictated by the channel coding scheme adopted in the link between the MUE and the SCceNB. The average delay associated to the correct reception of \( N_{u,v} \) bits at the server side, incorporating packet duration and retransmissions, is then

\[
\bar{D}_{u,v}(p_{u,v}) = \bar{S} \cdot D_{u,v}(p_{u,v}) = \frac{N_{u,v} T_b}{(1 - P_e) \log_2 \left( 1 + \frac{p_{u,v} |h|^2}{\Gamma(BER)d^\beta N_0} \right)} \quad (8)
\]

Expression \( (8) \) shows that offloading may be advantageous if the distance \( d \) between MUE and SCceNB is sufficiently small, so that the transmission rate is high. This is a further justification for favouring the proximity radio access to the cloud through SCceNB’s. Now, setting

\[
a = \frac{|h|^2}{\Gamma(BER)d^\beta N_0} \quad (9)
\]

and adopting natural logarithms, Eq. \( (8) \) can be rewritten as

\[
\bar{D}_{u,v}(p_{u,v}) = \frac{N'_{u,v}}{\log \left( 1 + ap_{u,v} \right)}, \quad (10)
\]

where \( N'_{u,v} = \frac{N_{u,v} T_b \log 2}{(1 - P_e)} \). Expression \( (10) \) represents the average delay associated to the data transfer required to execute procedure \( v \) remotely, when called by procedure \( u \). Thus, exploiting \( (10) \), the average energy \( \bar{C}_{u,v} \), associated to the transfer of the program state \( N_{u,v} \), is given by

\[
\bar{J}_{u,v}(p_{u,v}) = p_{u,v} \bar{D}_{u,v}(p_{u,v}) = \frac{N'_{u,v} p_{u,v}}{\log \left( 1 + ap_{u,v} \right)} \quad (11)
\]

Thus, using expressions \( (1), (11), \) and \( (10) \), the solution of problem \([P.1]\) provides a joint optimization of radio resources (power and bits/symbol) and code partitioning for mobile computation offloading. Since

\(^1\)Whenever the base is unspecified, \( \log(x) \) has to be intended as the natural base logarithm.
the state variables $I_u$ are integer, problem $[P.1]$ is inherently a mixed integer programming problem [45]. Hence, its solution in principle requires to explore all possible combinations in $I$, solve for the power vector $p$, and then compare the final values of the objective function in order to choose the best configuration. The size of the problem might explode because, in principle, the number of combinations grows exponentially with the number of vertices $V$ of the call graph. Nevertheless, a series of simplifications are possible. The first useful remark is that, with respect to the integer variables $I_v$, the problem is a linear (binary) integer programming problem. This remark is useful because there exist efficient algorithms, such as the branch and bound algorithm for example, to solve a linear integer problem with reasonable complexity [45]. An alternative approach to handle the integer part is to resort to a backward induction approach to find out the optimal partition [46]. To this end, starting from the call graph, we build a tree containing all possible offloading choices, for each module. Then, starting from the leaves of the tree, we explore the tree back up to the root, by removing all the branches which are suboptimal. Of course, the overall complexity depends on the granularity of the call graph construction: A fine-grain model provides better performance, but with a much higher cost; conversely, a coarse-grain call graph provides worse performance, but with affordable complexity. An important simplification of the problem comes from observing that, for any set of integer values $I_v$, the remaining optimization over the power coefficients is a convex problem. This means that its solution is unique and it can be achieved with very efficient numerical algorithms. We prove now the previous statement.

Let us consider a generic combination $c \in C$, where $C$ is the set of all possible combinations of the binary variables $I_v$, $v \in V$. For each combination $c$, the value of the variables $I_v$ is fixed to some value $I^c_v$, and problem $[P.1]$ becomes a radio resource allocation problem where we are attempting to minimize the average energy spent for transmission subject to a constraint on the average transmission delay, i.e.,

$$[P.2] \quad \min_{p_c} \quad \sum_{(u,v) \in I_c} \frac{N'_{u,v} p_{u,v}}{\log (1 + a p_{u,v})}$$

subject to

$$\sum_{(u,v) \in I_c} \frac{N'_{u,v}}{\log (1 + a p_{u,v})} \leq L'$$

$$0 < p_{u,v} \leq P_T, \quad \forall (u, v) \in I_c.$$

where, for any combination $c$, $I_c$ is the set of edges $(u,v) \in E$ for which $I^c_u = 0$ and $I^c_v = 1$; $p_c = \{p_{u,v}\}_{(u,v) \in I_c}$ is vector containing all the powers, and

$$L' = L - \sum_{v \in V} \left[ (1 - I^c_v) \frac{w_v}{f_t} + I^c_v \frac{w_v}{f_s} \right] - \sum_{(u,v) \in I_c} \gamma_{u,v} \quad (12)$$
where $I'_c$ is the set of edges $(u, v) \in E$ for which $I^c_u = 1$ and $I^c_v = 0$. Going from $[P.1]$ to $[P.2]$, we have eliminated all the terms that do not depend explicitly on the transmit powers, since they do not affect the optimization problem $[P.2]$. The problem in $[P.2]$ is still nonconvex because the objective function is actually concave in the power allocation vector $p_c$. Nevertheless, we can prove the following result:

**Theorem 1:** If the following conditions

$$L' > 0 \quad \text{and} \quad \alpha \geq e^\frac{\sum_{(u,v)\in I_c} N'_{u,v}}{P_T} - 1,$$

(13)

are satisfied, then: (a) the feasible set of problem $[P.2]$ is non-empty; (b) the constrained energy minimization problem in $[P.2]$ is equivalent to the convex problem $[P.5]$ (see Appendix A); (c) the latency constraint in $[P.2]$ is always satisfied with strict equality.

**Proof:** See Appendix A.

**Remark 2:** By virtue of Theorem 1 and the results in Appendix A, the delay constrained energy minimization problem in $[P.2]$ is equivalent to a convex problem with strictly convex objective function. Thus, if the feasible set is non-empty, the problem admits a unique global solution that can be found using numerically efficient algorithms [47].

The main steps for solving the joint optimization of radio resources and code partitioning for mobile cloud computation offloading are summarized in the following Algorithm.

**Algorithm: Joint Optimization of Radio Parameters and Call Graph’s Partitioning**

1) Given a generic call graph, compute the set of combinations $C = \{c\}$.

2) For every combination $c$, check the feasibility of problem $[P.2]$ through the relation (13):

   a) if the problem is feasible, compute its optimal solution $p^*_c = \{p^*_u, v\}_{(u,v)\in I_c} \in \mathbb{R}^{\text{card}(I_c)}$ by solving the problem $[P.5]$ (Appendix A). Furthermore, evaluate the overall energy spent to run the application, for the given combination $c$, as:

   \[ E_c = E^l_c + E^r_c(p^*_c) = \sum_{v \in V} (1 - I^c_v) \cdot E^l_v + \sum_{(u,v) \in E} f_{u,v}(I^c_u, I^c_v, p^*_{u,v}) \]

   (14)

   where $f_{u,v}(I^c_u, I^c_v, p^*_{u,v})$ is defined as in Table I (a), with $\bar{J}_{u,v}(p_{u,v})$ given by (11).

   b) if the problem is not feasible, set $E_c = E^g_c$, where $E^g_c$ is the energy spent by the MUE to run the entire application locally.

3) Select the pair $(c^*, p^*_c) = \arg \min_c E_c$. 

May 7, 2014 DRAFT
C. Joint Optimization of Radio parameters and Code Partitioning in the Multiple Channel Case

We now extend the previous offloading optimization strategy to the wideband channel case, where the MUE transmits over a set of $K$ parallel subbands. In this case, we have the extra degree of freedom on how to distribute the transmit power across the parallel channels. In the multi-channel case, the average delay associated to the correct reception of $N_{u,v}$ bits at the server side, incorporating packet duration and retransmissions, is

$$
\bar{D}_{u,v}(p_{u,v}) = \frac{N_{u,v} T_b}{(1 - P_e) \sum_{k=1}^{K} \log_2 \left( 1 + \frac{p_{u,v}^k |h_k|^2}{\Gamma(BER)d^\beta N_0} \right)} = \frac{N'_{u,v}}{\sum_{k=1}^{K} \log_2 \left( 1 + a_k p_{u,v}^k \right)}
$$

with

$$
N'_{u,v} = \frac{N_{u,v} T_b \log 2}{(1 - P_e)} \quad \text{and} \quad a_k = \frac{|h_k|^2}{\Gamma(BER)d^\beta N_0},
$$

where $h_k$ and $p_{u,v}^k$ denote the fading coefficient and the power transmitted over the $k$-th subchannel, respectively, and $p_{u,v} = [p_{u,v}^1, \ldots, p_{u,v}^K]^T$. Exploiting (15), the average energy cost $\bar{J}_{u,v}(p_{u,v})$, associated to the transfer of the program state $N_{u,v}$, is given by

$$
\bar{J}_{u,v}(p_{u,v}) = \bar{D}_{u,v}(p_{u,v}) \sum_{k=1}^{K} p_{u,v}^k = \frac{N'_{u,v} \sum_{k=1}^{K} p_{u,v}^k}{\sum_{k=1}^{K} \log_2 \left( 1 + a_k p_{u,v}^k \right)}
$$

Let us define $p = \{p_{u,v}\}_{(u,v) \in E} \in \mathbb{R}^{K \cdot \text{card}(E)}$. Using the expressions (15)-(17), the joint optimization of radio resources and code offloading, in the case of transmission over a wideband channel, can be written as:

$$
\min_{p} \sum_{v \in V} (1 - I_v) \cdot E_v^d + \sum_{(u,v) \in E} f_{u,v}(I_u, I_v, p_{u,v})
\text{s.t.} \sum_{v \in V} \left[ (1 - I_v) T_v^d + I_v T_v^p \right] + \sum_{(u,v) \in E} g_{u,v}(I_u, I_v, p_{u,v}) \leq L
$$

$$
I_v \in \{0, 1\}, \quad I_v = 0, \quad \forall v \in V_1,
$$

$$
\sum_{k=1}^{K} p_{u,v}^k \leq P_T, \quad \forall (u, v) \in E,
$$

where $f(I_u, I_v, p_{u,v})$ and $g(I_u, I_v, p_{u,v})$ are defined as in Table I (a) and (b), respectively, and $P_T$ denotes the power budget constraint on the sum of powers that the MUE can transmit over all the subchannels. As for $[P.1]$, the optimization problem in $[P.3]$ is a mixed integer problem. Then, let us
consider a generic combination \( c \in C \), where \( C \) is the set of all possible combinations of the binary variables \( I_v, v \in V \). For each combination \( c \), the value of the variables \( I_v \) is fixed to some value \( I^c_v \), and problem \([P.3]\) becomes the radio resource allocation problem

\[
\begin{align*}
\text{min} & \quad \sum_{(u,v)\in I_c} N'_{u,v} \sum_{k=1}^{K} p^k_{u,v} \\
\text{s.t.} & \quad \sum_{(u,v)\in I_c} N'_{u,v} \sum_{k=1}^{K} \log \left( 1 + a_k p^k_{u,v} \right) \leq L' \\
& \quad \sum_{k=1}^{K} p^k_{u,v} \leq P_T, \quad \forall (u, v) \in I_c,
\end{align*}
\]

where \( I_c \) is the set of edges \((u, v)\in E\) for which \( I^c_u = 0 \) and \( I^c_v = 1 \) given the combination \( c \). Furthermore, the optimization vector is \( \mathbf{p}_c = \{p_{u,v}\}_{(u,v)\in I_c} \), where \( \mathbf{p}_{u,v} = [p^1_{u,v}, \ldots, p^K_{u,v}]^T \). The delay constraint \( L' \) is the same as \((12)\). Again, the problem in \([P.4]\) is not convex. However, it can be proved the following result:

**Theorem 2:** If the following feasibility conditions are satisfied:

\[
L' > 0 \quad \text{and} \quad \sum_{k=1}^{K} \log(1 + a_k p^*_k) \geq \frac{\sum_{(u,v)\in I_c} N'_{u,v}}{L'} , \tag{18}
\]

where \( \mathbf{p}^* = \{p^*_k\}_{k=1}^{K} \) is the water-filling solution \((34)\) (see Appendix B), (a) problem \([P.4]\) admits a non-empty feasible set; (b) problem \([P.4]\) is equivalent to the problem \([P.6]\) (see Appendix B), which is such that any local optimum is also globally optimal.

**Proof:** See Appendix B.

To solve the problem in \([P.3]\), we propose an algorithm totally similar to the one we have proposed in the previous subsection. The only difference is that, for each combination \( c \), we check the feasibility condition \((18)\), instead of \((13)\), and we solve the optimization problem in \([P.4]\) (i.e., \([P.6]\)), instead of \([P.2]\) (i.e., \([P.5]\)).

**Remark 3:** To check the feasibility condition in \((18)\), the water-filling solution \((34)\) (see Appendix B) must be first computed. This solution is given by a simple iterative algorithm that converges to the optimal solution after \( K \) iterations. Since the complexity of the Water-Filling computation is very low, the feasibility condition in \((18)\) can be evaluated without heavily affecting the execution time of the overall algorithm.
III. Numerical Results

In this section we provide some numerical results to validate our theoretical findings and to assess the performance of the proposed algorithms. First, we show the effect of the call graph on the performance of the optimization algorithm. Second, we illustrate the role played by the random wireless channel, thus showing how the performance of the algorithm is affected by radio parameters such as, e.g., the distance and the number of independent channels. Finally, we apply the proposed method to the call graph of a face recognition application, in the case the MUE transmits over a set of parallel subbands.

Numerical Example 1 - Performance: The performance of the proposed offloading strategy is affected by several factors. The first important factor is of course the call graph topology and its parameters. As an example, let us consider three different examples of call graphs, as illustrated in Fig. 2. To grasp the dependence of the performance on the graph’s parameters, we consider for simplicity three directed call graphs having the same topology, but different parameters. We consider the case where the root node must be computed only locally. This represents the case where the node handles the interface with the user. All other nodes of the graphs can be all offloaded, if needed. All the nodes in the three graphs have the same set of energies $E^l_v$ (mJoule) and number of instructions $w_l$ (Mcycles), associated to the local computation of a method. What is different from one graph to the other is the number $N_{u,v}$ of KByte associated to the size of the program state at each edge of the graph, or the parameters $\varepsilon_{u,v}$ and $\gamma_{u,v}$, which are the energy and the time needed by the MUE to get the $N_{u,v}$ bits of the program state transmitted back by the SCceNB, respectively. In particular, Graph 2 has different $N_{u,v}$’s with respect to Graph 1, but same $\varepsilon_{u,v}$’s and $\gamma_{u,v}$’s, whereas Graph 3 has different $\varepsilon_{u,v}$’s and $\gamma_{u,v}$’s with respect to Graph 2, but same $N_{u,v}$’s. Let us indicate with 0 the index of the MUE, which acts equivalently as an end-node for the call graph. For simplicity and to grasp the main features, for graphs 1 and 2, we have $\varepsilon_{u,v} = \gamma_{u,v} = 0.05$ for all $(u,v) \in E$. The parameters of Graph 3 are chosen as before except that $\varepsilon_{u,v} = \gamma_{u,v} = 0.5$ for $(u,v) = \{(6,0),(7,0)\}$. This means that the output of the procedures 6 and 7, which must be transmitted back to the MUE if the procedures are computed remotely, is much larger than before. The colors associated to the nodes in Fig. 2 denotes the final result of the proposed joint optimization algorithm for the specific graph. In particular, the black nodes are computed locally, whereas the white nodes are offloaded to the server. In this simulation, we consider the presence of a single channel between MUE and SCceNB. For the joint optimization problem on $[P.1]$, the latency constraint $L$ is chosen equal to the time needed by the MUE to compute the entire program locally,
and the power budget constraint is $P_T = 0.01$ Watt. The local CPU clock speed is chosen as $f_l = 10^8$ cycles/s, whereas the server speed is equal to $f_s = 10^{10}$ cycles/s. The normalized channel coefficient $a$ in (9) is set equal to 500, and the bit duration $T_b$ is $10^{-6}$.

As we can notice from Fig. 2 slightly different call graphs (in terms of number of bits $N_{u,v}$, or

Fig. 2: Examples of optimal graph’s partitions.
parameters $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s) lead to completely different results. In the first case (Graph 1), all the nodes (except the root of course) are offloaded to the server. This happens because the overall energy that the MUE should have spent for computing the entire application locally is much greater than the energy needed to transmit the initial state of size $N_{12} = 500$ KByte. In this case, it might be that, for each individual method after node 2, its remote execution is more expensive than its local execution, and yet remote execution can save energy by offloading all the program. It is indeed important to remark that the result of the optimization is globally optimal (i.e., across the entire program) rather than locally optimal (i.e., relative to a single method invocation). Let us consider now the case of Graph 2 in Fig. 2 where the size $N_{12}$ and $N_{24}$ of the program state are larger than before. In this case, the optimal solution is the one shown in Fig. 2 (Graph 2), where only four methods are offloaded. This happens because, from an energetic point of view, it has become non convenient to transmit the large states $N_{12}$ or $N_{24}$ to the SCceNB. Furthermore, once a computation is offloaded, the result must be transmitted back to the SCceNB. If the result yields too many bits, it may not be convenient to offload that procedure. This is exactly what happen in the case of Graph 3 in Fig. 2. Indeed, increasing the values of the output size of procedures 6 and 7 (by acting on $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s) with respect to the case of Graph 2, the optimal solution changes as shown in Fig. 2 (Graph 3).

A fundamental factor affecting the performance of the joint optimization algorithm is the wireless fading channel between MUE and SCceNB. For this reason, we check the performance of the optimization algorithm as a function of the distance between MUE and SCceNB, considering a Multiple-Input-Multiple-Output (MIMO) transmission scheme. In Fig. 3 we report the average energy spent for processing versus the distance between MUE and SCceNB, for different MIMO configurations. The results are averaged over 200 independent realizations. In the presence of flat fading, the normalized channel coefficient can be written as

$$a = \frac{\alpha}{\Gamma(\text{BER})d^\beta N_0}$$

(19)

where $\alpha$ is a random variable, whose probability density function (pdf) depends on the MIMO communication strategy. Thus, assuming the presence of $M$ statistically independent spatial channels, the pdf of the coefficient $\alpha$ is given by

$$p_A(\alpha) = \frac{1}{(M-1)! \bar{\alpha}^M} \alpha^{M-1} e^{-\alpha/\bar{\alpha}} u(\alpha)$$

(20)

where $\bar{\alpha}$ is the variance over the single channel coefficient and $u(\alpha)$ denotes the unitary step function. In particular, in Fig. 3 we assume the use of SISO (i.e., $M = 1$), 1x2 SIMO (i.e., $M = 2$), and 2x2 MIMO (i.e., $M = 4$) communication strategies. The BER is chosen equal to $10^{-3}$, the path-loss coefficient $\beta$ is
Fig. 3: Average energy spent for processing versus distance (from MUE to SCceNB), for different communication strategies.

equal to 2, the noise power is given by $N_0 = 5 \times 10^{-5}$, and the variance $\bar{\alpha} = 1$. Further, we consider the Graph 1 in Fig. 2 as a call graph for this example. As expected, from Fig. 3, we notice how computation offloading is more convenient if the distance between MUE and SCceNB is sufficiently low, in order to allow the offloading of the entire program with high probability. This is a further numerical justification for favouring the access to the cloud through small cells. Further, it is possible to see how, by using MIMO strategies, which increase the probability to have a large normalized channel coefficient, we get a larger energy saving with respect to the SISO strategy, thanks to the increased offloading of instructions to the SCceNB.

**Numerical Example 2 - Application to a face recognition program:** We now apply our proposed joint optimization approach to the case of a realistic call graph of a program, representing a face recognition application [18]. The application’s call graph is shown in Fig. 4. The call graph has a tree structure, where the “user interface” procedure is the root node, and all the links are unidirectional such that the direction is from left to right. The “user interface” procedure can not of course be offloaded and must necessarily be evaluated locally. All the other methods can be offloaded. For this application, we illustrate an example of optimal power allocation and call graph’s partition, in the case the MUE is
transmitting by using multiple channels. For this purpose, in Fig. 5 we report the result of our joint optimization algorithm, which is applied to the call graph of the face recognition application in Fig. 4. We consider an OFDM system with $K = 8$ subcarriers. For the problem in [P.3], the latency constraint $L$ is chosen equal to the time needed by the MUE to compute the entire program locally, and the power budget constraint is $P_T = 0.02$ Watt. The local CPU clock speeds $f_I$ and $f_S$, and the bit duration $T_b$ are chosen as in previous simulations. The BER is chosen equal to $10^{-3}$, the path-loss coefficient $\beta$ is equal to 2, the noise power is given by $N_0 = 5 \times 10^{-5}$. The fading over the single carrier is given by the output of a random variable with pdf given by (20), with $M = 1$ and variance $\alpha = 1$. The resulting normalized wireless channels between MUE and SCceNB are shown in Fig. 5 (middle). The optimal graph’s partition for this parameter setting is shown in Fig. 5 (bottom), where, again, the white nodes denote procedures computed at the SCceNB side. As we can notice from Fig. 5 (bottom), all the remoteable nodes are offloaded to the SCceNB. This means that the optimization has found that the most convenient solution in terms of energy is to transmit the 182 KB of the program state between the “user interface” and the “FindMatch” procedures, and then compute all the rest of the program at the server. Thus, only one link in the call graph is selected to be used for the transmission of data. In particular, the top plot in Fig. 5 (top) shows the optimal power allocation over the multiple channels, achieved as a result of the optimization problem [P.4]. As we can see from Fig. 5 (top and middle), the power allocation shows a water-filling behavior, where all the power is concentrated over the best channels, while no bits are transmitted over the worse channels. The energy saving is potentially huge in this case. Indeed, by computing locally nodes 2, 3, and 4 of the graph in Fig. 4 the MUE would have spent 18.6 Joule, while, by offloading the

Fig. 4: Example of call graph of a face recognition application from [18].
entire program as in Fig. 5 (bottom), the MUE would spend only 25 mJoule. The gain in terms of energy saving is about 740 times. Since the size of the program states on the other links is much bigger than the one on the first link, it is clear that, for this application, there are only two possible optimal graph’s configurations: offload all the remoteable procedures or compute all the program locally. Indeed, if the normalized channels in Fig. 5 (middle) were uniformly worse, the optimization problem \[ P.4 \] might have been unfeasible, thus leading to the impossibility to offload the computation to the server. On the other side, even improving the channels, there is no possibility to improve the energy spent for processing by varying the graph’s partitioning.

IV. CONCLUSIONS

In this paper, we have proposed a method to optimize the allocation of communication resources and the call graph’s partition of a computer program jointly, in a mobile cloud computing context, with the aim of minimizing the energy consumption at the mobile user, while satisfying a delay constraint imposed by the application. The problem turns out to have a combinatorial complexity that increases with the granularity of the call graph structure. Nevertheless, we have proved that, for any given partition, the remaining optimization problem with respect to the radio parameters is convex and then it can be solved with
numerically efficient methods. Furthermore, with respect to the integer variables, the problem is a binary linear programming problem, for which several efficient algorithms exist, with affordable complexity. In general, the granularity of the call graph will be selected as a tradeoff between computational optimality and complexity. Simulation results corroborate our theoretical findings and illustrate for what kind of application and channel conditions, computation offloading can provide a significant performance gain with respect to local computation. In this work, we assumed the parameters of the call graph to be known. An interesting development of this work is the incorporation of learning mechanisms to tackle the case where these parameters are not known. A further development concerns the multiuser case, where a set of mobile users concur for the use of the same computational resources. Some preliminary results are reported in [53], but further extensions incorporating the optimal graph partitioning devised here in the multiuser care are currently investigated.

APPENDIX A

PROOF OF THEOREM 1

In the following, for simplicity of notation and without any loss of generality, we assume that the links \((u,v) \in I_c\) are identified by the index \(i = 1, \ldots, \text{card}(I_c)\).

To prove point (a), which ensures that the feasibility set is non-empty, we note that by inverting the relation

\[
\sum_{i=1}^{\text{card}(I_c)} \frac{N'_i}{\log(1 + aP_T)} \leq L',
\]

which is the first constraint in \([P.2]\) where instead of the \(p_i\)’s we have considered the maximum power value \(P_T\), we get the inequality (13), which is a sufficient condition ensuring the existence of a non-empty feasible set.

We proceed now in proving point (b). As said before, problem \([P.2]\) is not convex due to the concavity of the objective function with respect to the power allocation vector. However, let us consider the following change of variables

\[
t_i = \log(1 + ap_i) \Rightarrow p_i = \frac{e^{t_i} - 1}{a},
\]

\(\forall \ i = 1, \ldots, \text{card}(I_c)\). Relation (22) is a one to one mapping such that it is always possible to find uniquely the variable \(p_i\) from \(t_i\) and viceversa. Thus, using (22), the optimization problem in \([P.2]\)
becomes equivalent to the following problem (see [47, p. 130]):

\[
\begin{align*}
&P.5 \quad \min_{t_c} \quad \sum_{i=1}^{\text{card}(I_c)} \frac{N'_i e^{t_i} - 1}{a t_i} \\
&\text{s.t.} \quad \sum_{i=1}^{\text{card}(I_c)} \frac{N'_i}{t_i} \leq L' \\
&\quad 0 < t_i \leq T_{\max}, \quad i = 1, \ldots, \text{card}(I_c),
\end{align*}
\]

where \( t_c = [t_1, \ldots, t_{\text{card}(I_c)}]^{T} \), and \( T_{\max} = \log(1+aP_T) \). It is now straightforward to check the convexity of the optimization set, which is given by the intersection of convex sets. We now have to prove the strict convexity of the objective function. Let us define

\[
f(t_c) = \sum_{i=1}^{\text{card}(I_c)} N''_i \frac{e^{t_i} - 1}{t_i} \tag{23}
\]

where \( N''_i = N'_i/a > 0 \). Since the function in (23) is separable in the optimization variables, its Hessian is a diagonal matrix. Thus, to show the strict convexity of function (23), it is sufficient to prove that each diagonal element of the Hessian matrix is strictly positive inside the optimization set. Let us consider the \( i \)-th component. The second order partial derivative is given by

\[
\frac{\partial^2 f(t_c)}{\partial^2 t_i} = N''_i t_i^2 e^{t_i} - 2 e^{t_i} (t_i - 1) - 2 N''_i \frac{g(t_i)}{t_i^3} = N''_i \frac{g(t_i)}{t_i^3} - 2 \frac{g(t_i)}{t_i^3} \tag{24}
\]

where \( g(t_i) = t_i^2 e^{t_i} - 2 e^{t_i} (t_i - 1) = e^{t_i} (t_i - 1)^2 + e^{t_i} \). Since \( t_i \) and \( N''_i \) are positive, we have to analyze only the positiveness of the function \( g(t_i) - 2 \). It is easy to verify that

\[
\lim_{t_i \to 0^+} g(t_i) = 2, \tag{25}
\]

where the notation \( \lim_{t_i \to t_0^+} h(t) \) means the right limit of function \( h(t) \) around the point \( t_0 \). Then, if we prove that \( g(t_i) \) is monotonic increasing for \( t_i > 0 \), we have showed that the second derivative in (24) is strictly positive. In particular, we have

\[
\frac{dg(t_i)}{dt_i} = e^{t_i} [(t_i - 1)^2 + 1] + 2 (t_i - 1) e^{t_i} = t_i^2 e^{t_i} > 0, \tag{26}
\]

\( \forall t_i > 0 \). Since the previous arguments hold true for all \( i \in [1, \ldots, \text{card}(I_c)] \), we can conclude that the Hessian is positive definite and the objective function (23) is strictly convex, thus concluding the proof of point (b).

To prove point (c), we note that, since problem [P.5] is convex, its optimal solutions must satisfy the KKT conditions [47]. Furthermore, as proved in (a), since the feasible set is not empty under (13), the
Slater’s condition is verified and strong duality holds [47]. Let us consider the following Lagrangian function for the problem \([P.5]\):

\[
\mathcal{L}(t, \lambda, \mu, \nu) = \sum_{i=1}^{\text{card}(I)} \frac{N'_i e^{t_i} - 1}{t_i} + \lambda \left( \sum_{i=1}^{\text{card}(I)} \frac{N'_i}{t_i} - L' \right) + \sum_{i=1}^{\text{card}(I)} \mu_i (t_i - T_{\text{max}}) - \sum_{i=1}^{\text{card}(I)} \nu_i t_i
\]

where \(\lambda, \mu = [\mu_1, \ldots, \mu_{\text{card}(I)}]^T\), and \(\nu = [\nu_1, \ldots, \nu_{\text{card}(I)}]^T\) are the Lagrange multipliers associated to the constraints. Then, the KKT conditions for problem \([P.5]\) read as:

\[
\nabla_{t_i} \mathcal{L}(t, \lambda, \mu, \nu) = 0,
\]

(27)

(28)

(29)

(30)

Considering the \(i\)-th component of (27), we have

\[
\frac{N'_i e^{t_i} (t_i - 1) + 1}{t_i^2} - \lambda \frac{N'_i}{t_i^2} + \mu_i - \nu_i = 0
\]

(31)

From (31), since because of (30) we have \(\nu_i = 0 \ \forall i\), we get

\[
\lambda = \frac{\mu_i + N'_i e^{t_i} (t_i - 1) + 1}{N'_i t_i^2} > 0, \quad \forall t_i > 0.
\]

(32)

The previous sentence holds true because \(\mu_i \geq 0, \ \forall i\), from (29), and \(e^{t_i} (t_i - 1) + 1 > 0\) for all \(t_i > 0\). As a consequence, because of (32) and (28), the latency constraint must be satisfied with strict equality, thus concluding also the proof of point (c).

APPENDIX B

PROOF OF THEOREM 2

To prove point (a), which ensures that it is possible to transmit data with a delay less equal than the maximum tolerable latency, let us consider the two constraints of problem \([P.4]\). For simplicity of notation let us assume that the links \((u, v) \in I_c\) are identified by the index \(i = 1, \ldots, \text{card}(I_c)\). To get a sufficient condition, which guarantees that the delay constraint in \([P.4]\) is satisfied, we must find a lower bound of the first constraint with respect to all the possible power allocations \(\{p^k_i\}\), subject to the presence of the power budget constraints that compose the second constraint. This can be achieved...
by maximizing the denominator of the first constraint, for all $i$, subject to the presence of the second constraint in [P.4]. Thus, since the power budget constraints in [P.4] are the same for all $i$, the feasibility condition is given by (18), where $p^* = \{p^*_k\}_{k=1}^K$ is the solution of the following problem:

$$
\max_{\mathbf{p}} \; \sum_{k=1}^K \log \left( 1 + a_k p^*_k \right) \quad \text{s.t.} \quad \sum_{k=1}^K p^*_k = P_T. \tag{33}
$$

The solution of (33) is the well known Water Filling [49], which allocates power as:

$$
p^*_k = \left[ \mu - \frac{1}{a_k} \right]^+, \tag{34}
$$

where $[x]^+ = \max(0, x)$, $\mu = 1/\lambda$ is the water level, and $\lambda$ is the Lagrange multiplier associated to the constraint in (33). The value of the water level $\mu$ is found by imposing the power budget constraint in (33) to the optimal solution (34). Given $p^* = \{p^*_k\}_{k=1}^K$ by (34), it is clear that the inequality (18) holds, thus giving a sufficient condition on the channel values ensuring the existence of a non-empty feasible set.

To prove point (b), we exploit arguments that are similar to what we have used in Appendix A. In particular, let us consider the following change of variables:

$$
t^k_i = \log(1 + a_k p^*_k) \rightarrow p^*_k = \left( e^{t^k_i} - 1 \right) / a_k, \tag{35}
$$

$\forall \; i = 1, \ldots, \text{card}(I_e), \; \forall \; k = 1, \ldots, K$. Relation (35) is a one to one mapping such that it is always possible to find uniquely the variable $p^*_k$ from $t^k_i$ and vice versa. Thus, using (35), the optimization problem in [P.4] becomes equivalent to the following problem:

$$
[P.6] \quad \min_{\mathbf{t}_c} \; \sum_{i=1}^{\text{card}(I_e)} \frac{N'_i \sum_{k=1}^K e^{t^k_i} - 1}{\sum_{k=1}^K t^k_i} \quad \text{s.t.} \quad \sum_{i=1}^{\text{card}(I_e)} \frac{N'_i \sum_{k=1}^K t^k_i}{K} \leq L',
$$

$$
\sum_{k=1}^K \frac{e^{t^k_i} - 1}{a_k} \leq P_T, \quad i = 1, \ldots, \text{card}(I_e)
$$

where $\mathbf{t}_c = [t_1, \ldots, t_{\text{card}(I_e)}]^T$ and $t_i = [t^1_i, \ldots, t^K_i]$. We first prove the convexity of the set. It is straightforward to see that all the constraints grouped in the second constraint in [P.6] are convex.
Further, let us define

$$g(t_c) = \sum_{i=1}^{\text{card}(I_c)} \frac{N_i'}{K} = \sum_{i=1}^{\text{card}(I_c)} g_i(t_i). \tag{36}$$

If we prove that all the functions $g_i(t_i)$ are convex, the overall sum function $g(t_c)$ is convex. We proceed by showing that $g_i(t_i)$ is log-convex. Indeed, it is known how log convexity of a function ensures also its convexity [47]. Then, we have

$$\log g_i(t_i) = \log N_i' - \log \left( \sum_{k=1}^{K} t_i^k \right). \tag{37}$$

It is now easy to prove that the Hessian of function (37) is given by $H_{g_i}(t_i) = \frac{1}{K} \times \frac{1}{K}$, where $1_{K \times K}$ denotes a squared matrix of size $K$ composed of all ones. Matrix $H_{g_i}(t_i)$ has $K-1$ zero eigenvalues and one strictly positive eigenvalue, $\forall t_i$ in the optimization set. Thus, the functions $g_i(t_i)$ are log-convex, and then $g(t_c)$ is convex. The optimization set is then convex since it is given by the intersection of convex sets. Consider now the objective function of [P.6], which reads as:

$$f(t_c) = \sum_{i=1}^{\text{card}(I_c)} w_i(t_i) = \sum_{i=1}^{\text{card}(I_c)} \frac{N_i'}{K} \sum_{k=1}^{K} \frac{e_{t_i}^k - 1}{a_k} \sum_{k=1}^{K} t_i^k \tag{38}$$

where $w_i(t_i) = \frac{N_i'}{K} \sum_{k=1}^{K} \frac{e_{t_i}^k - 1}{a_k} \sum_{k=1}^{K} t_i^k$. Before to proceed we have to introduce some definitions on generalized convexity which are instrumental to prove that any stationary point is a globally optimal solution. A differentiable function $f(t_c)$ is pseudoconvex at a point $t_c^0$ [50], [51], if

$$(t_c - t_c^0)^T \nabla f(t_c^0) \geq 0 \implies f(t_c) \geq f(t_c^0) \tag{39}$$

for any $t_c \in \text{dom } f$. In [50][pag. 142](see also [52][pag. 147]) it is proved that, if $f$ is pseudoconvex at $t_c^0$, then having $\nabla f(t_c^0) = 0$ implies that $t_c^0$ is a globally optimal point. Hence, if $f$ is pseudoconvex at every stationary point, then every stationary point is a global minimum. To exploit this property, we need to prove that $f(t_c)$ is pseudoconvex at every stationary point. More specifically, consider $f(t_c) = \sum_{i=1}^{\text{card}(I_c)} w_i(t_i) = \sum_{i=1}^{\text{card}(I_c)} \frac{h(t_i)}{v(t_i)}$ where $h(t_i) = N_i' \sum_{k=1}^{K} \frac{e_{t_i}^k - 1}{a_k}$ and $v(t_i) = \sum_{k=1}^{K} t_i^k$. Let us now show that
each function $w_i(t_i)$ is pseudoconvex $\forall t_i \in \text{dom } w_i$. Assume that $(t_i - t_i^0)^T \nabla w_i(t_i^0) \geq 0$, i.e.

$$(t_i - t_i^0)^T \left( \frac{\nabla h(t_i^0)}{v(t_i^0)} - \frac{h(t_i^0)}{v^2(t_i^0)} \nabla v(t_i^0) \right) \geq 0.$$  \hspace{1cm} (40)$$

Since $h(t_i)$ is a differentiable convex function and $v(t_i)$ is linear, the following first-order conditions \([47]\) hold for all $t_i^0, t_i \in \text{dom } w_i$

$$h(t_i) \geq h(t_i^0) + \nabla^T h(t_i^0)(t_i - t_i^0)$$

$$v(t_i) = v(t_i^0) + \nabla^T v(t_i^0)(t_i - t_i^0).$$

Hence, by using these inequalities in (40) and since $v(t_i) > 0$, it results

$$\left( \frac{h(t_i)}{v(t_i^0)} - \frac{h(t_i^0)}{v^2(t_i^0)} v(t_i) \right) \geq 0$$

i.e.

$$w_i(t_i) = \frac{h(t_i)}{v(t_i^0)} \geq \frac{h(t_i^0)}{v(t_i^0)} = w_i(t_i^0)$$

for any $t_i, t_i^0 \in \text{dom } w_i$. Then, from \([39]\), we can state that $w_i(t_i)$ is a pseudoconvex function for every $i = 1, \ldots, \text{card}(I_c)$. Let us now prove that $f(t_c) = \sum_{i=1}^{\text{card}(I_c)} w_i(t_i)$ is a pseudoconvex function at every stationary point $t_c^0$. Hence assume that $(t_c - t_c^0)^T \nabla f(t_c^0) \geq 0$, i.e.

$$\sum_{i=1}^{\text{card}(I_c)} (t_i - t_i^0)^T \nabla w_i(t_i^0) \geq 0$$

where $t_c^0 = [t_1^0, \ldots, t_{\text{card}(I_c)}^0]$ is a stationary point of $f(t_c)$. Let us denote with $S = \{t_c \in \text{dom } f : \nabla f(t_c^0) = 0\}$ the set of the stationary points of $f(t_c)$. Observe that since the functions $w_i(t_i)$ are uncoupled, the set $S$ is equal to the cartesian product of the sets $S_i$ of the stationary points of each function $w_i(t_i)$, i.e. $S = S_1 \times S_2 \times \ldots \times S_{\text{card}(I_c)}$. Hence from the pseudoconvexity of each function $w_i(t_i)$ we can state that

$$(t_i - t_i^0)^T \nabla w_i(t_i^0) = 0 \Rightarrow w_i(t_i) \geq w_i(t_i^0) \quad \forall t_i \in \text{dom } w_i$$

so that

$$\sum_{i=1}^{\text{card}(I_c)} w_i(t_i) \geq \sum_{i=1}^{\text{card}(I_c)} w_i(t_i^0).$$

Hence, at each stationary point $t_c^0$, we get

$$(t_c - t_c^0)^T \nabla f(t_c^0) = 0 \Rightarrow f(t_c) \geq f(t_c^0)$$

for any $t_c \in \text{dom } f$. Then, from \([39]\), $f$ is pseudoconvex at every stationary point, thus ensuring that every stationary point of $f$ is a global minimum. This completes the proof of point (b).
REFERENCES

[1] J. Kincaid, “YouTube Mobile Uploads Up 400 since iPhone 3GS Launch,” http://www.techcrunch.com/2009/06/25/youtube-mobile-uploads-up-400-since-iphone-3gs-launch/, 2009.

[2] J. Lewin, “iPhone Users 30 Times More Likely To Watch YouTube Videos,” http://www.podcastingnews.com/2008/03/19/iphone-users-30-times-watch-youtube-videos/, 2008.

[3] R. A. Powers, “Batteries for low power electronics,” Proceedings of the IEEE, vol.83, pp. 687-693, April 1995.

[4] M. R. Palacin, “Recent advances in rechargeable battery materials: a chemists perspective,” Chemical Society Reviews, no. 38, pp. 2565-2575, 2009.

[5] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, “A Survey of Computation Offloading for Mobile Systems”, Mobile Networks and Applications, pp. 129–140, Feb. 2013.

[6] B. Hayes B, “Cloud computing,” Communications of the ACM, vol. 51, no. 7, pp. 9-11, July 2008.

[7] Z. Li, C. Wang C, R. Xu R, “Computation offloading to save energy on handheld devices: a partition scheme”, Proc. International conference on compilers, architecture, and synthesis for embedded systems, pp. 238-246, Atlanta, Georgia, USA, 2001.

[8] C. Wang, Z. Li, “Parametric analysis for adaptive computation offloading”, Proc. SIGPLAN conference on programming language design and implementation, pp. 119-130, Washington, DC, June 2004.

[9] R.K. Balan, “Powerful change part 2: reducing the power demands of mobile devices”, IEEE Pervasive Computing mag., vol. 3, no. 2, pp. 71-73, 2004.

[10] G. Chen, B.T. Kang, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, R. Chandramouli, “Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices,” IEEE Transactions on Parallel and Distributed Systems, vol 15, no. 9, pp. 795-809, 2004.

[11] S. Ou, K. Yang, A. Liotta, L. Lu, “Performance analysis of offloading systems in mobile wireless environments,” Proc. IEEE international conference on communications, pp. 1821-1806, Glasgow, June 2007.

[12] C. Xian, Y-H Lu, Z. Li, “Adaptive computation offloading for energy conservation on battery-powered systems,” International conference on parallel and distributed systems, pp. 1-8, Hsinchu, Dec. 2007.

[13] G. Huerta-Canepa, D. Lee, “An adaptable application offloading scheme based on application behavior,” International conference on advanced information networking and applications, pp 387-392, Okinawa, March 2008.

[14] R. Wolksi, S. Gurun, C. Krintz, D. Nurmi, “Using bandwidth data to make computation offloading decisions”, Proc. IEEE symposium on parallel and distributed processing, pp. 1-8, Boston, Apr. 2008.

[15] K. Yang, S. Ou, H-H. Chen, “On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications,” IEEE communications magazine, vol. 46, no.1, pp. 56-63, 2008.

[16] Y. J. Hong, K. Kumar, Y. H. Lu, “Energy efficient content-based image retrieval for mobile systems,” Proc. of the International symposium on circuits and systems, pp. 1673-1676, Taipei, 24-27 May 2009.

[17] B.-G. Chun, P. Maniatis, “Augmented Smartphone Applications Through Clone Cloud Execution,” Proc. of the HotOS, Monte Verita, Switzerland, May 2009.

[18] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “MAUI: making smartphones last longer with code offload,” Proc. of the ACM International Conference on Mobile Systems, Applications, and Services, pp.49-62, San Francisco, CA, USA, 15-18 June 2010.

[19] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, “CloneCloud: Boosting Mobile Device Applications Through Cloud Clone Execution,” available on Arxiv at http://arxiv.org/abs/1009.3088
[20] Y. Nimmagadda, K. Kumar K, Y. H. Lu, C. S. G. Lee, “Realtime moving object recognition and tracking using computation offloading,” Proc. IEEE international conference on intelligent robots and systems, pp 2449-2455, San Francisco, Oct. 2010.

[21] K. Kumar, Y.-H. Lu, “Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?,” Computer, pp. 51–56, April 2010.

[22] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, “ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading”, Proc. of INFOCOM 2012, pp. 945–953, March 2012.

[23] D. Kovacev, T. Yu, R. Klamma, “Adaptive Computation Offloading from Mobile Devices into the Cloud”, Proc. of IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 784–791, July 2012.

[24] B. Gao, L. He, L. Liu, K. Li; S. A. Jarvis, “From Mobiles to Clouds: Developing Energy-Aware Offloading Strategies for Workflows”, Proc. of ACM/IEEE 13th Int. Conf. on Grid Computing (GRID), pp. 139–146, Sept. 2012.

[25] Y. Wen, W. Zhang, H. Luo, “Energy-Optimal Mobile Application Execution: Taming Resource-Poor Mobile Devices with Cloud Clones”, Proc. of INFOCOM 2012, pp. 2716–2720, March 2012.

[26] D. Huang, P. Wang, D. Niyato, “A Dynamic Offloading Algorithm for Mobile Computing”, IEEE Trans. on Wireless Communications, pp. 1991–1995, June 2012.

[27] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, “To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing”, Proc. of INFOCOM 2013, April 2013, Turin, Italy.

[28] M. Goyal, A. Kumar, and V. Sharma, “Power constrained and delay optimal policies for scheduling transmission over a fading channel,” Proc. of IEEE Infocom, vol. 1, pp. 311-320, Mar/Apr 2003.

[29] E. Uysal-Biyikoglu, B. Prabhakar, and A. El Gamal, “Energy-efficient packet transmission over a wireless link,” IEEE/ACM Transactions on Networking, vol. 10, no. 4, pp. 487-499, Aug 2002.

[30] A. El Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, “Energy-efficient scheduling of packet transmissions over wireless networks,” Proceedings of IEEE Infocom, vol. 3, pp. 773-1782, New York, USA, Nov 2002.

[31] A. Fu, E. Modiano, and J. Tsitsiklis, “Optimal Energy Allocation for Delay-Constrained Data Transmission over a Time-Varying Channel,” IEEE Conf. on Computer Communications, vol. 2, pp. 1095–1105, San Francisco, USA, 2003.

[32] M. A. Zafer and E. Modiano, “Delay-constrained energy efficient data transmission over a wireless fading channel,” Information Theory and Applications Workshop, pp. 289-298, Lake Tahoe, USA, Jan/Feb 2007.

[33] R. A. Berry and R. G. Gallager, “Communication over fading channels with delay constraints,” IEEE Transactions on Information Theory, vol. 48, no. 5, pp. 1135-1149, May 2002.

[34] Bettesh and S. Shamai, “Optimal power and rate control for minimal average delay: The single-user case,” IEEE Transactions on Information Theory, vol. 52, no. 9, pp. 4115-4141, Sep 2006.

[35] Y. Liang, R. Zhang, J. Cioffi, “Transmit Optimization for MIMO-OFDM With Delay-Constrained and No-Delay Constrained Traffic,” IEEE Transactions On Signal Processing, vol. 54, no. 8, pp. 3190-3199, Aug. 2006.

[36] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, “The Case for VM-Based Cloudlets in Mobile Computing”, IEEE Pervasive Computing, pp. 14–23, Oct.-Dec. 2009.

[37] TROPIC: Distributed computing, storage and radio resource allocation over cooperative femtocells, http://www.ict-tropic.eu.

[38] B. G. Ryder, “Constructing the Call Graph of a Program,” IEEE Transactions on Software Engineering, vol. SE-5, no. 3, pp. 216-226, May 1979.

[39] D. Callahan, A. Carle, M.W. Hall, K. Kennedy, “Constructing the procedure call multigraph,” IEEE Transactions on Software Engineering, vol.16, no.4, pp.483-487, Apr. 1990.
[40] D. Grove, G. DeFouw, J. Dean, C. Chambers, “Call graph construction in object-oriented languages,” *SIGPLAN Notices* vol. 32, no. 10, pp. 108-124, Oct. 1997.

[41] D. Grove, C. Chambers, “A framework for call graph construction algorithms,” *ACM Transactions on Programming Languages and Systems*, vol. 23, no. 6, pp. 685-746, Nov. 2001.

[42] S. Cui, A. J. Goldsmith, A. Bahai “Energy-constrained modulation optimization,” *IEEE Transactions On Wireless Communications*, vol. 4, no.5, pp. 2349-2360, Sept. 2005.

[43] R. S. Prabhu, B. Daneshrad, L. Vandenberghe, “Energy minimization of a QAM system,” *IEEE Conference on Wireless Communications and Networking*, pp. 729-734, Kowloon, March 2007.

[44] R. S. Prabhu, and B. Daneshrad, “Energy minimization of a QAM system with fading,” *IEEE Transactions on Wireless Communications*, vol. 7, no.12, pp.4837-4842, Dec. 2008

[45] G. L. Nemhauser, L. A. Wolsey, *Integer and combinatorial optimization*, Wiley, 1988.

[46] D. Fudenberg, J. Tirole, *Game Theory*, MIT Press, 1991.

[47] S. P. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2003.

[48] D. P. Bertsekas, *Convex Analysis and Optimization*, Athena Scientific, Belmont, Massachusetts, 2003.

[49] J. G. Proakis, *Digital Communication Systems*, Fourth edition, McGraw-Hill, New York, 2001.

[50] Olvi L. Mangasarian, *Nonlinear Programming*. SIAM, Philadelphia, 1994.

[51] C. Olsson, F. Kahl, “Generalized Convexity in Multiple View Geometry,” *Jour. of Math. Imaging Vision*, Sept. 2010.

[52] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming, Theory and Algorithms*. Wiley-Interscience, 1993.

[53] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, “Joint allocation of computation and communication resources in multiuser mobile cloud computing.” *Proc. of IEEE Workshop on Signal Processing Advances in Wireless Comm.*, Darmstadt 2013.