Sergio Barbarossa, Stefania Sardellitti, and Paolo Di Lorenzo

Communicating While Computing

Distributed mobile cloud computing over 5G heterogeneous networks

Current estimates of mobile data traffic in the years to come foresee a 1,000 × increase of mobile data traffic in 2020 with respect to 2010, or, equivalently, a doubling of mobile data traffic every year. This unprecedented growth demands a significant increase of wireless network capacity. Even if the current evolution of fourth-generation (4G) systems and, in particular, the advancements of the long-term evolution (LTE) standardization process foresees a significant capacity improvement with respect to third-generation (3G) systems, the European Telecommunications Standards Institute (ETSI) has established a roadmap toward the fifth-generation (5G) system, with the aim of deploying a commercial system by the year 2020 [1]. The European Project named "Mobile and Wireless Communications Enablers for the 2020 Information Society" (METIS), launched in 2012, represents one of the first international and large-scale research projects on fifth generation (5G) [2]. In parallel with this unparalleled growth of data traffic, our everyday life experience shows an increasing habit to run a plethora of applications specifically devised for mobile devices, (smartphones, tablets, laptops)for entertainment, health care, business, social networking, traveling, news, etc. However, the spectacular growth in wireless traffic generated by this lifestyle is not matched with a parallel improvement on mobile handsets' batteries, whose lifetime is not improving at the same pace [3]. This determines a widening gap between the energy required to run sophisticated applications and the energy available on the mobile handset. A possible way to overcome this obstacle is to enable the mobile devices, whenever possible and convenient, to offload their most energy-consuming tasks to nearby fixed servers. This strategy has been studied for a long time and is reported in the literature under different names, such as *cyberforaging* [4] or *computation offloading* [5], [6]. In recent years, a strong impulse to computation offloading has come through cloud computing (CC), which enables the users to utilize resources on demand. The resources made available by a cloud service provider are: 1) infrastructures, such as network devices, storage, servers, etc., 2) platforms, such as operating systems, offering an integrated environment for developing and testing custom applications, and 3) software, in the form of application programs. These three kinds of services are labeled, respectively, as *infrastructure as a service*, *platform as a service*, and *software as a service*. In particular, one of the key features of CC is virtualization, which makes it possible 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 (VMs) can scale on demand, thus improving the overall system computational

©ISTOCKPHOTO.COM/ZONADEARTE

Digital Object Identifier 10.1109/MSP.2014.2334709
Date of publication: 15 October 2014

efficiency. Mobile CC (MCC) is a specific case of CC where the user accesses the cloud services through a mobile handset [5]. The major limitations of today's MCC are the energy consumption associated to the radio access and the latency experienced in reaching the cloud provider through a wide area network (WAN). Mobile users located at the edge of macrocellular networks are particularly disadvantaged in terms of power consumption and, furthermore, it is very difficult to control latency over a WAN. As pointed out in [7]–[9], humans are acutely sensitive to delay and jitter: as latency increases, interactive response suffers. Since the interaction times foreseen in 5G systems, in particular in the so-called tactile Internet [10], are quite small (in the order of milliseconds), a strict latency control must be somehow incorporated in near future MCC. Meeting this constraint requires a deep rethinking of the overall service chain, from the physical layer up to virtualization.

Within this framework, the goal of this article is to review first a series of offloading mechanisms and then to provide a mathematical formulation of the computation offloading problem aimed at optimizing the communication and computation resources jointly, posing a strict attention to latency and energy constraints. Wherever possible, we try to emphasize those features of 5G systems that can help meet the strict latency constraints while keeping the energy consumption at a minimum level. Signal processing can play a significant role in mobile cloud computing from different perspectives: from a rigorous mathematical formulation and efficient solution of the resource allocation problem to the development of applications specifically built to take full advantage of computation offloading, etc.

COMPUTATION OFFLOADING FOR MOBILE CLOUD COMPUTING

MCC can bring the following advantages:

- prolong battery lifetime, by offloading energy-consuming tasks from the mobile handset to the cloud
- enable mobile devices to run sophisticated applications and provide significantly higher data storage capabilities
- improve reliability, since data can be stored and backed up from the mobile device to a set of reliable fixed devices specifically designed for storage purposes.

These advantages come on top of the usual advantages of CC, specifically making resources, either storage or applications, available on demand without the need for the user to own sophisticated devices or software tools.

Offloading strategies may be classified in different ways, depending on what aspects are identified as most relevant. From the point of view of the protocols used to handle the exchange of data between mobile device and server, three classes are identified [5]: client-server communication, virtualization, and mobile agents. (A mobile agent is a program able to migrate across a network carrying its own code and execution state.) The client–server protocol requires the services to be preinstalled in the participating devices. Examples of this class are: Spectra [11], Chroma [12], and Cuckoo [13]. The second class of methods requires the instantiation of VMs on the server. Virtualization ensures a relatively secure execution as a VM encapsulates and separates the guest software from the host software. Examples of methods based on virtualization are Mobile Assistance Using Infrastructure (MAUI) [7], CloneCloud [14], and MobiCloud [15]. Finally, methods based on mobile agents use a mobile approach to partition and distribute jobs and are more suitable for disconnected operations typical of wireless access. An example of this class is Scavenger [16].

Offloading a computation does not necessarily imply transferring all the program execution to the remote server. Typically, a program is first subdivided into modules, some of which need to be run on the mobile handset, such as, for instance, all modules controlling the input/output peripherals. For the rest of the modules, a decision has to be taken on what is more appropriate to offload. Offloading can be either static or dynamic. Static offloading means that the program partitioning is given before execution, and the decision about what modules to transfer is taken, once for all, at the beginning of the execution. Examples of static offloading are Spectra [11], [17], [18] and Chroma [12]. In contrast, in dynamic offloading, the decision on whether and what to offload is taken at run-time based on current conditions. Dynamic offloading is, in principle, more efficient than static offloading; however, it induces more overhead on the system relating to latency, profiling, and run-time decision making. Examples of dynamic offloading are [19]–[22].

Offloading typically requires code partitioning [7], [23], [24], 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. MAUI [7] is an example of an offloading method aimed at selecting what program modules to offload to minimize energy consumption at the mobile terminal. The approach is based on the so-called call graph representation of a program. A call graph is a representation that models the relations between the modules (procedures) of a computer program in the form of a directed graph $\mathcal{G} = (V, E)$, where each vertex $v \in V$ represents a procedure in the call stack, and each directed edge $e = (u, v)$ represents the invocation of procedure v from procedure u. The call graph includes also auxiliary information concerning, for instance, the number of instructions within each module and the amount of data exchanged among modules. For nonrecursive languages with reasonable assumptions on the program structure [25], the call graph is a directed, acyclic graph. Given the call graph of the application, MAUI collects information about energy consumption and data transfer requirements and solves an integer linear program to determine which modules are more suitable for being offloaded. In this way, MAUI does not offload the whole application, but only the most energy-consuming modules. The critical aspect is the prediction of energy consumption. MAUI saves information about past offloaded methods and uses online profiling to create an energy consumption model. When new offloading requests are received, MAUI uses history data to predict the execution

time of the task. Of course, the effectiveness of this offloading scheme depends on the accuracy achievable in the prediction of energy consumption and time execution. ThinkAir is an alternative strategy supporting method-level offloading to a smartphone clone executing in the cloud [26]. ThinkAir uses three profilers at the mobile side, monitoring: the energy consumption of the device hardware; the program parameters, such as execution time, acquired memory, number of instructions; and communication related parameters, such as bandwidth, connectivity, and delay.

Besides computational aspects, there are two major issues about offloading associated with radio access: power consumption and latency. These are indeed two of the major bottlenecks in the deployment of an effective MCC in current cellular networks. In macrocellular systems, the power spent from mobile users, especially those located in the edge of the cell, may be significant. In some cases, this large transmit power may nullify all potential benefits in terms of energy saving. A possible way to reduce this power consumption is to bring computational resources closer to the mobile user. This idea was put forward in [8], where the concept of a cloudlet was introduced. In such a case, the mobile handset offloads its workload to a local cloudlet consisting of a set of multicore computers connected to the remote cloud server. The storage and computational capabilities of the cloudlet are much smaller than those available at the cloud server, but, at the same time, installing a cloudlet is much less expensive than installing a cloud server. The main advantage of this solution is scalability—the powerful cloud resources are used only when really necessary, otherwise computation is offloaded to a cloudlet. The radio access to the cloudlet could be through Wi-Fi. The idea of bringing cloud services closer to the mobile users has been further pushed in the current European Union project named "Distributed Computing, Storage, and Radio Resource Allocation over Cooperative Femtocells" (TROPIC) [30], where it was proposed to endow small-cell base stations with additional, albeit limited, cloud functionalities. The new base stations are denoted, in LTE terminology, small-cell cloud enhanced e Node B (SCceNB). In this way, a mobile user is able to find a radio access point within a short distance, enabling it to access cloud functionalities. The scenario is depicted in Figure 1, where the SCceNB's are interconnected through the so-called femtoclouds, i.e., small clouds with intermediate storage and computation capabilities. The femtoclouds manage the allocation of VMs to the users accessing through the associated base stations. The femtoclouds are then interconnected with each other and to the cloud provider. Whenever the users' request can be met by the local femtocloud, everything is performed locally. Otherwise, the SCceNB may ask the intervention of the cloud server through high-capacity wired links. In this way, both radio and computational resources are brought closer to the user, thus improving scalability in both radio and computation aspects. The radio access is based on LTE, which yields some advantages over Wi-Fi: 1) it provides a single technology solution for offloading, with no need to switch from 3G/4G to Wi-Fi and vice versa, and 2) it provides QoS guarantees.

Bringing resources closer to the user improves not only power consumption at the terminal side but also the other major issue, latency. More specifically, the latency ℓ contains three terms:

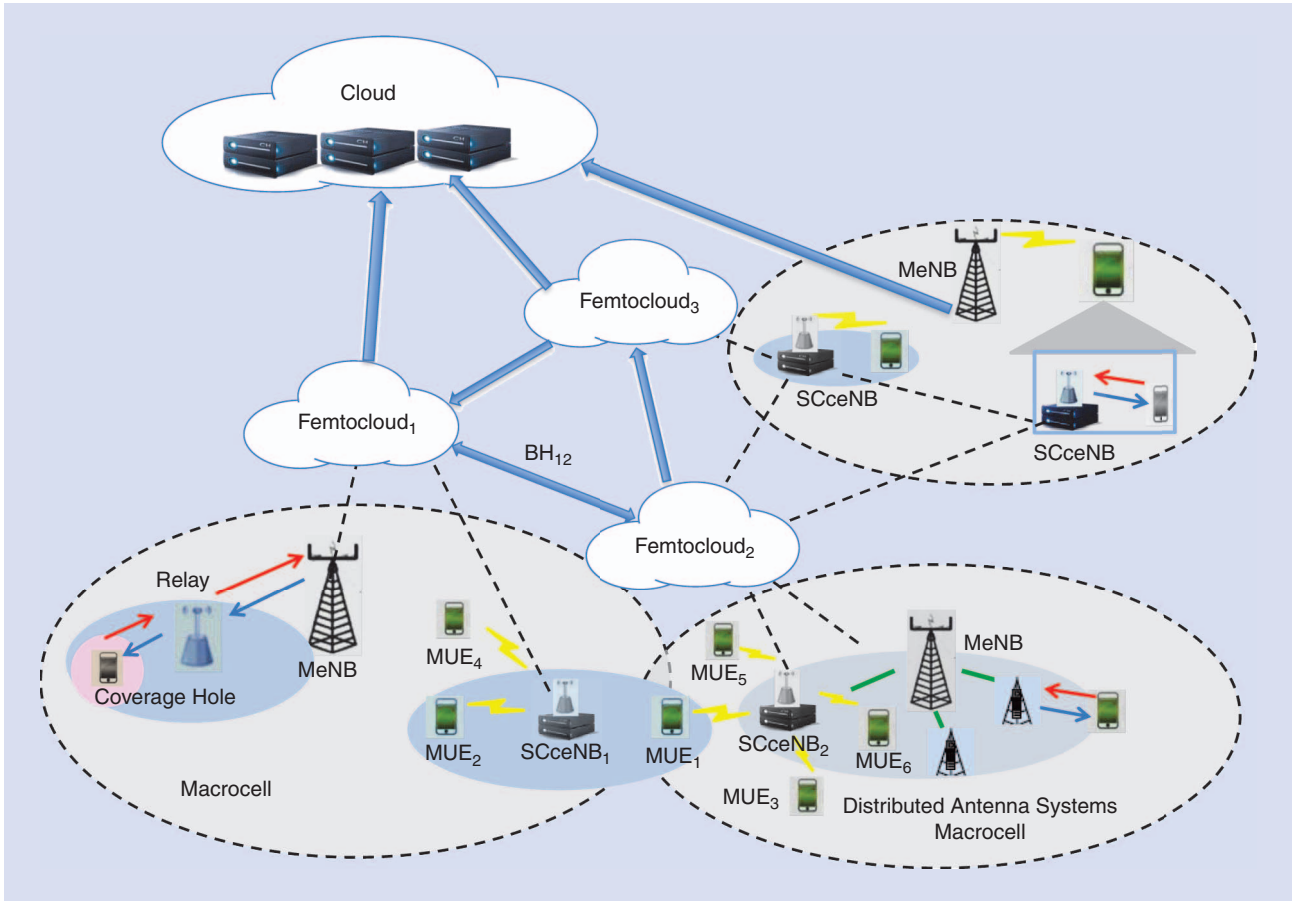
$$\ell = \Delta_T + \Delta_{\text{exe}}^{\text{rem}} + \Delta_R, \tag{1}$$

where Δ_T is the time needed to send the information necessary to transfer the program execution from the mobile device to the cloud, $\Delta_{\text{exe}}^{\text{rem}}$ is the time necessary to run the program at the remote side (cloud), and Δ_R is the time necessary to the cloud to send the result back to the mobile unit. More specifically, the time Δ_T includes the time for the mobile unit to reach the radio access point plus the time for this information to travel from the access point to the cloud server through the backhaul. This second term depends on the technology involved in the backhaul: the latency over a fiber optic cable may be negligible, but the latency over an asymmetric digital subscriber line (ADSL) link may not, depending on traffic. The term $\Delta_{\text{exe}}^{\text{rem}}$ depends on the computational load and on how many computational resources [e.g., VM, central processing unit (CPU) cycles, etc.] are assigned to the user. This depends on the server capabilities, but also on the number of users asking for the service. Equation (1), in its simplicity, shows how, in MCC, the issues related to communication and computation are strictly related to each other. In fact, the effectiveness of an offloading scheme depends on both the radio access and computational aspects. Very schematically, every application, or part of it, is characterized by an input, a number of instructions to be executed, and an output. To offload a computation, it is necessary to transfer the program state (input and state variables) from the mobile user to the server (cloud). Clearly, the applications more suitable for offloading are those characterized by a limited input (or state) size and a high number of instructions to be executed. An example of this class of applications is a chess game. On the contrary, offloading may not be convenient for those applications where it is necessary to transmit a large set of data and the computational load is not so heavy. These qualitative statements will be corroborated by quantitative evaluations in the ensuing sections. But before delving into the mathematical formulation, it is worth outlining how the 5G revolution can have an impact on MCC.

MOBILE CLOUD COMPUTING IN 5G SYSTEMS

Even if 5G is still undefined, some key features have already been identified; see, e.g., [1]. A nonexhaustive list of some of the key features, as relevant to computation offloading, is reported below:

- (*Ultra*)*dense cell deployment:* The deployment of small-cell networks is already part of 4G evolution, but network "densification" is going to play a major role in 5G systems [29]. Small-cell networks include femtocell networks, specifically devised to cover indoor environments or outdoor cells served by small base stations placed on lamp posts or on the facades of a building. Small-cell networks, covering areas with a radius in the order of a few tens of meters, are going to coexist with conventional macrocell networks. With respect to MCC, a dense deployment of small-cell base stations carries two advantages: 1) it reduces the transmit power necessary for



[FIG1] The distributed cloud scenario.

computation offloading and the latency over the wireless access channel and 2) it increases the probability for the mobile handset to find an access point within a short range; furthermore, if some small-cell base stations are endowed with additional cloud functionalities, scalability improves along both radio and computational resources.

■ *Millimeter-wave links:* The usage of wideband links, with very high capacity and directivity, provides an effective way for the radio access points to forward the users' offloading requests to the cloud with reduced latency, thus overcoming the limitations of today's ADSL backhaul links, often used in femtocell networks; furthermore, these high-capacity links facilitate cooperation among small-cell base stations at both radio and computing levels.

■ *Massive multiple-input, multiple-output (MIMO):* MIMO transceivers improve spectral efficiency, thus reducing the time necessary to transfer the program execution from the mobile site to the cloud; furthermore, the usage of extensive beamforming allows an efficient management of intercell interference through adaptive null steering.

■ *Multicell cooperation:* Since computation offloading involves both communication and computation aspects, the cooperation among cells is fundamental to distribute radio access and computing requests in the most effective way; in particular, cooperation may occur at the radio level to reduce interference and, at the application level to implement a distributed cloud capability [28].

■ *Cognitive radio:* The incorporation of cognitive radio capabilities helps to improve the overall system efficiency; the specific novelty brought by cognitive techniques in the MCC context is that the cognition activity can involve both radio aspects as well as learning and adaptation of appropriate energy consumption models, which play a basic role in devising the most appropriate offloading strategy across a network of computing resources; furthermore, exploiting the cooperation among base stations, it is possible to implement collaborative sensing techniques to augment the learning capabilities of the individual access points, in terms of channel sensing and energy consumption models.

■ *Quality of experience (QoE) versus quality of service (QoS):* A design driven by a user's QoE implies a system approach that does not consider radio or networking aspects separated from the application requirements; in this sense, this change of perspective matches perfectly with the synergic approach that is going to be most effective for MCC.

OPTIMAL ALLOCATION OF RADIO RESOURCES IN A SINGLE-USER SCENARIO

For simplicity, we start with a single user case before moving to the multiser/multicell scenario. We consider alternative optimization criteria to have a set of strategies to tackle alternative application requirements and user needs.

MINIMUM TRANSMIT ENERGY UNDER COMPUTATIONAL CONSTRAINT

The first criterion we analyze is the minimization of the energy consumption at the mobile side under a computational rate constraint, besides the usual power budget constraint. We assume here that the computation has to take place within a time window of T seconds. The mobile decides to carry out the computations locally or to offload computations to the cloud depending on which strategy requires less energy consumption at the mobile side. In case of offloading, the user needs to send all necessary data (input, program state, etc.) to the cloud. This involves a time τ. The other system parameters are: f_{loc} is the local computational rate (CPU cycles/s), f_S is the server computational rate (CPU cycles/s), B is the bandwidth of the link from MUE to SCceNB, p_{proc} is the power spent for local processing, N is the dimension (number of bits) of the program state, and λ is the computational rate (CPU cycles/s) required to run the application while meeting the user's requests.

The degrees of freedom are the following: $d \in \{0,1\}$ is the decision variable, set to 0 if the program is executed at the mobile side, or 1 otherwise; $p \in [0, P_T]$ is the power spent for transmitting the program state from mobile to server; $\tau \in [0, T]$ is the duration of the interval necessary for transmitting the program state to the server, necessary to enable the program execution transfer.

Offloading takes place if the energy $p\tau$ consumed by the terminal device for offloading is less than the energy $p_{\text{proc}}T$ needed for carrying out the computations locally. In case of offloading, the effective computing rate, taking into account the time necessary to transfer the program execution to the cloud, is $\bar{f} = (1 - \tau/T) f_S$. The decision variable d is explicitly related to p and τ through

$$d = u(p_{\text{proc}}T - p\tau), \tag{2}$$

where $u(\cdot)$ is the unit step function. Assuming adaptive modulation, the time τ necessary to send the N bits encoding the program state across a channel of bandwidth B is related to p as

$$\tau = \frac{N}{B\log(1+\alpha p)} \tag{3}$$

with $\alpha = |h|^2/\Gamma(\text{BER})\sigma_n^2$, where h is the channel coefficient, $\Gamma(\text{BER}) = -(2\log(5\,\text{BER})/3)$ is the signal-to-noise ration (SNR) margin introduced to meet a target BER, and σ_n^2 is the receiver noise power. Note that the gap factor $\Gamma(\text{BER})$ is valid under the assumption $\Gamma(\text{BER}) > 0$, i.e., BER $< 1/5$. Exploiting the relations among the free variables p, τ, and d, the objective function can be expressed in terms of the single variable τ and the optimization problem can be formulated as:

$$\min_{\tau} \min\left[p_{\text{proc}}T, \frac{\tau}{\alpha}(2^{N/B\tau} - 1)\right] \tag{4}$$

subject to (s.t.)

$$C.1\ \frac{N}{B\log(1+\alpha P_T)} \le \tau \le T$$

$$C.2\ f_{\text{loc}} + [f_S(1-\tau/T) - f_{\text{loc}}] \cdot u(p_{\text{proc}}T - p\tau) \ge \lambda.$$

After some algebraic manipulations, it is possible to show that this problem is feasible if $f_S \ge \lambda$ and the (equivalent) channel coefficient α exceeds a minimum value, i.e., $\alpha > \alpha_{\min}$, where

$$\alpha_{\min} = \max\left[\frac{1}{P_T}, \frac{(1-\frac{\lambda}{f_S})}{p_{\text{proc}}}\right](2^{\frac{N}{BT(1-\lambda/f_S)}} - 1). \tag{5}$$

This last condition states, in closed form, that offloading can take place only if the channel is sufficiently good, as expected. The interesting point is that the minimum channel value is dictated by parameters related to both radio and computational parameters. The previous problem is convex and, if the feasible set is nonempty, the optimal transmit power can be expressed in closed form as

$$p = \frac{1}{\alpha}(2^{\frac{N}{BT(1-\lambda/f_S)}} - 1). \tag{6}$$

Clearly, since the wireless channel is random, there is a nonnull probability that offloading takes place or not, depending on both channel status and computational requests. The consequence is that the real computing rate experienced over a fading channel is going to depend on the channel statistics and typically will be lower than the rate achievable under ideal channel conditions. Interestingly, building on previous expressions, we can derive the equivalent computing rate λ^*, to be asked by the mobile device to ensure the desired rate λ, in the presence of fading, provided that the fading statistics are known. More specifically, the probability of offloading is simply

$$\mathcal{P} := \text{Prob}\{\alpha > \alpha_{\min}\} = 1 - D_A(\alpha_{\min}), \tag{7}$$

where $D_A(\alpha)$ denotes the cumulative distribution function of α and $\alpha_{\min}$ is given in (5), with λ^* instead of λ. The expected value of the rate is then

$$f_{\text{ave}} = (1 - \mathcal{P}(\lambda^*))f_{\text{loc}} + \mathcal{P}(\lambda^*)\lambda^*, \tag{8}$$

where we made explicit the dependence of $\mathcal{P}$ from λ^* (through $\alpha_{\min}$). Imposing this average rate to be equal to the target rate λ, we end up with a nonlinear equation in λ^*. This equation admits a unique solution. As a numerical example, Figure 2(a) shows λ^* as a function of the distance between mobile user and base station for different antenna configurations. The parameters of the simulation are $f_S = 10^{10}$, $f_{\text{loc}} = 10^7$, $p_{\text{proc}} = 0.1$, $P_T = 0.1$, $N = 5 \times 10^3$, $B = 2 \times 10^6$, $T = 10^{-2}$, and $\lambda = 10^8$. The channels are generated as statistically independent Rayleigh fading channels. As expected, at short distances, λ^* tends to coincide

with λ, because channel attenuation is negligible and offloading occurs most of the times. However, as the distance exceeds a certain value, λ^* starts increasing considerably to compensate for the missing opportunities to offload. In parallel, Figure 2(b) shows the average energy spent for offloading as a function of the distance between mobile user and base station, for different antenna configurations. As expected, the energy increases at larger distance until reaching a constant value dictated by the energy required to run the application locally. It is interesting to see how MIMO transceivers yield a larger saving and then, ultimately, widen the area over which offloading is beneficial.

MINIMUM TRANSMIT POWER UNDER DELAY CONSTRAINT

In this section, we assume all radio equipment to be equipped with multiple antennas and the action available at the mobile user to select the precoding matrix to satisfy some optimality criterion for offloading. We use the following symbols: n_T and n_R are the number of antennas at the transmit and receive sides, respectively; L is the maximum latency limit; P_T is the maximum transmit power budget; w is the number of CPU cycles to be executed; N is the number of bits to be transmitted in case of offloading; $\mathcal{E}_{\text{loc}}$ is the energy spent to run the program at the mobile side; $\mathbf{Q}$ is the covariance matrix of the transmitted symbols; $\mathbf{H}$ is the $n_R \times n_T$ channel matrix between mobile user and base station; and $\mathbf{R}_n$ is the disturbance (interference plus noise) covariance matrix.

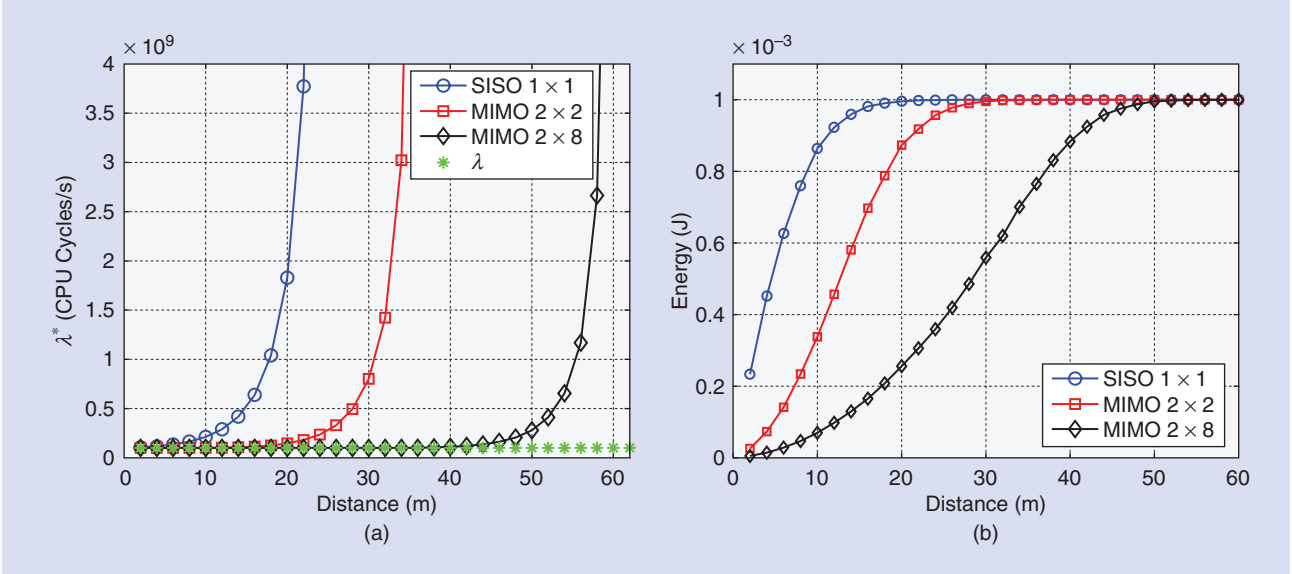
The mobile user offloads its computations if the energy spent for offloading is less than the energy necessary for processing the data locally. The goal of the optimization is then to find the optimal precoding matrix (equivalently, the covariance matrix of the transmitted symbols) to minimize power consumption, s.t. the following constraints: 1) latency constraint, 2) energy for offloading less than energy to be spent for local processing, and 3) transmit power less than available power budget. The problem can be formulated as:

$$\min_{\mathbf{Q}\succeq 0} \ \operatorname{trace}(\mathbf{Q}) \quad \text{s.t.} \quad \left.\begin{aligned} &\text{i)} \ \frac{N}{B\log_2\det(\mathbf{I}+\mathbf{H}\mathbf{Q}\mathbf{H}^H\mathbf{R}_n^{-1})}+\frac{w}{f_S}+\Delta_R \le L \\ &\text{ii)} \ \frac{\operatorname{tr}(\mathbf{Q})N}{B\log_2\det(\mathbf{I}+\mathbf{H}\mathbf{Q}\mathbf{H}^H\mathbf{R}_n^{-1})} \le \mathcal{E}_{\text{loc}} \\ &\text{iii)} \ \operatorname{tr}(\mathbf{Q}) \le P_T, \quad \mathbf{Q}\succeq 0 \end{aligned}\right\} \triangleq \mathcal{X} \qquad (\mathcal{P}.1) \tag{9}$$

where the three line constraints reflect the constraint list mentioned above; $N/B\log_2\det(\mathbf{I}+\mathbf{H}\mathbf{Q}\mathbf{H}^H\mathbf{R}_n^{-1})$ is the time necessary to transmit N bits over a bandwidth B using optimal coding. The symbol $\mathcal{X}$ denotes the feasible set: If $\mathcal{X}$ is empty, offloading is not convenient or impossible to carry out within the user's requirements and processing is performed at the mobile device; if $\mathcal{X}$ is nonempty, the previous problem is convex, offloading takes place and the optimal precoding matrix can be expressed in closed form, as proved in [31]. In particular, denoting with $\mathbf{H}^H\mathbf{R}_n^{-1}\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{U}^H$ the eigendecomposition of the composite channel matrix, weighted with the inverse of the disturbance covariance matrix $\mathbf{R}_n$, the optimal covariance matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \mathbf{U}(\alpha\mathbf{I} - \mathbf{D}^{-1})^+\mathbf{U}^H, \tag{10}$$

where $\alpha = (\beta\mathcal{E}_{\text{loc}} + \lambda(L - w/f_S - \Delta_R))/(\mu + \beta c + 1)$ is a positive constant, a function of the three Lagrangian multipliers β, λ, and μ is associated with the three constraints in (9) with $c = N/B$. Interestingly, the solution (10) has the well known "water-filling"



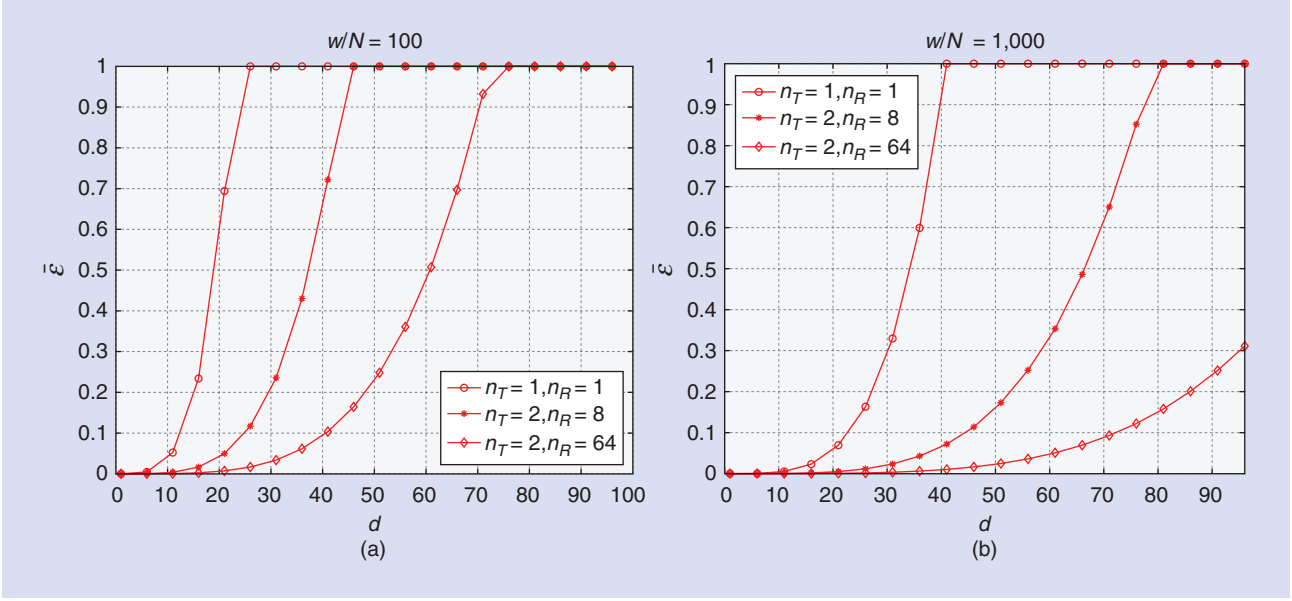
[FIG2] (a) The equivalent rate λ^* versus distance between mobile user and access point for different communication strategies. (b) The average energy spent for offloading versus distance between mobile user and access point for different communication strategies.

form but with a water level depending on parameters specifying the computational features, e.g., a number of CPU cycles, size of program state, energy necessary for local processing. As a numerical example, in Figure 3, we report the average energy spent with or without offloading (normalized to the energy to be spent locally) as a function of the distance between a mobile user and access point for different MIMO configurations and different applications. The average has been evaluated over 100 independent realizations of Rayleigh fading channels. The difference between the applications is captured by the ratio w/N between the number w of CPU cycles to be executed to run the program and the number of bits N to be transmitted to transfer the program execution. The maximum level in each figure is the energy spent for local processing at the mobile side (all energy values are normalized with respect to this value, so that it is easier to read offloading gains in percentage). From Figure 3, we can observe how offloading is advantageous at short distances and, depending on the distance, it can also bring substantial savings. Furthermore, it is also evident how MIMO transceivers enlarge the area over which offloading is advantageous. Comparing (a) and (b), we can observe that, as expected, offloading is more advantageous for program modules characterized by a higher ratio w/N, i.e., a higher computational load (number of CPU cycles), for a number of bits to be exchanged. These curves are simple examples of how MCC can benefit from a dense deployment of base stations and from massive MIMO, as foreseen in 5G, because having proximity access facilitates offloading and then enables a higher energy saving at the mobile terminal.

JOINT OPTIMIZATION OF PROGRAM PARTITIONING AND RADIO RESOURCE ALLOCATION

So far, we have considered a single module to offload. In this section, we consider a more structured program, represented as a call graph, as in MAUI, and we illustrate an approach that optimizes code partitioning and radio resource allocation jointly. By code partitioning, we mean the decision about which modules are to be offloaded and which ones are to be executed locally. The difference with respect to MAUI is that, whereas in MAUI the energy consumption and latencies are supposed to be given (or estimated), here we optimize jointly across code partitioning and radio resource allocation. In our computation offloading framework, we label each vertex $v \in V$ of the call graph with the energy E_v^l it takes to execute the procedure locally, and with the overall number of instructions w_v (CPU cycles), of which the procedure is composed. 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, such as the program modules controlling the user interface or the interaction with input/output devices. The set of procedures that must be executed locally is denoted by V_l. 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 vice versa. The objective is to minimize the energy consumption at the mobile site, under power budget and latency constraints.

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. To incorporate the fact that the program initiates and terminates at the mobile site, we introduce



[FIG3] The average energy spent for offloading versus distance between mobile device and radio access point for different antenna configurations and applications: (a) $w/N = 100$ and (b) $w/N = 1{,}000$.

two auxiliary vertices, namely the initial and terminating vertices, whose indicator variables are set to zero by default. The addition of these two nodes gives rise to an extended edge set $\mathcal{E}_e$ that comprises all the edges of the original call graph, plus the edges from the initiating node to the call graph, and the edges from the call graph to the terminating node. We also denote by $p_{u,v}$ the power spent to transmit the program state between the procedures u and v. To decide which modules of the call graph should be executed remotely, we need to solve the following optimization problem [27]:

$$(\mathcal{P}.2) \quad \min_{I,p} \sum_{v\in V}(1-I_v)\cdot E_v^l + \sum_{(u,v)\in\mathcal{E}_e}[J_{u,v}(p_{u,v})I_v + \varepsilon_{u,v}I_u - (J_{u,v}(p_{u,v})+\varepsilon_{u,v})I_uI_v] \tag{11}$$

s.t.

$$\sum_{v\in V}[(1-I_v)T_v^l + I_vT_v^r] + \sum_{(u,v)\in\mathcal{E}_e}[D_{u,v}(p_{u,v})I_v + \gamma_{u,v}I_u - (D_{u,v}(p_{u,v})+\gamma_{u,v})I_uI_v] \le L \tag{12}$$

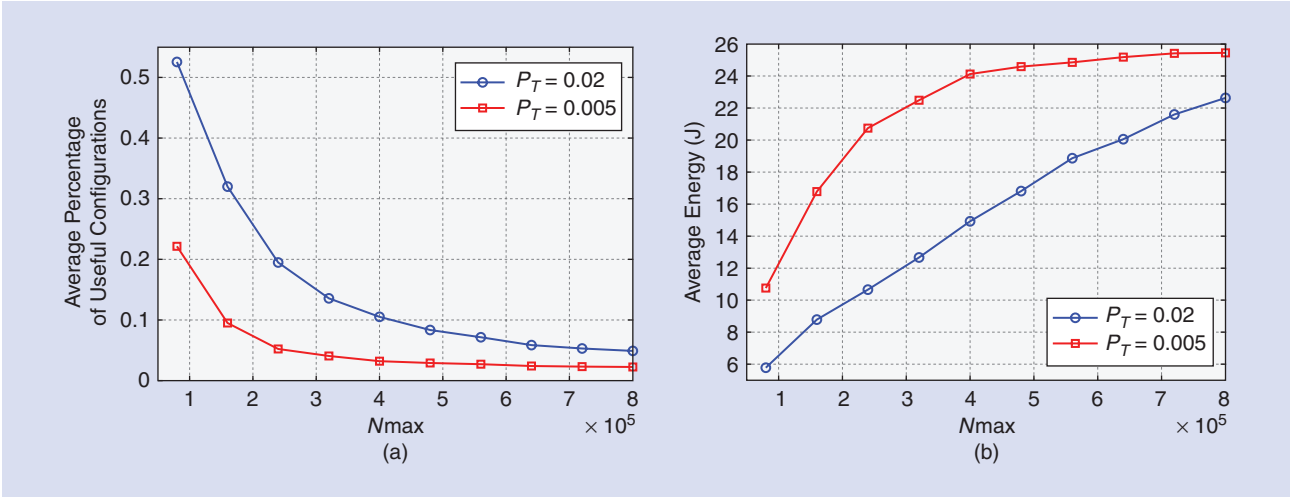
$$I_v\in\{0,1\},\ I_v=0,\ \forall v\in V_l,\ 0\le p_{u,v}\le P_T\ \forall(u,v)\in\mathcal{E}_e, \tag{13}$$

where $I=\{I_v\}_{v\in V}$, $p=\{p_{u,v}\}_{(u,v)\in\mathcal{E}}\in\mathbb{R}^{\mathrm{card}(\mathcal{E}_e)}$, with $\mathrm{card}(\mathcal{E}_e)$ denoting the cardinality of set $\mathcal{E}_e$, T_v^l, and T_v^r are the time it takes to execute the program module v locally or remotely, respectively. The objective function in (11) represents the total energy spent by the MUE for executing the application. In particular, the first term in (11) 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 extended call graph) of the energies spent to transfer the execution from the MUE to the SCceNB. The quantity $J_{u,v}(p_{u,v})$ represents the energy necessary to transmit the $N_{u,v}$ bits encoding the program state from the MUE to the SCceNB, whereas $\varepsilon_{u,v}$ is the cost for 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 specific form of the objective function in (11) has been derived so that there is an energy cost associated to offloading only if the 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 by the SCceNB. Now, assuming an adaptive modulation scheme that selects the QAM constellation size as a function of channel conditions and computational requirements, 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 as in (3), with $p=p_{u,v}$. The energy $J_{u,v}(p_{u,v})$, associated to the transfer of a program state of size $N_{u,v}$, is simply $J_{u,v}(p_{u,v})=p_{u,v}D_{u,v}(p_{u,v})$. The constraint in (12) is a latency constraint, and it contains two summations: the first summation includes the time to run the local modules plus the time to run the offloaded modules; the second summation is the overall 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 by the application. The quantity $\gamma_{u,v}$ is the time needed by the MUE to decode the $N_{u,v}$ bits of the program state transmitted back by the SCceNB. From (12), 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 $D_{u,v}(p_{u,v})$, whereas, if $I_u=1$ and $I_v=0$, the delay is equal to $\gamma_{u,v}$. The constraint in (13) 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 (13), is the power budget constraint on the maximum transmit power P_T.

Clearly, this optimization procedure is rather complex. Since the state variables I_u are integer, problem $(\mathcal{P}.2)$ is inherently a mixed nonlinear integer programming problem, which might be very complicated to solve. However, a series of simplifications are possible to reduce the complexity of the overall algorithm. An important simplification comes from observing that, for any set of integer values I_v, the remaining optimization over the power coefficients is a convex problem [27]. Furthermore, it is possible to derive closed-form expressions that allow us to check the feasibility of the convex optimization problem for any fixed graph configuration. This feasibility check enables us to discard a priori all graph configurations that cannot be offloaded, for any transmission power satisfying the budget constraint P_T in $(\mathcal{P}.2)$. This check may considerably reduce the complexity of the search. These statements are corroborated by the numerical results reported in Figure 4(a), where we report the average number of call graph configurations worth of offloading (the ones passing the feasibility check) and the total energy consumption (including the energy spent for local processing and the energy spent for offloading) versus the maximum size $N\max$ of the program state to be transferred. The results have been averaged over 1,000 call graph realizations, where the graph has six modules and the number of bits to be transferred for each module are generated as a uniform random variable in the interval $[0, N\max]$. From Figure 4(a), we can see how, increasing the transmit power P_T, there are more feasible configurations because offloading is more likely to occur. At the same time, Figure 4(b) shows that the energy consumption is smaller for the higher transmit power, because offloading occurs more frequently (and then less energy is consumed for local processing). This curve shows an interesting tradeoff between energy consumption and complexity.

OPTIMAL ALLOCATION OF COMMUNICATION/ COMPUTATION RESOURCES IN A MULTIUSER SCENARIO

We consider now the more challenging scenario composed of a set of N_c clouds, N_b small-cell access points (eNBs), and K mobile users. The goal is to find the optimal strategy to assign each user to a base station and to a cloud, to minimize the overall energy



[FIG4] **(a) The average percentage of useful call graph configurations versus maximum program state size for different mobile transmit power. (b) The average energy spent for offloading versus maximum program state size for different mobile transmit power.**

consumption, under latency constraints. The degrees of freedom are the precoding matrix $\mathbf{Q}_k \in \mathbb{C}^{n_T \times n_T}$ for each user, assuming MIMO transceivers with n_T transmit antennas, the number f_{mk} of CPU cycles/second for running the application of user k over the mth cloud, and the assignment of each user to a base station and then to a cloud. The optimal assignment is performed by selecting the binary values $a_{nmk} \in \{0,1\}$ for $n = 1,\ldots,N_b$, $m = 1,\ldots,N_c$, $k = 1,\ldots,K$. For each k, $a_{nmk} = 1$ if user k accesses the network through the nth BS and it is then served by the mth cloud; all other values are set to zero. In principle, a user could be served by multiple base stations, as in cooperative communications, and by multiple clouds. However, this scenario would make the overall computation management much more complicated. The objective is the minimization of a weighted sum of the energies spent by each mobile terminal: $\mathcal{E}_{\text{tot}} := \sum_{k=1}^{K} c_k \mathcal{E}_k(\mathbf{Q}, \mathbf{a}_k)$, with $\mathcal{E}_k(\mathbf{Q}, \mathbf{a}_k) = \text{trace}(\mathbf{Q}_k) \sum_{n=1}^{N_b} \sum_{m=1}^{N_c} a_{nmk} \Delta_{nk}^t(\mathbf{Q})$, where $\mathbf{Q} \triangleq (\mathbf{Q}_k)_{k=1}^K$ is the set of all covariance matrices. The coefficients c_k are positive parameters that could be varied dynamically to enforce some sort of fairness among the users. The overall latency experienced by the kth MUE for accessing the network through the base station n and being served by the cloud m is now

$$\Delta_{nmk} = \Delta_{nk}^t + \frac{w_k}{f_{mk}} + T_{rx}^{nmk} + T_{Bmn}, \tag{14}$$

where Δ_{nk}^t is the time needed to send the information necessary to transfer the program execution from the kth MUE to the nth base stations, w_k/f_{mk} is the time necessary to execute w_k CPU cycles at the mth server, and T_{rx}^{nmk} is the time necessary for the server to send the results back to the kth MUE. The new term with respect to (1) is the delay T_{Bmn} over the backhaul used to transfer the program state from the nth base station to the mth cloud. The transmission delay Δ_{nk}^t is

$$\Delta_{nk}^t(\mathbf{Q}) = \frac{N_k}{B \log_2 \det\left(\mathbf{I} + \mathbf{H}_k^n \mathbf{Q}_k \mathbf{H}_k^{nH} \tilde{\mathbf{R}}_{nk}(\mathbf{Q}_{-k})^{-1}\right)}, \tag{15}$$

where $\mathbf{H}_l^n$ is the channel matrix between the lth user and the nth base station and the covariance matrix

$$\tilde{\mathbf{R}}_{nk}(\mathbf{Q}_{-k}) = N_0 \mathbf{I} + \sum_{l=1, l\neq k}^{K} \mathbf{H}_l^n \mathbf{Q}_l \mathbf{H}_l^{nH}$$

now contains noise plus multiuser interference.

Formally, the optimization problem can be formulated as:

$$\begin{aligned}
&\min_{\mathbf{Q},\mathbf{f},\mathbf{a}} \sum_{k=1}^{K} c_k \mathcal{E}_k(\mathbf{Q}, \mathbf{a}_k), \text{ subject to:} \\
&\text{i) } a_{nmk}\left(\frac{N_k}{B\log_2 \det\left(\mathbf{I}+\mathbf{H}_k^n \mathbf{Q}_k \mathbf{H}_k^{nH} \tilde{\mathbf{R}}_{nk}(\mathbf{Q}_{-k})^{-1}\right)} + \frac{w_k}{f_{mk}} + T_{Bmn}\right) \\
&\qquad \le L_k, \quad \forall k, n, m \\
&\text{ii) } \text{tr}(\mathbf{Q}_k) \le P_T, \quad \mathbf{Q} \succeq 0, \forall k = 1,\ldots,K, \\
&\text{iii) } f_{mk} \ge 0, \forall m,k \quad \sum_{k=1}^{K} \sum_{n=1}^{N_b} a_{nmk} f_{mk} \le F_m \quad \forall m = 1,\ldots,N_c \\
&\text{iv) } \sum_{n=1}^{N_b} \sum_{m=1}^{N_c} a_{nmk} = 1, a_{nmk} \in \{0,1\}, \quad \forall k, n, m,
\end{aligned} \tag{$\mathcal{P}$.3}$$

(16)

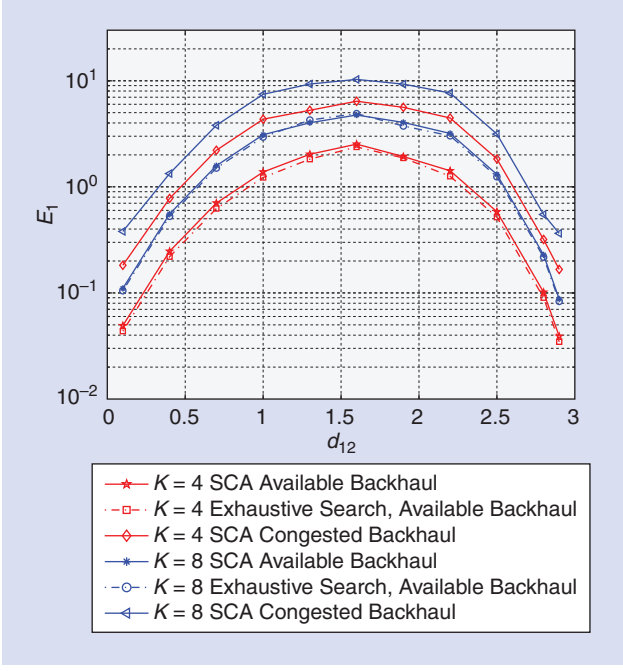
where the constraints are: 1) the overall latency for each user k must be lower than the maximum tolerable value L_k, 2) the total power spent by each user must be lower than its total power budget, 3) the sum of the computational rates assigned by each server cannot exceed the server computational capability F_m, and 4) each mobile user should be served by one couple base station-cloud. To drive the solution toward the situation where each user is served by a single base station and a single cloud, we enforce the constraint $\sum_{n=1}^{N_b} \sum_{m=1}^{N_c} a_{nmk} = 1$ for each k. For simplicity, we have incorporated the term T_{rx}^{nmk} in the latency limit L_k.

MOBILITY MANAGEMENT

As a particular case of the above formulation, we can handle user mobility as follows. Consider, with reference to Figure 1, a single user case, i.e., $K = 1$, where a user (MUE 1) moves from a

position near SCceNB 1 to a position close to SCceNB 2. While moving, a conventional cellular system would perform a base station handover to associate that user to the best station, i.e., the station providing the higher SNR. However, if we consider the allocation of radio and computing resources jointly, we need to consider also a cloud handover. This means that if we switch the radio access from SCceNB 1 to SCceNB 2 but we keep the VM running over SCceNB 1, to avoid VM migration, we need to take into account the delay for sending data over the backhaul from SCceNB 2 to SCceNB 1. Alternatively, we might consider switching both radio access point and serving cloud, but in such a case, we need to migrate the VM. The solution of this problem can be achieved as a particular case of problem $\mathcal{P}.3$, with $K = 1$.

Unfortunately, problem $(\mathcal{P}.3)$ is inherently combinatorial and then NP hard. This makes its exact solution hard to achieve even for moderate values of N_c, N_b, and K. To overcome this obstacle, we assume the coefficients a_{nmk} to be real variables belonging to the interval $[0, 1]$. Then, we adopt a successive convex approximation (SCA) approach to solve problem $(\mathcal{P}.3)$ as a sequence of strongly convex problems converging to a local solution of the original problem (see, e.g., [31] for details). As an example, we consider the scenario sketched in Figure 1, where MUE 1 moves from SCceNB 1 toward SCceNB 2. While moving, the optimal assignment and resource allocation is periodically recomputed as a solution of problem $(\mathcal{P}.3)$. The optimization involves $N_b = 4$ base stations, $N_c = 4$ clouds and $K = 4$ or 8 mobile users. In Figure 5 we compare the average energy consumption obtained by solving the relaxed form of problem $(\mathcal{P}.3)$ in the case where the backhaul between the clouds is congested or only lightly loaded. We also report, as a benchmark, the optimal results achievable with the exhaustive search. It is remarkable to see how our relaxed algorithm gets very close to the optimal exhaustive search solution. Also, we can see the advantage resulting from having a good backhaul between the two base stations.



[FIG5] Energy consumption versus distance d_{12}.

CONCLUSIONS AND FURTHER DEVELOPMENTS

In this article, we have proposed a system perspective of next 5G systems centered on the need to empower energy-hungry mobile terminals with computation offloading capabilities via proximity radio access through small-cell base stations endowed with cloud functionalities. We showed how the optimal resource allocation involves a joint allocation of radio and computation resources, within a fully cross-layer approach. The proposed solution has an impact on several aspects related to the new radio interface, signaling strategies and overall network management. First of all, in case of pervasive computation offloading, the traffic statistics over the uplink and downlink channels are going to change significantly with respect to the current situation, where there is a clear preponderance of traffic on the downlink channel. This change will call for a different (possibly dynamic) partitioning between the capacity associated to the uplink and downlink channels. Furthermore, since the proposed distributed cloud approach requires a (possibly) intensive exchange of data among base stations (in case of base station/cloud handover), the traffic over the wired backhaul linking the base station may become a critical issue. This problem call for the inclusion of high-capacity wireless links between small-cell base stations, e.g., using millimeter-wave direct links, already considered for 5G. We showed how computations can be distributed among a pool of clouds, but assuming only one cloud is active at each time. In principle, the approach can be extended to the parallel computing case, but this would raise extra complexity issues. We also considered a centralized solution. A distributed solution could be reached through the usual primal/dual decomposition methods. Of course this will require a limited exchange of data among the nodes concurring to reach the solution. The wide cross-layer approach proposed in this article of course does not come without a price—the need for higher signaling among applications and physical layers. However the results can justify this extra complexity. Finally, in this article, we assumed many parameters to be known, such as channel state, computation load, traffic over the backhaul, latency, etc. In practice, it would be useful to incorporate suitable learning mechanisms to predict the evolution of most of them. In summary, we believe that in the scenario depicted in this work, signal processing can play a key role in several aspects.

ACKNOWLEDGMENTS

This work was funded by the European Community 7th Framework Programme Project ICT-TROPIC, under grant 318784.

AUTHORS

Sergio Barbarossa (sergio@infocom.uniroma1.it) is a full professor at the University of Rome "La Sapienza." He has held various visiting positions at the Environmental Institute of Michigan (1988), the University of Virginia (1995 and 1997), and the University of Minnesota (1999). He received the 2010 EURASIP

Technical Achievements Award and the 2000 IEEE Best Paper Award. He is an IEEE Distinguished Lecturer and an IEEE Fellow. He is a member of the editorial board of *IEEE Signal Processing Magazine*. He is involved in international projects on heterogeneous networks, cloud computing, and radar remote sensing. His current research interests include distributed optimization, bioinspired signal processing, and self-organizing networks.

Stefania Sardellitti (stefania.sardellitti@uniroma1.it) received the M.Sc. degree in electronic engineering from the University of Rome "La Sapienza," Italy, in 1998 and the Ph.D. degree in electrical and information engineering from the University of Cassino, Italy, in 2005. She is currently a research assistant in the Department of Information Engineering, Electronics, and Telecommunications, University of Rome. She has participated in the European project WINSOC (on wireless sensor networks) and in the European project FREEDOM (on femtocell networks). She is currently involved in the European project TROPIC on distributed computing, storage, and radio resource allocation over cooperative femtocells. Her research interests are in the area of statistical signal processing, mobile cloud computing, femtocell networks, and wireless sensor networks.

Paolo Di Lorenzo (dilorenzo@infocom.uniroma1.it) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Rome "Sapienza," Italy, in 2008 and 2012, respectively, where he is currently a postdoctoral researcher in the Department of Information, Electronics, and Telecommunications. He has participated in the European projects Freedom (on femtocell networks) and SIMTISYS (on moving target detection through satellite constellations). He is currently involved in the European project TROPIC (on distributed computing, storage, and radio resource allocation over cooperative femtocells). His research interests are in bioinspired signal processing, adaptation and learning over networks, mobile cloud computing, and synthetic aperture radar processing. He received Best Student Paper Awards at the 2010 IEEE International Workshop on Signal Processing Advances for Wireless Communications, the 2011 European Signal Processing Conference, and 2011 International Workshop on Computational Advances in Multisensor Adaptive Processing. He also received the 2012 Gruppo Telecomunicazioni e Tecnologie dell'Informazione Best Doctoral Thesis Award.

REFERENCES

[1] (2013, Nov.). ETSI summit on future mobile and standards for 5G. [Online]. Available: http://www.3gpp.org/news-events/conferences/1515-etsi-summit-on-future-mobile-and

[2] European Project METIS. [Online]. Available: https://www.metis2020.com

[3] M. R. Palacin, "Recent advances in rechargeable battery materials: A chemists perspective," *Chem. Soc. Rev.*, vol. 38, no. 9, pp. 2565–2575, 2009.

[4] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Commun. Surveys Tutorials*, vol. 14, no. 4, pp. 1232–1243, 2012.

[5] N. Fernando, S. W. Loke, and W. Rahayu "Mobile cloud computing: A survey," *Future Gen. Comput. Syst.*, vol. 29, pp. 84–106, Jan. 2013.

[6] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks Applicat.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[7] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM Int. Conf. Mobile Systems, Applications, and Services*, San Francisco, CA, 15–18 June 2010, pp. 49–62.

[8] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[9] 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," in *Proc. INFOCOM 2013*, Apr. 2013, Turin, Italy, pp. 1285–1293.

[10] G. Fettweiss, "A 5G wireless communication vision," *Microwave J.*, vol. 55, no. 12, pp. 24–36, Dec. 2012.

[11] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Proc. 22nd IEEE Int. Conf. Distributed Computing Systems*, 2002, pp. 217–226.

[12] R. Balan, M. Satyanarayanan, S. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proc. 1st Int. Conf. Mobile Systems, Applications and Services*, 2003, pp. 273–286.

[13] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. 2nd Int. Conf. Mobile Computing, Applications, and Services (MobiCASE)*, pp. 59–79, Oct. 2010.

[14] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proc. HotOS*, Monte Verita, Switzerland, p. 8, May 2009.

[15] D. Huang, X. Zhang, M. Kang, and J. Luo, "Mobicloud: Building secure cloud framework for mobile computing and communication," in *Proc. 5th IEEE Int. Symp. Service Oriented System Eng. (SOSE)*, June 2010, pp. 27–34.

[16] M. D. Kristensen and N. O. Bouvin, "Scheduling and development support in the scavenger cyber foraging system," *Pervasive Mobile Comput.*, vol. 1, no. 6, pp. 677–692, 2010.

[17] J. Flinn, D. Narayanan, and M. Satyanarayanan, "Self-tuned remote execution for pervasive computing," in *Proc. 8th IEEE Workshop Hot Topics in Operating Systems*, Schloss Elmau, Germany, 2001, pp. 61–66.

[18] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[19] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *IEEE Pervasive Comput.*, vol. 3, no. 3, pp. 66–73, 2004.

[20] S. Ou, K. Yang, and Q. Zhang, "An efficient runtime offloading approach for pervasive services," in *Proc. IEEE Wireless Communication and Networking Conf. (WCNC2006)*, Las Vegas, NV, 2006, pp. 2229–2234.

[21] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, June 2012.

[22] G. Chen, B. T. Kang, M. Kandermir, N. Vijaykrishnan, M. J. Irwin, and R. Chandranouli, "Studying energy trade offs in offloading computation/compilation in Java-enabled mobile devices," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 9, pp. 795–809, Sept. 2004.

[23] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," *Proc. IEEE Int. Conf. Computer Communications 2012*, Mar. pp. 2716–2720.

[24] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems*, Atlanta, GA, 2001, pp. 238–246.

[25] B. G. Ryder, "Constructing the call graph of a program," *IEEE Trans. Softw. Eng.*, vol. 5, no. 3, pp. 216–226, 1979.

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

[27] P. Di Lorenzo, S. Barbarossa, and S. Sardellitti, "Joint optimization of radio resources and code partitioning in mobile cloud computing," submitted for publication.

[28] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Select. Areas Commun.*, vol. 31, pp. 2685–2700, Dec. 2013.

[29] I. Hwang, B. Song, and S. Soliman, "A holistic view on hyper-dense heterogeneous and small cell networks," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 20–27, June 2013.

[30] FP7 European Project. (2012). Distributed computing, storage and radio resource allocation over cooperative femtocells (TROPIC). [Online]. Available: http://www.ict-tropic.eu/

[31] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for mobile cloud computing in a multicell environment," submitted for publication.

[SP]