Efficiency of Virtualization over MEC plus Cloud

Vincenzo Mancuso*, Paolo Castagno†, Matteo Sereno†
* IMDEA Networks Institute, Madrid, Spain † University of Turin, Italy

***Abstract*—We study average performance and costs for routing service requests in a virtualized environment, where either the MEC or the Cloud can serve user's requests. Employing a simple yet precise analytical model validated via simulation, we focus on latency, service request losses, energy consumption, and provider utility. The model is very effective in providing insight and guidelines for setting up server selection strategies with different characteristics (e.g., energy consumption, penalties, etc.) and performance requirements. In particular, our results show that the MEC is latency-efficient but incurs higher costs than Cloud, and then, to make its use sustainable, it is desirable that the MEC server is powered with renewable source energy.**

I. Introduction

The Mobile-Edge Computing (MEC) architecture aims to bring computing capabilities as close as possible to mobile subscribers [1]. This technology is a crucial ingredient of a variety of applications, such as autonomous driving, smart factory, virtual and augmented reality, and smart city applications. In this type of architecture, multiple small-scale server farms dedicated to the MEC computation are either co-located with base stations or placed within the backhaul ring [2]. User services can be provided by MEC hosts or Cloud datacenters, with different characteristics, because the MEC is characterized by computational resources (CPU/memory/storage) that are rather limited compared to those offered by the Cloud. However, the MEC can be reached with much lower delay, because of proximity. Energy consumption and costs can be also quite different for reaching and using either MEC or Cloud resources, the latter being generally more cost-efficient. Thereby, we suggest that MEC and Cloud resources should be combined to provide virtualized services with good performance and high energy efficiency. We study performance and costs of such system.

Related work. A relevant issue in virtualizing services via the MEC instead of the Cloud concerns energy consumption and strategies that could be used for its optimization. Several recent papers address these issues. For instance [3], [4] and [5] propose methods to optimize the energy consumption in the MEC by using different assumptions/scenarios, while [6] provides a summary of the possible applications of the MEC computing paradigm in the field of energy consumption optimization. In other studies, energy consumption and its optimization are tailored to specific application scenarios where the energy consumption is a cost optimization issue. Energy is a fundamental ingredient for the availability and reliability of such systems, for instance with mobile edge and vehicular MEC applications [7], or with MEC-assisted unmanned aerial vehicles [8]. Existing studies neglect the possibility to integrate MEC and Cloud paradigms, and there is no study on the efficiency of mixed MEC and Cloud virtualization solutions, which is the object of our work.

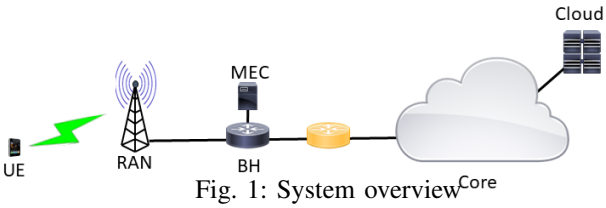


Fig. 1: System overview

Our contribution. We consider a simple application scenario in which a set of users submit service requests to the network. These requests can be processed either in a MEC host or in the Cloud, according to a generic request routing strategy. We develop an analytical framework to evaluate the average latency, loss, energy consumption, and utility yielded by the selected strategy. We use a discrete event simulator based on OMNeT++ to validate the model, and show that relying on both MEC and Cloud is key to achieve high energy efficiency and good service quality. The MEC results to be less cost-effective than the Cloud, but it offers low latency, and, given its small scale, could be run on green energy sources to make its adoption sustainable.

II. MEC-Cloud Virtualization

We consider the system illustrated in Fig. 1. Mobile users are connected to a base station, which in turn is connected to a backhaul subsystem. A MEC server is also attached to the same backhaul. A Core network connects the backhaul to the Internet, where a Cloud server is located.

Users generate service requests with arrival rate λ, which are sent for computation to a server running at either a MEC host or in the Cloud. We assume that the request can be dropped by the server, but not by network nodes. Routing a request towards the MEC or the Cloud is decided according to a probabilistic routing policy which sends a request to the MEC with probability α.

A request is a message of P_R bits, on average. When it reaches a server, some computation is carried out, which requires F floating-point operations, and a response is sent, consisting of P_S bits, on average. The intensity of served requests that are delivered to the mobile users is $\xi \leq \lambda$.

The total service time is T, which includes the transfer of the request to a server, the computation time at the server, and the transport of the server response back to the user.

Access. Service requests access the network via a base station, with the RACH procedure. The average time spent in the RACH, that we denote by T_A, is practically constant

and the RACH is loss-less as long as the request rate is below a few thousands per second [9], as we consider in this paper.

Base station forwarding. Once a user has obtained a grant to access the network, she can send the actual service request to the base station, who has sliced resources to store and forward the request towards a server. This uplink process incurs latency depending on the arrival rate λ and the service rate μ_U. In the downlink, the base station will receive service answers with intensity ξ and serve with rate μ_D.

Backhauling. The backhaul consists in typically one or more optical rings that connect several base stations and MEC servers, as described in [10]. With network slicing, the backhaul can be seen as a set of point-to-point links between base stations and MEC hosts. Service rates in the backhaul are indicated as μ_{Bu} and μ_{Bd} for uplink requests and downlink server responses, respectively.

Reaching the Cloud. The Core connects backhaul and Cloud. We represent the path as two links to and from the Cloud, although each link represents tens of hundreds of high-capacity links. We denote by μ_{Cu} and μ_{Cd} the service rate at the Core to and from the Cloud, respectively.

Service at MEC or Cloud. At the MEC host and at the Cloud server, we assume that service requests are served as in a FIFO queue with one or more processors and a finite buffer. The number of processors represents the number of virtual machines or CPU cores allocated to the service. The computation needed to provide service occurs at rate μ_M requests/s at the MEC and μ_C at the Cloud. We denote by π_M and π_C the corresponding buffer overflow (i.e., the loss).

Energy for networking. We distinguish between various sources of energy consumption: (i) negotiating network access over the RACH, (ii) user uplink transmissions, (iii) base station's transmissions to users, (iv) backhaul transmissions, and (v) Core network transmissions. For the RACH, we also consider that failures result in retransmissions at progressively higher transmit power, according to the standard power ramping scheme of 3GPP [11].

Energy for service execution. A service at the MEC or at the Cloud incurs an energy cost mainly due to computation or proportional to that. Thus, we assume that the service cost is proportional to the number of CPU cycles, thereby proportional to the average service time, be it μ_M^{-1} or μ_C^{-1}. The coefficient that multiplies the service time, however, could be different for MEC and Cloud, because of the efficiency of the different hardware used at the MEC and at the Cloud.

III. Model

A. Network

1) Access: Assuming that RACH losses are negligible, we can neglect the effects of RACH timeouts, finite number of RACH retries, and blocking at the base station queue. Therefore, using the RACH model presented in [9] with failures only caused by radio issues, the probability that a request succeeds in i RACH attempts can be obtained as

$$p_A(i) = e^{-\frac{i(i-1)}{2}} \left(1 - e^{-i}\right), \tag{1}$$

which does not depend on the arrival rate λ (at least before the arrival rate becomes comparable to the RACH capacity).

The average number of RACH attempts is

$$n_A = \sum_{i=1}^{\infty} i\, p_A(i) = \sum_{i=1}^{\infty} i\, e^{-\frac{i(i-1)}{2}} \left(1 - e^{-i}\right) 1.42. \tag{2}$$

The average latency is the latency in a successful attempt plus the latency in $n_A - 1$ failed attempts:

$$T_A = \frac{W}{2} + (n_A - 1)\,(Z + E[B_A]), \tag{3}$$

where W is the maximum time allowed in between a successful request is sent to the RACH and a grant is issued, $Z < W$ is the maximum time to accept a request at the base station, and $E[B_A]$ is the average RACH backoff time taken after a RACH failure, before a new attempt.

2) Base station, backhaul and Core: We use an M/M/1 FIFO queue to model the behavior of each network segment traveled by a service request. Therefore, we have 6 independent queues (base station, backhaul and Core in uplink and in downlink), although the Core queues are only for the Cloud.

The arrival rate at the base station uplink queue is the throughput of the RACH, which is λ, since we have assumed that the RACH introduces no losses. Therefore, the average time spent at the base station is $\frac{1}{\mu_U - \lambda}$. Similarly, the downlink latency at the base station is $\frac{1}{\mu_D - \xi}$.

The expressions for latency at backhaul and Core links are alike. We only have to consider that the backhaul sees arrival rate λ in uplink and ξ in downlink, while the Core sees $(1-\alpha)\lambda$ in uplink and $(1-\pi_C)(1-\alpha)\lambda$ in downlink.

The throughput, expressed in served requests per second, is simply given by the following formula:

$$\xi = ((1-\pi_M)\,\alpha + (1-\pi_C)\,(1-\alpha))\;\lambda = (1-\pi_L)\;\lambda, \tag{4}$$

where the π_L is the overall loss probability.

The average network latency for an accepted service request, not considering the service itself, is as follows:

$$\begin{aligned} T_{\text{net}} = T_A &+ \frac{1}{\mu_U - \lambda} + \frac{1}{\mu_D - \xi} + \frac{1}{\mu_{Bu} - \lambda} + \frac{1}{\mu_{Bd} - \xi} \\ &+ \frac{(1-\pi_M)\,\alpha\,\lambda}{\xi}\, d_M + \frac{(1-\pi_C)(1-\alpha)\,\lambda}{\xi} \\ &\cdot \left(d_C + \frac{1}{\mu_{Cu} - (1-\alpha)\lambda} + \frac{1}{\mu_{Cd} - (1-\pi_C)(1-\alpha)\lambda} \right), \end{aligned} \tag{5}$$

where d_M and d_C are the round-trip times of MEC and Cloud in absence of traffic. This latency is strongly affected by the value of α, which appears directly in the formula but also affects the values of ξ and π_C.

3) Network costs: The average cost incurred per time unit, due to the network component is proportional to the utilization of network elements:

$$\begin{aligned} \Phi_{\text{net}} = \lambda\, E_K \left[\sum_{i=1}^{K} E_{\text{RACH}}(i) \right] + \lambda\, P_R E_{\text{user}} + (\lambda\, P_A + \xi\, P_S) E_{\text{bs}} \\ + (\lambda\, P_R + \xi\, P_S) E_{\text{ring}} + (1-\alpha)\,\lambda\, (P_R + (1-\pi_C)\, P_S)\, E_{\text{core}}, \end{aligned} \tag{6}$$

where $E_{\mathrm{RACH}}(i)$ is the energy used in the transmission of a RACH request at the i-th attempt (after $i-1$ RACH errors), which ramps up failure after failure, E_K indicates the average over the number of RACH attempts K, E_{user} is the transmission energy per bit incurred by the user, E_{bs} is the energy per bit at the base station—with P_S bits per message in downlink, and with a coefficient P_A that expresses the cost of RACH acknowledgements—E_{ring} is the energy per bit transmitted over the backhaul ring, and E_{core} is the energy spent per transmitted bit over the core.

B. Service

1) Average latency: The latency of a request entering an M/M/$n_{(\cdot)}$/$k_{(\cdot)}$ FIFO queue depends on the probability to find a certain number of requests in the queue. The probability to have j requests in the queue can be computed as

$$p_{(\cdot)}(j) = \begin{cases} p_{(\cdot)}(0)\frac{\rho_{(\cdot)}^{j}}{j!} & j = 0, \cdots, n_{(\cdot)}; \\ p_{(\cdot)}(0)\frac{\rho_{(\cdot)}^{i}}{n_{(\cdot)}!n_{(\cdot)}^{i-n_{(\cdot)}}} & j = n_{(\cdot)}+1, \cdots, k_{(\cdot)}, \end{cases} \tag{7}$$

where $p_{(\cdot)}(0)$ is computed by using $\sum_{j=0}^{k_{(\cdot)}} p_{(\cdot)}(j) = 1$, and $\pi_{(\cdot)} = p_{(\cdot)}(k_{(\cdot)})$ is the loss probability relative to arrivals at the queue, be $(\cdot)$ either M or C.

The average latency in MEC or Cloud is a service time plus the waiting time incurred in case to find $j \geq n_{(\cdot)}$ requests:

$$T_{(\cdot)}(j) = \begin{cases} \frac{1}{\mu_{(\cdot)}}, & j < n_{(\cdot)}; \\ \frac{1}{\mu_{(\cdot)}} + \frac{j-n_{(\cdot)}+1}{n_{(\cdot)}\,\mu_{(\cdot)}}, & n_{(\cdot)} \leq j < k_{(\cdot)}. \end{cases} \tag{8}$$

The overall average server latency in the system depends on the routing probability, which affects the distributions of served requests over MEC and Cloud:

$$T_{\mathrm{serv}} = \frac{(1-\pi_M)\,\alpha\,\lambda}{\xi}\,E\left[T_M\right] + \frac{(1-\pi_C)\,(1-\alpha)\,\lambda}{\xi}\,E\left[T_C\right]. \tag{9}$$

2) Service cost: MEC and Cloud resources are only used for service requests that reach the server when there is available space in the local queue. Therefore, the average cost per second can be expressed as follows:

$$\Phi_{\mathrm{serv}} = (1-\pi_M)\,\alpha\,\lambda\,F\,E_M + (1-\pi_C)\,(1-\alpha)\,\lambda\,F\,E_C, \tag{10}$$

where E_M and E_C are energy spent per CPU cycle at the MEC and at the Cloud, respectively.

C. Utility

We model the utility of the system according to rewards, penalties and costs. The reward of a successful service request is denoted by R and is expressed in monetary units. However, to account for the importance of latency, we discount a fraction of the reward proportionally to the average latency of a served request, with a coefficient C_l. Discounted rewards generate a reward rate proportional to the intensity of served requests ξ. Losses might incur a monetary penalty P, at a loss rate $\pi_L\lambda = \frac{\pi_L}{1-\pi_L}\xi$ (note that $\pi_L < 1$ for any finite load). Eventually, network and server costs are additive and so we scale Φ_{net} and Φ_{serv} by a common coefficient C_e representing energy cost. The resulting utility is:

$$U = R\left(1 - C_l\left(T_{\mathrm{net}} + T_{\mathrm{serv}}\right) - P\frac{\pi_L}{1-\pi_L}\right)\xi - C_e\left(\Phi_{\mathrm{net}} + \Phi_{\mathrm{serv}}\right). \tag{11}$$

The coefficient C_l can be set to 0 if the service is not sensible to latency, or it can be set, e.g., to the inverse of the maximal tolerable latency. Note that the resulting reward could be negative in case of exceedingly high latency or high penalty, which might make sense in case of strict service level agreements. The utility function expresses the monetary flow (e.g., in Eur/s) for the provisioning of a service when the request rate is λ and the routing probability is α.

D. Approximated optimal α

It is intuitive that the optimal value of α must lead the system as far as possible from loss. Based on this consideration, we find near-optimal values of α.

Claim 1 (under-loaded system)**.** *If the costs C_l and P tend to zero and the following conditions hold*

$$\begin{cases} n_M\,\mu_M + n_C\,\mu_C \geq \lambda, \\ n_M\,\mu_M \leq n_C\,\mu_C, \end{cases} \tag{12}$$

then for R large enough, the optimal value of the routing probability is included in the following interval:

$$\left[\max\left(0, 1 - \frac{n_C\,\mu_C}{\lambda}\right), \min\left(1, \frac{n_M\,\mu_M}{\lambda}\right)\right]. \tag{13}$$

Rationale: Since the system can serve all the traffic due to the first condition, the loss can be taken towards zero, which is always convenient in terms of utility if R is large enough, so that the rewards grows with λ faster than the cost does. Therefore, imposing that neither the MEC nor the Cloud suffer losses, the following conditions are necessary:

$$\begin{cases} \alpha\,\lambda \leq n_M\,\mu_M, \\ (1-\alpha)\,\lambda \leq n_C\,\mu_C. \end{cases} \tag{14}$$

The above system is satisfied by all values of α in the range $\left[1 - \frac{n_C\,\mu_C}{\lambda}, \frac{n_M\,\mu_M}{\lambda}\right]$. The range is non-empty because of (12). Since values of α smaller than 0 and larger than 1 do not have a physical meaning, the interval reduces to (13). □

Claim 2 (overloaded system)**.** *If costs C_l and P tend to zero and the following condition holds*

$$n_M\,\mu_M + n_C\,\mu_C \leq \lambda, \tag{15}$$

then, for R large enough, the optimal value of the routing probability is included in the following interval:

$$\left[\frac{n_M\,\mu_M}{\lambda}, 1 - \frac{n_C\,\mu_C}{\lambda}\right]. \tag{16}$$

Rationale: Here losses cannot be avoided. Proceeding like for Claim 1, we reach the following inequalities, which guarantee full utilization of both MEC and Cloud:

$$\begin{cases} \alpha\lambda \geq n_M\,\mu_M, \\ (1-\alpha)\lambda \geq n_C\,\mu_C. \end{cases} \tag{17}$$

The above inequalities are equivalent to (16). □

Observation 1. *Simple, near-optimal values of α are given by the extremities of the intervals expressed by either* (13) *in case of under-loaded system, or* (16) *in case of overloaded system. Because of the meaning of α, we call the approximations computed with the leftmost (resp., rightmost) values of the intervals as "Cloud-pushing" (resp., "MEC-pushing").*

IV. Performance evaluation

Here we present a realistic evaluation scenario. We use $\rho = \frac{\lambda}{n_M \mu_M + n_C \mu_C}$ to denote the total load of the system and set $C_l = P = 0$ to evaluate the impact of energy efficiency. Furthermore, to validate our model we compare the model predictions against the results obtained by using a discrete event simulator developed by using OMNeT++. The simulator reproduces the topology of Figure 1, and includes detailed RACH and queueuing operations. The parameters used in our numerical study are as follows.

Messages. Users send requests of $P_R = 2000$ bits with exponential inter-arrival times. They receive service replies of $P_S = 4000$ bits, which is representative of short Internet transactions. RACH replies have $P_A = 200$ bits.

Network slicing and topology. The base station has a capacity of 500 Mbps in downlink and 200 Mbps in uplink, in line with a 5G cell in the sub-6 GHz channels. We use a network-sliced system and evaluate a slice that takes 5% of base station resources. In the backhaul, the network slice counts on 35 Mbps in downlink and 20 Mbps in uplink. Base station and MEC are separated by 2 optical segments. The round-trip time between user and MEC is $d_M = 27$ ms, not accounting for queueing. The Core network offers Gbps capacity, and Cloud and backhaul are separated by 100 links, which is the order of magnitude observed for real Clouds. The round-trip time between user and Cloud is $d_C = 66$ ms, not accounting for queueing.

RACH. We use a power ramping scheme with steps of 2 dB, starting at -11 dBm, which is the power required in a residential area with a target power signal level of -74 dBm and a typical pathloss of 63 dB, like in [11]. We use $W = 2$ ms, $Z = 0.5$ ms, and $E[B_A] = 10$ ms, with a RACH transmission opportunity every 1 ms, and 54 orthogonal RACH codes.

Transmission power. We take ETSI specifications for class-2 UEs (mobile devices) and medium-range base stations. The UE can transmit up to 0.2 W [12] (but can only use 5% of airtime) and the base station uses 6.3 W [13] (out of which, only 5% is for our slice). The backhaul efficiency is 2 Gbps/W per link, which is a typical value for good network equipment. Instead, in the Core, the efficiency is 20 Gbps/W per traversed link, which is a value representative of the currently available high-end network devices.[1]

Server specifications. The MEC uses a virtual machine for our service, pinned to a single core of a shared CPU, therefore $n_M = 1$. Its buffer can host 9 requests, so that $k_M = 10$. The Cloud uses 10 cores ($n_C = 10$) and has a buffer of 40 (i.e., $k_C = 50$). At the MEC, we use the specifications of an average dual-core 2-GHz CPU that consumes 80 W at full load. We do the same at the Cloud, with quad-core 2-GHz CPUs, each consuming 140 W in total, at full load. Serving each request requires $F = 10^7$ CPU cycles (i.e., $\mu_C^{-1} = \mu_M^{-1} = 5$ ms at 2 GHz, on average, with exponential distribution).

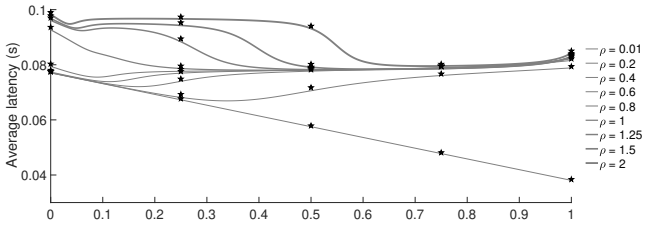


Fig. 2: Latency vs routing probability: model (lines) and simulation (markers)

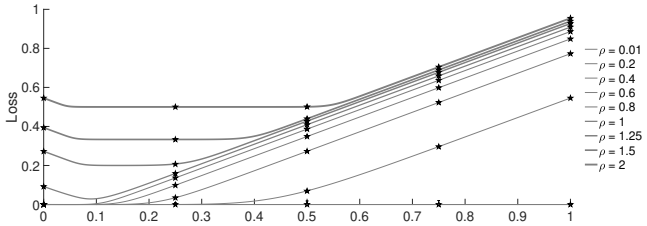


Fig. 3: Loss vs routing probability: model (lines) and simulation (markers)

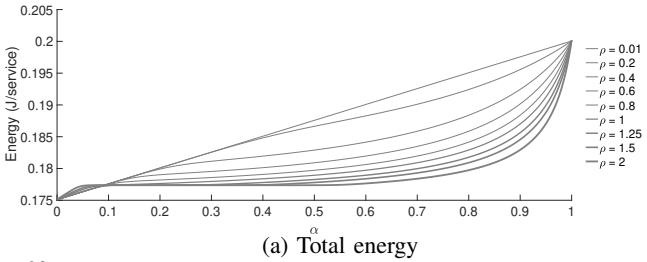
Costs and service revenue. We consider that the energy has a cost of $C_e = 0.13$ Euros/kWh, unless it is generated locally from renewable sources, in which case we assume that it has no cost. The service revenue R is set to 0.01 Euros per million of services, which yields revenues of the same order of magnitude of energy costs (so that they play a role in the optimization).

A. Latency and loss

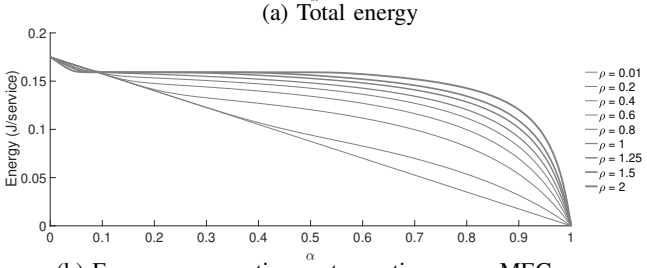
The average service latency is depicted in Figure 2 vs the routing probability for various values of the offered load. The figure shows model (solid lines) and simulation results (markers). The value of α that yields the minimum of latency widely changes with the load: low loads can be served at low latency at the MEC, while medium-high loads—before the system becomes overloaded—require more Cloud activity. However, under overloaded conditions, the MEC becomes progressively preferable, since the latency at the Cloud can become too high. In overloaded conditions the latency has a lower plateau due to the MEC queue, and a higher plateau that corresponds to the saturation of the Cloud.

Figure 2 shows that using the MEC is key to achieve low latency. However, using mostly the MEC incurs high losses at even low-medium loads, as shown in Figure 3. This occurs because, in the considered example, the MEC only provides a small fraction of the service capacity. The figure shows that low losses can only be achieved at either very low load or at low-medium loads and low values of α, i.e., by routing requests (in most cases) to the Cloud. Note that the above described results do not depend on energy, costs and prices.

[1] Source: https://www.servethehome.com/qsfp-v-sfp-v-10gbase-t-testing-power-consumption-results-and-recommendations/

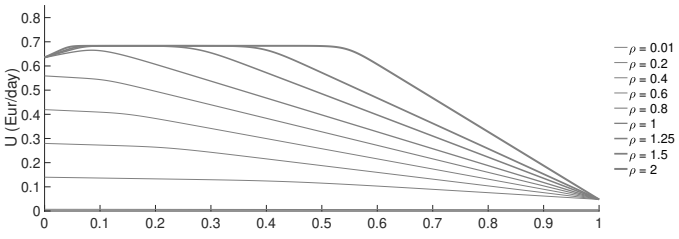


(a) Total energy

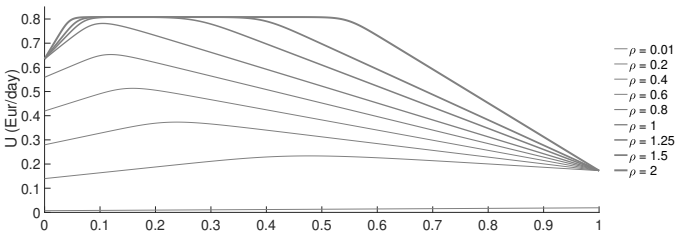


(b) Energy consumption, not counting green MEC energy

Fig. 4: Average energy spent overall in the system per each served request



(a) With non-free MEC energy



(b) Green MEC (free energy)

Fig. 5: Utility vs routing probability

Figures 2 and 3 show a good accuracy between model and simulation. Limited errors are visible in the latency, in the order of one millisecond. Errors are due to the fact that flows in the simulator are not necessarily Markovian like we have assumed in the model, where we also neglect the impact of load on the RACH.

B. Energy efficiency

Figure 4 shows the average energy used for each service. This includes the energy spent for unserved requests. In Figure 4a we can see that using the Cloud is more energy-efficient, which is due to low networking cost with respect to CPU, and to the fact that most of networking costs are due to wireless transmissions, which cannot be avoided, as also pointed out in [14]. Curves for high load, which experience high loss and saturated throughput, only slightly vary with α in a wide range of values. This means that changing α can have large impact on latency (especially for under-loaded systems) and loss, while the energy efficiency can be kept under control although the MEC is not sustainable.

It is interesting to point out that if the MEC only uses energy produced by renewable sources the trend flips, and the MEC becomes always convenient (see Figure 4b).

C. Utility

The utility is reported in Figure 5a. Underloaded systems are optimized by using mostly the Cloud and significant MEC operations are required only for loads well above 1. However, if the MEC is green and its energy has no cost (or negligible), Figure 5b shows that the value of α that maximizes the utility can change a lot with the load. In particular, we can see that the MEC is progressively less useful as the load increases, until the system begins to experience significant losses. This is qualitatively similar to the behavior of latency curves in Figure 2. In general, these results tell that the MEC can be latency-efficient but not energy- and utility-efficient unless it can run (almost) fully on green power. That would be doable because running a (small) MEC host in full can require as low as a few hundreds of W.

Note that the approximations shown in Section III-D identify the interval in between the two knees of the utility curves of Figure 5 (and the left knee is replaced by 0 in underloaded systems, as per (13)). Using the Cloud-pushing and MEC-pushing approximations would yield the sub-optimality ratios shown in Figure 6. The former always yields utility values very close to the optimum if the MEC energy has a cost, otherwise, the "MEC-pushing" approximation is preferable.

V. Conclusions

We modeled performance, costs and utility of a simple system that randomly routes service request to virtualized resources in the MEC and the Cloud. The model is tractable and allows to evaluate the average system behavior, which in turns allows to optimize the routing probability, for which we provided extremely simple-to-compute approximated values. Energy costs are key to evaluate the system, as shown by comparing a legacy system to one in which the otherwise not-sustainable MEC can be run on renewables. The model can be also used to set up optimization problems with latency and loss constraints, whose study we leave for future work.

Acknowledgements

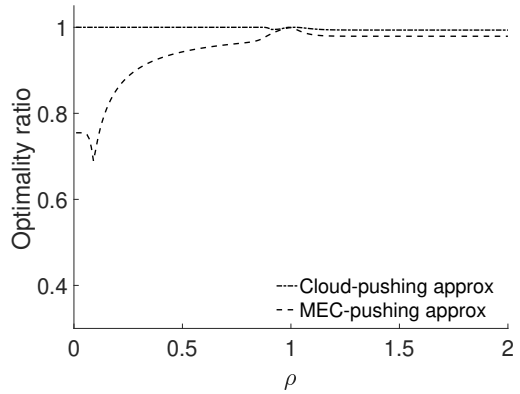
The work was supported by the Regione Piemonte, Italia through the HOME project of the framework program POR FESR 14/20, and by the Region of Madrid, Spain, through the TAPIR-CM project (S2018/TCS-4496).

References

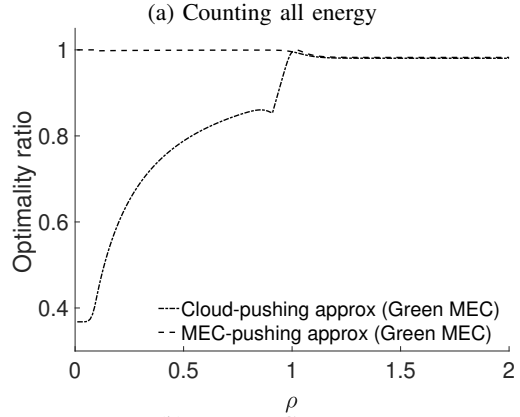
[1] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," ETSI White paper, 2014. [Online]. Available: https://portal.etsi.org

[2] A. A. Franklin and S. D. Tambe, "Multi-access edge computing in cellular networks," *CSI Transactions on ICT*, vol. 8, 2020.

[3] L. Chen, J. Wu, and J. Zhang, "Long-term optimization for MEC-enabled HetNets with device–edge–cloud collaboration," *Computer Communications*, vol. 166, 2021.



(a) Counting all energy



(b) Free MEC energy

Fig. 6: Ratio between utility values computed with the Cloud-pushing and the MEC-pushing approximations defined in Observation 1 and the optimal utility

[4] C. Guleria, K. Das, and A. Sahu, "A Survey on Mobile Edge Computing: Efficient Energy Management System," in *Innovations in Energy Management and Renewable Resources*, 2021.

[5] S.-T. Hong and H. Kim, "QoE-Aware Computation Offloading to Capture Energy-Latency-Pricing Tradeoff in Mobile Clouds," *IEEE Transactions on Mobile Computing*, vol. 18, 2019.

[6] R. Gopi *et al.*, "n Enhanced Green Cloud Based Queue Management (GCQM) System to Optimize Energy Consumption in Mobile Edge Computing," vol. 117, 2021, p. 3397–3419.

[7] X. Gu and G. Zhang, "Energy-efficient computation offloading for vehicular edge computing networks," *Computer Communications*, vol. 166, pp. 244–253, 2021.

[8] N. N. Ei *et al.*, "Multi-uav-assisted mec system: Joint association and resource management framework," in *ICOIN*, 2021.

[9] P. Castagno, V. Mancuso, M. Sereno, and M. Ajmone Marsan, "A Simple Model of MTC Flows Applied to Smart Factories," *IEEE Transactions on Mobile Computing*, 2020.

[10] L. Cominardi *et al.*, "Understanding qos applicability in 5g transport networks," in *IEEE BMSB*, 2018.

[11] J. Thota and A. Aijaz, "On performance evaluation of random access enhancements for 5g urllc," in *IEEE WCNC*, 2019.

[12] ETSI, "5G; NR; User Equipment (UE) radio transmission and reception; Part 2: Range 2 Standalone," ETSI, Technical Specification (TS) 138 101-2, 07 2019, version 15.7.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/13810102/15.07.00_60/ts_13810102v150700p.pdf

[13] ——, "5G; NR; Base Station (BS) radio transmission and reception," ETSI, Technical Specification (TS) 138 104, 07 2020, version 16.4.0. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/138104/16.04.00_60/ts_138104v160400p.pdf

[14] Y. Ming *et al.*, "Modeling the Total Energy Consumption of Mobile Network Services and Applications," *Energies*, vol. 12, no. 1, 2019.