A Mobile Application Offloading Algorithm for Mobile Cloud Computing

Amal Ellouze, Maurice Gagnaire, Ahmed Haddad

A Mobile Application Offloading Algorithm for Mobile Cloud Computing

Amal Ellouze, Maurice Gagnaire, Ahmed Haddad
TELECOM ParisTech
Institut Mines-Telecom
46 rue Barrault 75013
Paris, France
amal.ellouze, maurice.gagnaire, ahmed.haddad2@telecom-paristech.fr

***Abstract*—In mobile cloud computing, offloading mobile applications to close remote servers appears as a straightforward solution to overcome mobile terminals processor and battery limitations. Remote execution leverages the high computation capacity of the server to enrich user experience and extend battery autonomy through energy savings. However, application offloading is energy efficient only under various conditions. For that purpose, we propose an original algorithm called MAO (Mobile Application's Offloading) triggered by two conditions : the current CPU load and State of Charge (SoC) of the battery. On the basis of various traffic scenarios mixing interactive and delay tolerant mobile applications, we study through numerical simulations the efficiency of the MAO algorithm and assess its performance in terms of rejected jobs and the amount of energy savings achieved. Rejected jobs are those unable to meet user quality of experience (QoE) and/or energy efficiency requirements. Evaluations on simulated workloads show that both traffic loads and user's radio mobile environment have a direct impact on the efficiency of the MAO algorithm.**

I. Introduction

The performance of existing mobile handsets are today limited by two factors: the scale of integration of electronic devices and the capacity of the batteries. In this context, several recent studies propose to encompass the limited capacity of the existing batteries by means of a transient offload of some of the applications from the mobile terminal to a remote server. In this matter, the round-trip time between the mobile terminal and the server is a key parameter that conditions the level of interactivity of the applications that can be offloaded. Several investigations have been recently dedicated so far to extend the battery capacity of mobile terminals. Remote execution ([1],[2],[3]) can be a solution to go over the limitations of mobile devices and can be seen as a way for extending battery autonomy by means of applications offloading [4]. By benefiting from an increased computation offloading capacity of the remote server, the execution time of these applications is reduced enabling to decrease the enrgy consumption. Similarly, thanks to the increased memory of the remote server, it's possible to provide a large set of applications to the end user ([5],[6]). The rest of the paper is organized as follows. Section II reviews the related work on offloading systems for mobile environments. Section III presents the MAO algorithm and its main components. We evaluate by means of numerical simulations the performance of MAO algorithm and analyze the results in Section IV. We conclude and discuss future work in Section V.

II. Related Work

For the last two decades, there have been many attempts to improve energy and CPU efficiency in mobile handsets. These approaches enable to reduce application execution time on mobile devices, thus decreasing the energy consumption of CPU [7]. These attempts could be classified into two approches: fine-grained and coarse grained tasks offloading schemes [8]. The first one relies on application developers to modify the code to handle partitioning, state migration, and adaption to various changes in network conditions. The second approach assumes that the full process/program or full Virtual Machine (VM) is migrated to the remote servers. Then programmers do not have to modify the application source code to take advantage of computation offloading.
Recent solutions include MAUI and CloneCloud([9],[10]). In [9], Microsoft's researchers presented a fine-grained solution called Mobile Assistance Using Infrastructure (MAUI) to reduce programmers work by partitioning the application code automatically, deciding at runtime which methods should be remotely executed. MAUI formulates the problem as an integer linear programming (ILP). A comparison of the energy consumption of three applications (face recognition application, video game and chess game) pointed out that energy savings from 27% up to 80% can be achieved thanks to MAUI. However, this approach works only on Microsoft Windows Platform. Moreover, they provide a server for each application, which is not scalable to handle many new applications. In [10], the authors proposed a more coarse-grained offloading approach that eliminates the programmer's help called CloneCloud where offloading is performed by a modified Dalvik VM automatically. Meanwhile, this approach requires pre-processing on the client side, which can also lead to excessive network traffic from the cloud network to the device. These two major approaches put forward some motivation, but do not provide Quality of Experience (QoE) guarantees

for Mobile users in case of offloading. Moreover, the impact of mobile environment in matter of quality of radio interface according to the position of the end user in the cell has not been considered in these works. Our offloading approach considers a combination of multiple constraints including CPU, State of Charge (SoC) of the battery, and bandwidth capacity. Since the concept of QoE is attracting a growing attention, the MAO algorithm aims to provide an admissible QoE to the end user whatever its position within the cell for a given application.[1] We also evaluate the performance of CPU intensive and I/O interactive applications. The objective is to focus on job offloading which is a convenient way to achieve energy consumption optimization if certain constraints are satisfied. The offload is assumed to be coarse grained. Our analysis is conducted in LTE environment operating at carrier frequencies lower than 2 GHz.

Our goal is to demonstrate that thanks to MAO, it is possible to increase indirectly the capacities of the CPU and of the battery of mobile terminals.

Two reasons motivate the rejection of a job by MAO: either a violation of the QoE of the end user or/and a negative power budget by offloading. By sake of simplification, we assume that an application server is located at the antenna's site.

Table I points out the specifities of MAO with respect to the most popular offloading algorithms in literature (MAUI and CloneCloud).

TABLE I
COMPARISON OF OFFLOADING ALGORITHMS .

Aspects	QoE Satisfaction	Network Conditions	User Mobility	CPU	SoC
MAUI	N.C	C	N.C	C	N.C
CloneCloud	N.C	C	N.C	C	N.C
MAO	C	C	C	C	C

where, "Network Conditions" refers to the fluctuated quality of the radio interface, "User Mobility" refers to the position of the end user within the cell, "N.C" as not considered and "C" as considered.

III. MAO ALGORITHM

In this section, we describe in more details MAO. In the proposed solution, the algorithm continuously monitors the CPU usage of the mobile phone and selects the appplication which is overloading the CPU. This application would be eventually offloaded. As soon as the SoC of the battery gets below 20% of the total capacity of the battery, the MAO algorithm is activated to test if the current application can be offloaded.

A. MAO Flowchart

When user runs an application, it should follow the process depicted in the flowchart shown in (Fig. 1). First, the system determines the expected response time when running on the mobile phone and examines the current SoC of the battery. If this response time is higher than a critical value determined by user QoE or/and the SoC of the battery is below of 20%. The algorithm determines whether this application 'can' be energy efficient when offloaded to the server or not. If it is not the case, the application is run on mobile. For those 'can', the system will check the quality of service of the whole system by collecting the response time of the remote server and the current bit rate of the radio channel. Then, the mobile send input data to the server which runs the application and sends back results. A job is rejected being unable to meet user QoE or/and energy efficiency requirements. In this case, a rejected job is neither run on the mobile terminal nor on ther server side. As indicated in the flowchart (Fig. 1), the algorithm loops back to decide whether to offload the next job or not.

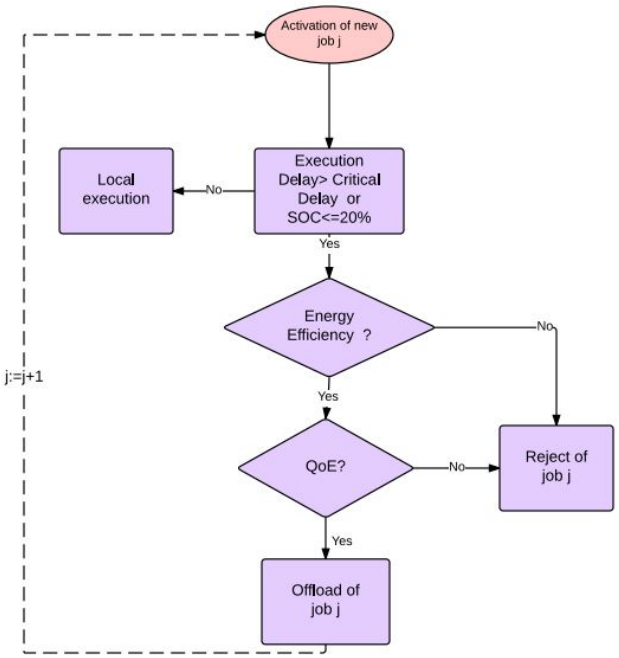

Fig. 1. MAO Flowchart.

B. Modeling of concurrent job processing on a single processor

We consider that each job j is characterized by a given number $N(j)$ of instructions. The processor of the mobile terminal is itself characterized by the number of instructions C_M it may process per second. The minimum delay $D^M_{min}(j)$ to process job j when it is the only one to be active is given by:

$$D^M_{min}(j) = \frac{N(j)}{C_M} \quad (1)$$

In practice, the processor of the mobile terminal has to deal with parallel jobs. Let us assume that this processor serves

[1]QoE: The overall acceptibility of an application or service, as perceived by the end user [23].

each job by small quanta of q instructions corresponding to a processing delay T of q/C_M seconds. We assume that the successive quanta inherent to the same job are enqueued in a FIFO queue before being effectively processed. As soon as a second job is activated, the quanta of instructions belonging to this second job are placed in a parallel queue. The CPU can then be seen as a server that manages alternatively the successive quanta from the two queues (principles of processor sharing)[11]. More generally, if n jobs are activated simultaneously on the mobile terminal, the CPU of the mobile terminal serves cyclically the n queues of quanta of instructions inherent to these parallel jobs. We have developed a processor sharing model enabling to simulate the behavior of such a queueing system. We assimilate the successive activations of a same application to distinct jobs. Based on this proceessor sharing model and given the processing speed C_M of the mobile terminal, the size $N(j)$ in number of instructions and the date of activation $t^a(j)$ of each job j, it is possible to determine the lower bound of the delay $D^{expected}(j)$ to run job j. Since our tool ignores in advance the eventual jobs that could be activated after instant $t^a(j)$, we can consider delay $D^{expected}(j)$ as a lower bound for serving job j. By sake of simplicity, we assume in the remaining of this paper that the size of the FIFO queue inherent to each active job is infinite. We also assume that the number of parallel queues (the number of simultaneous jobs) is not upper-bounded.

C. Concept of Critical Delay

The MAO algorithm is in charge of deciding to offload or not a new job request. For that purpose, it compares the expected execution delay $D^{expected}(j)$ of this new job to its average ideal[2] execution delay $D(j)$ on the considered smartphone[3]. We add a preferred waiting delay $d(j)$ within which the end-user is satisfied as it is specified in the ETSI 2009 [23]. We call then $D^*(j)$ the critical delay above which job j cannot be activated as it increases the probability of CPU overload and degrades the quality of experience:

$$D^*(j) = D(j) + d(j) \tag{2}$$

A new application is then eventually offloaded if one has:

$$D^{expected}(j) > D^*(j) \tag{3}$$

D. Energy balance

Let C_S and C_M be the capacity of the processors of the server located at the Base Station (BS) and of the mobile terminal respectively (both expressed in number of instructions per second). A same job j requires then a delay $D^S(j) = N(j)/C_S$ and a delay $D^M(j) = N(j)/C_M$ to be executed either on the server or on the mobile terminal respectively. We assume that $C_S > C_M$. If the offloading of job j is decided by the MAO algorithm, let $V(j)$ be the size in bytes of the data inherent to the activation of job j that have to be transferred via the air interface onto the remote server. We include in $V(j)$ the input data that have to be transferred from the mobile terminal to the server to activate the corresponding application, the output data corresponding to the result of his activation to be transferred back from the server to the mobile terminal[4]. Let $B = W \times \log_2(1 + \frac{S}{N})$ be the bit rate of the modem of the mobile terminal expressed in bit per second[5]. In case of offload, the transfer of the input/output data of job j necessitates globally a transmission delay of $V(j)/B$ seconds. Let us assume that the mobile terminal consumes P_M^{active} Watt to process a job and that this same CPU consumes P_M^{idle} Watt when it is idle. Let $\overline{P}$ the power consumption of an LTE-enabled User Equipment (UE) in Watt. $\overline{P}$is calculated using the transmission power P_{tr} and based on the power model in [14] which includes the contribution of active components of UE (amplifiers, oscillators,...). The power consumption curve can be divided into two pieces with two slopes depending on the transmission power with respect a device specific threshold γ. The two curve pieces are independently approximated by linear functions with slopes α_L and α_H respectively as in equation 4 [12]:

$$\overline{P}(P_{tr}) = \begin{cases} \alpha_L \times P_{tr} + \beta_L \text{ for } P_{tr} \leq \gamma \\ \alpha_H \times P_{tr} + \beta_H \text{ for } P_{tr} > \gamma \end{cases} \tag{4}$$

with the device specific parameters α_L, α_H, β_L, β_H and γ as given in [14]. P_{tr} is calculated using equation 5 from the system parameters with the following considerations: neglecting the closed-loop components of the Transmit Power Control (TPC) formula given in [15] and applying full path loss compensation:

$$P_{tr} = min(P_{max}, P_0 + 10 \times \log_{10}(M) + PL) \tag{5}$$

with the maximum transmission power allowed for LTE $Pmax$ (23 dBm for class 3 UE [16]), $P0$ the transmission power per RB, M the number of allocated RB and PL the path loss (the expression of PL and of the mobile environment are specified in Table III). If job j is processed on the mobile terminal, the minimum energy consumption $E_{min}^M(j)$ for this terminal is given by:

$$E_{min}^M(j) = P_M^{active} \times \frac{N(j)}{C_M} = P_M^{active} \times D_{min}^M(j) \tag{6}$$

If this same job is processed on the server, let us determine the energy $E^S(j)$ consumed by the mobile terminal for this offload. This energy is the the sum of the energy $E_{tr}(j)$ to proceed the up/down data transfers between the mobile terminal and the server, the energy $E_{idle,p}^M(j)$ consumed by the mobile terminal in the idle state while the job is processed on the remote server and the energy consumed by the mobile

[2]We mean by "ideal delay", the delay to process job j on the mobile terminal when it is the only one to be active.

[3]Delay $D(j)$ is determined based on number of instructions of job j as it is specified in the SPEC CPU2006 benchmark description [21].

[4]For the sake of simplicity, we consider that the data core of an application is already installed on the remote server.

[5]Shannon-Hartely Theorem where W is the channel bandwith, S is signal power and N is noise power of the radio signal.

terminalwhile the job is in the remote server queue $E^M_{idle,w}(j)$. We have then:

$$E^S(j) = E_{tr}(j) + E^M_{idle,p}(j) + E^M_{idle,w}(j) \tag{7}$$

The energy consumed by the up/down transmission is given by:

$$E_{tr}(j) = \overline{P} \times \frac{V(j)}{B} \tag{8}$$

The duration of the idle state of the mobile terminal corresponds to the average time spent by a job in the server. In reference to [27], the server can be modeled by a M/G/1 queueing system. The jobs arrive serially to the server according to a Poisson process[6] with an arrival rate λ. We assume that the server processes offloaded jobs not by quantas as in the case of the mobile terminal but as variable size batches of instructions (with no interleaving between the instructions as it has been considered at the mobile terminal). The service time of a job is then assumed to have a general distribution according to the different nature of the applications. In this context, the mean response time of the server per job is given by the response time of a M/G/1 queue in which we assume that the fraction of time that the server is busy is given by utilization rate ρ. We have then for the energy dissipated by the mobile terminal during the offload:

$$E^M_{idle,p}(j) = \frac{N(j)}{C_S} \times P^{idle}_M \tag{9}$$

$$E^M_{idle,w}(j) = \frac{\rho \times \frac{N(j)}{C_S}}{2 \times (1-\rho)} \times P^{idle}_M \tag{10}$$

The offloading of job j is meaningful only if the energy required by its offload on the remote server is lower than the energy that it should consume if processed on the mobile terminal. The delay required to transmit and receive $D_{tr}(j)$, process the job remotely on the server $D_{Server,p}(j)$ and wait in the server queue $D_{Server,w}(j)$is lower than the critical delay. These two requirements for offloading a job j are given by the following inequalities:

$$\begin{cases} E^M_{min}(j) > E_S(j) \\ \\ D_{Server,p}(j) + D_{tr}(j) + D_{Server,w}(j) < D^*(j) \end{cases} \tag{11}$$

Inequalities in (11) can be written as:

$$\begin{cases} P^{active}_M \times \frac{N_(j)}{C_M} > \overline{P} \times \frac{V(j)}{B} + P^{idle}_M \times \frac{N(j)}{C_S} \\ \qquad\qquad + P^{idle}_M \times \frac{\rho \times \frac{N(j)}{C_S}}{2 \times (1-\rho)} \\ \frac{N(j)}{C_S} + \frac{V(j)}{B} + \frac{\rho \times \frac{N(j)}{C_S}}{2 \times (1-\rho)} < D(j) + d \end{cases} \tag{12}$$

[6]Strictly speaking, such an approximation is meaningful only if the number of jobs to be processed by the server is large during the critical delay. This is not the case if one considers a single cell with a single user.

E. State of Charge of the battery

Smartphones designers observe that for SoC below of 20%, the behaviour of the battery shortly collapses. For this reason, we use it as a threshold to offload applications meeting the requirements. In this article, MAO decides to offload a job based on the tradeoff between battery consumption and QoE. We use a circuit-based battery model to capture the battery performance and predict accurately the amount SoC at both constant and variable loads [13]. The proposed model enables us to capture both battery circuit features and nonlinear battery capacity effects. SoC can be expressed as: $SoC = 1 - \frac{\alpha^A}{\alpha^f}$, where, α_A is the accumulated capacity during time period $[t_s, t_e]$ at the discharge current I and α_f is the full capacity. The consumed capacity $\alpha^c(I, \beta, L, t_s, t_e)$ which is dissipated during the total period $[t_s, t_e]$ at the discharge rate I, can be written as:

$$\begin{cases} \alpha^c = \alpha^f - \sum_{i=1}^{N} \alpha^A(I_i, \beta, L, t_{i-1}, t_i) \\ \alpha^A(I, \beta, L, t_s, t_e) = IF(L, t_s, t_e, \beta) \\ F(L, t_s, t_e, \beta) = t_s - t_e + \\ 2\sum_{m=1}^{+\infty}(\frac{e^{-\beta^2 m2(L-t_s)} - e^{-\beta^2 m2(L-t_e))}}{\beta^2 m^2}) \end{cases} \tag{13}$$

In equation(13), the first term $I(t_s - t_e)$ is the consumed capacity by the load I^C during the load period $[t_s, t_e]$. The second term $2I\sum_{m=1}^{+\infty}(\frac{e^{-\beta^2 m2(L-t_s)} - e^{-\beta^2 m2(L-t_e)}}{\beta^2 m^2})$ is the amount of discharging loss due to the current effect, which is the maximum recoverable battery capacity at t_e. L is the total operating time of the battery. m detremines the computational complexity and accuracy of the model.

The energy gain provided by the MAO algorithm can be expressed as the ratio of the gap of energy consumption between offload and no offload to the energy without offload.

$$G = \frac{E^M_{min}(j) - E^S(j)}{E^M_{min}(j)} \tag{14}$$

IV. Performance evaluation of the MAO algorithm

A. Mobile terminal and server characteristics

We consider a Samsung Galaxy S2 with a CPU operating at a speed C_M of 1.2 GHz. The power consumed when the terminal is in the Idle state is P^{idle}_M = 0.53 Watt. The power consumed when the terminal is in the Active state is P^{active}_M = 3.351 Watt. The Tx/Rx capacity between the mobile handset and the server is B [25]. We assume that the server on which is activated a new VM for each application offloading is equipped with a X86 CPU operating 4 times faster than the CPU of the mobile terminal (F).

B. Applications characteristics

We assume that three types of applications can be activated on the mobile terminal.

- **Chess game**: processing inherent to the execution of one move on the chess-board.
- **Speech recognition**: conversion of an analog speech to a text and vice-versa.

TABLE II
APPLICATIONS PARAMETERS.

Application	CPU Computation	Interactivity	$d(j)$ in seconds	$D(j)^*$in seconds
Chess game	High	High	0.2	7.329
Speech recognition	Low	High	1	1.75
Virus Scanning	High	Low	10	25.671

- **Virus scanning**: a program able to detect the presence of virus and, and if possible, to kill it[7].

The set of these three applications covers roughly the range of time constraints that are in general required by current mobile applications available on Smartphones and Tablets. The time constraint of an application corresponds to the admissible delay for the end-user between the instant of activation of the application and the instant at which this application provides the expected answer to the end-user.
Similarly, this set of three applications covers roughly all the range of CPU capacities that are in general required by current applications. We use for our numerical simulations the specifications of various applications' benchmarks provided in the Standard Performance Evaluation Corportation (SPEC-2006) [21]. We summarize in Table II the typical values of the delays $d(j)$ and $D(j)^*$inherent to the three applications [23].

C. Simulation Parameters

We conducted a simulation campaign in order to validate the analytical model. The main focus of the study was on confirming whether the available simulation parameters, behave as theoretically expected. Simulation assumptions were based on LTE environment, as summarised in Table III.

TABLE III
SIMULATION PARAMETERS.

Parameter	Value
Cellular layout	$1\,Single\,hexagonal\,cell$
Cell radius	$500\,m$
Path loss model	$PL = A{+}30{\times}\log_{10}(\frac{d}{d0}){+}X_f{+}X_h + \sigma$
Channel fading	$Typical\,urban$
Carrier frequency	$2GHz$
System Bandwidth	$5MHz\,(25RBs)$
Thermal Noise per RB	$-121.45\,dBm$
Bit rate of the mobile	$600Kbps < B < 16Mbps$
Traffic	$Control\,messages,\,data\,traffic$
Number of UE	1
Number of application activation	100
Inter-arrival in seconds λ^{-1}	$2, 15, 30$
Simulation duration	$3\,hours$

where, A is a free space pathloss, $d0$ is the reference distance, X_f is a frequency correction factor, X_h is a correction factor for BS high and σ is the shadowing standard deviation [28].

[7]The size we used of the file to scan is fixed to 1.5MB.

D. Numerical results

For the set of simulations considered in this part, we assume that the three applications softwares are installed on the remote server.

1) Variable Traffic Workloads:
In the first set of simulations, we randomly activate 100 jobs corresponding to variable loads of applications under given network conditions[8]; this activation is done in a serial manner.

- **Simulation 1:**

We simulate three scenarios, each of which considers the activation of single application over a time period that ensures the end of execution of the last activated job (3 hours). The activation process is assumed to follow a Poisson distribution with parameter λ^{-1} representing the inter-arrival between two jobs in seconds. The higher loads considered in these simulations for the chess game, speech recognition and virus scanning applications correspond to jobs inter-arrival of 15, 2 and 30 seconds respectively. For such loads an energy gain of 68% and 49% can be observed for the chess game and the virus scanning application respectively (Fig. 2,4) while the rejection ratio turns around 3.5% (Fig. 2) and 6.6% (Fig. 4). Meanwhile, for speech recognition application, the rejection ratio at higher loads of 2 seconds is about 99% and no energy gain is observed. In fact, under such conditions, the intolerant constraint's latency of this application made the server unable to meet the user's satisfaction. When the inter-arrival of speech recognition jobs is around 2.5 seconds, we could drive to an energy gain of 86% and a rejection ratio of 0.2%.
At high loads, applications with both high interactivity and CPU computation are prone to be offloaded while at low loads, applications with high CPU demanding and low intervactivity are those to be migrated.
At average loads, applications involving high interavctivity with the end user and a low CPU computation are those expected to be executed remotely.

- **Simulation 2:**

To evaluate the performance of MAO under the interaction of multiple applications, we have considered a second scenario corresponding to mixing equally (33% each) the chess game, the speech recognition and the virus scanning applications. The results are shown in (Fig. 5). We observe that beyond an inter-arrival λ^{-1} of (15,60) seconds (corresponding respectively to the highest, lowest considered offered load), the energy savings are equal to (34%,2%) where about (3%,0%) of jobs are rejected. The energy gain for the battery of the mobile terminal decreases considerably with λ^{-1} to reach a low bound of 3% when the MAO algorithm is applied for a few number of applications.

2) User Mobility:
Mobile users are usually subject to changing network environments, basically, due to user mobility. Hence, we evaluate in this part offloading decisions considering user mobility.

[8]We mean by network conditions the average value of shadowing when the user is at a short distance of the antenna (with no obstacles).

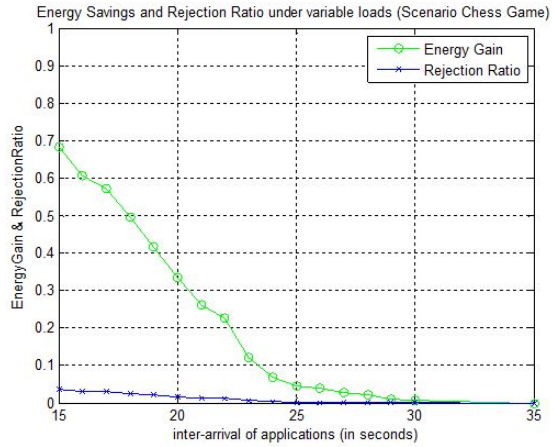

Fig. 2. Chess Game.

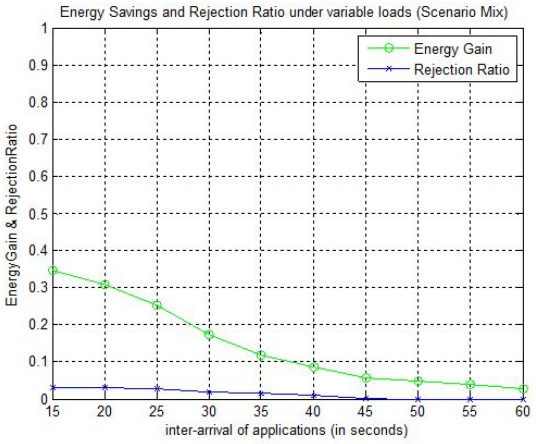

Fig. 5. Scenario Mix.

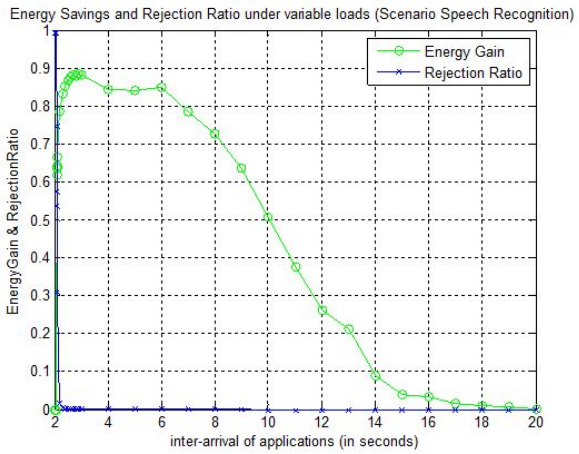

Fig. 3. Speech Recognition.

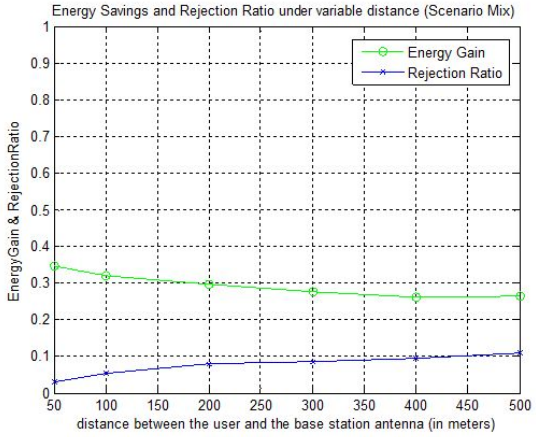

Fig. 6. User Mobility in the Cell.

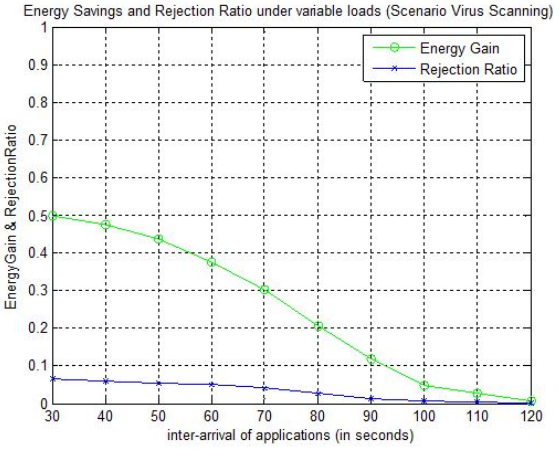

Fig. 4. Virus Scanning.

In this second set of simulations, we randomly activate 100 jobs of the mix scenario introduced previously considering average values of inter-arrival $\lambda^{-1} = 15s$ and shadowing. (Fig. 6) shows energy savings and the amount of rejected jobs of offloading applications considering user mobilty in the cell. When distance gets greater, offloading applications is less beneficial when comparing to a close distance to the BS's antenna. In fact, as far as the mobile user gets far from the antenna, more energy is required to offload the job. Meanwhile offloading remains profitable in terms of energy gains even at the edge of the cell (energy gain 24%, rejection ratio 11%, when the mobile user is positionned far away from the antenna).

V. Conclusion and perspectives.

In this work, we presented MAO, a decision model for mobile offloading that extends the battery autonomy of mobile terminals. We have first considered applications with low, medium and high stringent responses time (virus scanning, chess game and speech recognition). We have outlined that offloading some applications to remote VM located at the antenna could achieve significant energy gains on the mobile

terminal. This gain can then be exploited in two manners: Either it can be used to extend the battery autonomy of the mobile terminal, or it can also be used to activate enriched applications from this same terminal for a same operational cycle duration.

As further work, we intend to take into account the recovery effect of the battery, the overhead included by means of virtualization and architectures's compatibilty when offloading and also study the case of multi users within the cell. Our coming studies will also consider the case where the servers on which are activated the applications (a VM being activated for each application offloading) are not systematically located at the BS's site but higher in the radio network. In considering Passive Optical Network (PON) infrastructures to interconnect the BSs, it becomes then possible to mutualize the application servers in dedicated farms that could be used simultaneously by mobile users located in different cells. Meanwhile, an important additional constraint will have to be considered in this extended context in terms of admissible Round-Trip-Time between the mobile terminals and the server farm.

REFERENCES

[1] Bakos B., Fodor S., and Nurminen, J. K.: Distributed Computing With Mobile Phones: An Experiment with Mersenne Prime Search, Pervasive 2002.

[2] Gitzenis S., and Bambos N.: Mobile to base task migration in wireless computing. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, 2004.

[3] Kallonen T., and Porras J.: Use of distributed resources in mobile environment, The 14th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2006), September 29 - October 1, 2006, Split, Dubrovnik.

[4] Rudenko A., Reiher P., Popek G.J., and Kuenning G.H. Saving portable computer battery power through remote process execution. ACM SIGMOBILE Mobile Computing and Communications Review January 1998.

[5] Vallina-Rodriguez, N.; Crowcroft, J., "Energy Management Techniques in Modern Mobile Handsets," Communications Surveys and Tutorials, IEEE , vol.15, no.1, pp.179,198, First Quarter 2013.

[6] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. 2013. Mobile cloud computing: A survey. Future Gener. Comput. Syst. 29, 1 (January 2013), 84-106.

[7] Shiraz, M.; Gani, A.; Khokhar, R.H.; Buyya, R., "A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing," Communications Surveys and Tutorials, IEEE , vol.15, no.3, pp.1294,1313, Third Quarter 2013.

[8] X. Ma, Y. Cui, L. Wang and I. Stojmenovic "Energy Optimization for Mobile Terminals via Computation Offloading", 2012 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), 2012, pp. 236-241.

[9] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, Maui: making smartphones last longer with code offload, in Proceedings of the 8th international conference on Mobile systems, applications, and services, ser. MobiSys 10. New York, NY, USA: ACM, 2010, pp. 4962.

[10] B.-G. Cn, S. Ihm, P. Maniatis, M. Naik, and A. Patti, Clonecloud: elastic execution between mobile device and cloud. in EuroSys, C. M. Kirsch and G. Heiser, Eds. ACM, 2011, pp. 301314.

[11] Fabrice Guillemin and Jacqueline Boyer. 2001. Analysis of the M/M/1 Queue with Processor Sharing via Spectral Theory. Queueing Syst. Theory Appl. 39, 4 (December 2001), 377-397. DOI=10.1023/A:1013913827667 http://dx.doi.org/10.1023/A:1013913827667

[12] Dusza B, Ide C, Cheng L, Wietfeld C. An accurate measurement-based power consumption model for LTE uplink transmissions, In Proc. IEEE INFOCOM (Poster), Turin, Italy, 2013.

[13] J. C. Zhang, S. Ci and H. Sharif, An Enhanced Circuit-Based Model for Single-CellBattery, IEEE Applied Power Electronic Conference and Exposition (APEC), February 21-25, 2010, Palm Springs, pp. 672-675.

[14] Dusza, B., Ide, C., Cheng, L. and Wietfeld, C. (2013), CoPoMo: a context-aware power consumption model for LTE user equipment. Trans Emerging Tel Tech, 24: 615632. doi: 10.1002/ett.2702

[15] LTE physical layer procedures, Sep. 2009. 3GPP TS 36.213, V 9.3.0.

[16] UE radio transmission and reception, January 2012. 3GPP TS 36.101, V 9.10.0.

[17] R. Kemp, N. Palmer, T. Kielmann, H. Bal, Cuckoo: a computation offloading framework for smartphones, in: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE10.

[18] E. Lagerspetz and S. Tarkoma, Mobile search and the cloud: The benefits of offloading, in Ninth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2011, 21-25 March 2011, Seattle, WA, USA, Workshop Proceedings. IEEE, 2011, pp. 117122.

[19] China Mobile Research Institute, "Cloud-RAN (C-RAN): the road towards green RAN", White paper, Version 1.0.0.0, April 2010.

[20] T. Yu, R. Klamma, "Adaptive computation offloading from mobile devices into the Cloud", 10th IEEE Intl. Symposium of parallel and distributed processing with applications, 2012.

[21] John L. Henning, SPEC CPU2006 benchmark descriptions, SIGARCH Comput. Archit. News34,September 2006. Architecture for LTE Mobile Terminals", IEEE Conference, 2012.

[22] J. Bas, M. Katz, H. Lundqvist, T. Moreira, K. Ntontin, "Green-T: Enabling Techniques for Energy Efficient Mobile Terminals", IEEE 17th International Workshop of Computer Aided Modeling and Design of Communications Links and Networks (CAMAD), 2012.

[23] Human Factors (HF); Quality of Experience (QoE) reqirements for real-time communication services, ETSI TR 102 643 V1.0.1 (2009-12)

[24] A. Showk, S. Traboulsi, A. Bilgic, "An Energy Efficient Multi-Core Modem Architecture for LTE Mobile Terminals", IEEE Conference, 2012.

[25] Karel De Vogeleer, Grard Memmi, Pierre Jouvelot, Fabien Coelho: Modeling the temperature bias of power consumption for nanometer-scale CPUs in application processors. ICSAMOS 2014: 172-180

[26] Pan Hui, "Mobile Cloud Computing (MCC): a case for computation offloading and energy efficiency",EIT ICT Cloud Computing Summer School, Aalto University ,5th of June 2012.

[27] K. Kumar, Y. Lu, "Cloud Computing for mobile users: can offloading computation save energy?",Computer Journal, Vol. 43, Issue 4, pp. 51-56, IEEE Computer Society, April 2010.

[28] Shabbir, N., Sadiq, M. T., Kashif, H. and Ullah, R. (2011). Comparison of Radio Propagation Models for Long Term Evolution (LTE) Network. CoRR, abs/1110.1519.