# Energy-efficient resource allocation in multi-tenant edge computing using Markov decision processes

Alessandro Spallina, Andrea Araldo, Tijani Chahed, Hind Castel-Taleb, Antonella Di Stefano, Tülin Atmaca

# Energy-efficient Resource Allocation in Multi-Tenant Edge Computing using Markov Decision Processes

Alessandro Spallina[1,2], Andrea Araldo[1], Tijani Chahed[1], Hind Castel-Taleb[1], Antonella Di Stefano[2], and Tülin Atmaca[1]

[1]Telecom SudParis, France; Institut Polytechnique de Paris
[2]Dept. of Electrical, Electronics and Information Engineering, University of Catania, Italy
{firstname.lastname}@telecom-sudparis.eu, ad@diit.unict.it

***Abstract*—We address the problem of a Network Operator (NO) owning limited resources at the network edge. The NO wishes to enable advanced services, by virtualizing and allocating such resources among multiple tenants, i.e., third-party Service Providers, co-existing at the edge, with different Quality of Service (QoS) constraints. The NO applies a resource allocation policy with the objective of minimizing energy consumption via switching off non-used resources while guaranteeing tenants QoS requirements. We propose a resource allocation policy based on Markov Decision Processes (MDP). In simulation we show that our policy is able to reduce energy consumption, by turning off unused resources, while meeting heterogeneous SP requirements. Our code is available as open source.**

***Index Terms*—Resource Allocation, Energy-efficiency, Multi-tenant Systems, Edge Computing, Markov Decision Processes.**

## I. Introduction

With the fast growth of new services such as augmented reality, online games, e-health, etc., each with different QoS requirements, the management of 5G network resources has to face new challenges. Two of such challenges are (i) the extremely low latency demanded by some of such services and (ii) the huge amount of traffic needed to be transmitted. To tackle those, the paradigm of Edge Computing (EC) consists in deploying computational resources in *edge nodes*, very close to the users, e.g. at (micro) base stations, access points, roadside units, local central offices.

We consider a physical infrastructure, composed of an edge node or a cluster of edge nodes, owned by a *Network Operator* (NO). The NO virtualizes the available computational resources, in the form of servers, which may be Kubernetes Pods or similar, and makes them available to different Service Providers (SPs). Each SP may serve a certain class of traffic, e.g., vehicular, multimedia, etc., with different requirements, e.g., latency constraints. SPs act as tenants [1] and can run their services at the edge using the allocated resources.

The NO has to solve the resource allocation problem, i.e., to decide the amount of resources to allocate to each SP in order to meet their QoS requirements and at the same time minimize power consumption by switching off the non-used servers. In the cloud, resources may be assumed infinite and can be given to tenants as long as they are willing to pay; in the edge nodes, instead, resources are scarcer, which makes the allocation problem more challenging: allocating many resources to one SP implies allocating less to others. An appropriate allocation policy is thus needed and is the object of our present work.

The key contributions of our paper are the following:

1) We formalize a time-slotted Markov Decision Process (MDP) model representing our multi-tenant resource allocation problem.
2) We make use of histograms in our model for the traffic arrival as well as service. In this way, our approach can be adapted to capture real-life traffic traces and the way requests would be processed in the edge node.
3) We propose a resource allocation policy, which implements a trade-off between QoS requirements of tenants versus energy consumption of the infrastructure.
4) We show, through simulations, the performance of the policy in the presented scenario with heterogeneous requirements' SPs.
5) To ensure reproducibility, we provide an open-source implementation of our policy and the simulation environment. [2]

The remainder of this paper is organized as follows. Section II contains the description of our system and model. We then illustrate our results in Section III. Finally, section IV concludes the paper.

## II. System Model

We assume, as indicated above, an edge node owned and managed by a single NO with a limited quantity of general-purpose computational resources, in the form of servers. The aim of the NO is to allocate these resources to different tenants, each with specific QoS requirements, while minimizing energy consumption by switching off non-used servers.

978-1-6654-0601-7/22$31.00 

### A. Queuing model of Service Providers

Let $s^{\max}$ denote the maximum number of servers available in the system. We discretize the time into slots of duration $T$. In each time-slot, the number of *requests* arriving to the $i$-th SP is a discrete random variable with finite support. We denote by *Arrivals Histogram* $H_i^{\text{arr}}$ its probability mass function. The server performance for each SP is also represented by a discrete random variable with finite support, which counts the number of requests that can be processed by a server of SP $i$ in a time-slot. We denote by *Server Capacity Histogram* $H_i^{\text{cap}}$ its probability mass function. Observe that this formulation allows requests of different SPs to have different complexities, thus requiring different processing times, which would be reflected into different $H_i^{\text{cap}}$.

When requests of SP $i$ arrive, they are placed in the queue of SP $i$, of maximum size $m_i^{\max}$. Such queue is served by $s_i$ servers. The number $s_i$ can change at every time-slot: the NO decides at each time-slot $k$ the number $s_i(k)$ of servers allocated to SP $i$. As a consequence, the number of requests that an SP can process (i.e., *departures*) in a given time-slot is the sum of $s_i$ identical random variables $H_i^{\text{cap}}$ defined above, which is also a discrete random variable with finite support. More formally, the probability mass function of departures of SP $i$ is given by the following convolution:

$$\textit{Departures Histogram } H_{\text{dep}}^{s_i} \triangleq \underbrace{H_i^{\text{cap}} \circledast \cdots \circledast H_i^{\text{cap}}}_{s_i \text{ times}} \quad (1)$$

Although our model is time-slotted, the underlying system is in reality continuous. We represent the state of SP $i$ as a pair $(m_i(t), s_i(t))$, indicating the number of requests in the queue and the number of allocated servers, respectively, at instant $t$. Such a state continuously evolves with $t \in [0, +\infty[$, as $m_i(t)$ can change at any moment. To discretize the formulation, we sample the system and update the states of SPs only in the initial instant of the time-slots. For $T$ denoting the duration of a time-slot, for instance $T = 1$ sec, the $k$-th time-slot is $k = [t_k, t_k + T[$, where $t_k$ is the starting instant of the time-slot. We associate a state to each time-slot corresponding to the sampling of $(m_i(t), s_i(t))$ at the beginning of that time-slot:

$$m_i(k) = m_i(k - \text{th slot}) \triangleq m_i(t_k) \quad (2)$$

$$s_i(k) = s_i(k - \text{th slot}) \triangleq s_i(t_k) \quad (3)$$

### B. System Evolution

The NO decides at each time-slot how many servers $s_i$ to allocate to each SP $i$. Within time-slot $k = [t_k, t_k + T[$, we simplify the system evolution assuming the following order of events:

1) The NO observes the current state $\{(m_i(k), s_i(k))\}_{i=1}^N = \{(m_i(t_k), s_i(t_k))\}_{i=1}^N$, i.e., the number of requests in the queue and the number of allocated servers for SP $i = 1, \ldots, N$.
2) The NO decides the number of servers $(s_1, \ldots, s_N)$ for each SP, constrained to $\sum_{i=1}^N s_i \leq s^{\max}$. We assume that the number of servers changes instantaneously after $t_k$, i.e., $s_i(t) = s_i, t \in [t_k, t_k + T[$.
3) The requests that have arrived within time-slot $k - 1 = [t_k - T, t_k[$ for SP $i$, whose number we denote by $a_i(k-1)$, are placed in the respective queue, up to $m_i^{\max}$; all the exceeding requests are lost.
4) Each SP $i$ dequeues a request from its queue whenever one of its $s_i$ servers becomes free. The number of processed requests in time-slot $k = [t_k, t_k + T[$ is denoted by $p_i(k)$.

The number of requests in the queue of the $i$-th SP during the $k + 1$ time-slot is (see Sec. 3 of [3]):

$$\underbrace{m_i(k+1)}_{m_i(t_k+T)} = \min\left(m_i^{\max}, \underbrace{m_i(k)}_{m_i(t_k)} + a_i(k)\right) - p_i(k) \quad (4)$$

The number of lost requests of the $i$-th SP during the $k + 1$ time-slot is:

$$l_i(k+1) = \max\{0, m_i(k) + a_i(k) - m_i^{\max}\} \quad (5)$$

Observe that the first term of the state of each SP, i.e., the number $m_i(k)$ of requests in the queue, evolves independently from the other SPs; the second term, the number $s_i(k)$ of allocated servers, is instead coupled to the others, which makes the resource allocation problem challenging.

### C. Markov Decision Process model

We model the entire system as an MDP, where the set of states and actions are:

$$\mathcal{S} \triangleq \left\{ \{(m_i, s_i)\}_{i=1}^N | m_i \leq m_i^{\max}, i = 1, \ldots, N, \sum_{i=1}^N s_i \leq s^{\max} \right\} \quad (6)$$

$$\mathcal{A} \triangleq \left\{ (\hat{s}_1, \ldots, \hat{s}_N) | \sum_{i=1}^N \hat{s}_i \leq s^{\max} \right\} \quad (7)$$

We decompose the probability of transition from a state to another for each SP in isolation. To this aim, let us first define $\mathbb{P}(\text{proc}_i = x|y)$, the probability for SP $i$ of processing $x$ requests given that $y$ requests are found in the queue at the instant when the server starts to pick requests from the queue:

$$\mathbb{P}(\text{proc}_i = x|y) \triangleq \begin{cases} H_{\text{dep}}^{s_i}(x) & \text{if } x < y \\ \sum_{x=y}^{\infty} H_{\text{dep}}^{s_i}(x) & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

The transition probability from $(m_i, s_i)$ to $(m_i', s_i')$, when the NO decides to allocate $\hat{s}_i$ servers (i.e., the action is $\hat{s}_i$) is then:

$$Q^{\hat{s}_i}(m_i, s_i \to m_i', s_i') = \begin{cases} Q(m_i, s_i \to m_i', s_i') & \text{if } s_i' = \hat{s}_i \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where, denoting with $p_i$ the number of requests processed in a time-slot, we define

$$Q(m_i, s_i \to m_i', s_i') \triangleq \mathbb{P}\left( m_i \to m_i' \,\middle|\, \begin{matrix} s_i \text{ currently active servers and} \\ s_i' \text{ active servers in the next time-slot} \end{matrix} \right)$$

$$= \sum_{a=0}^{\infty} \mathbb{P}(\text{arrivals}_i = a) \cdot \mathbb{P}\left( \begin{matrix} m_i \to m_i' \text{ and} \\ \text{arrivals}_i = a \end{matrix} \,\middle|\, \begin{matrix} s_i \text{ currently active servers and} \\ s_i' \text{ active servers in the next time-slot} \end{matrix} \right)$$

$$= \sum_{a=\max\{0, m_i' - m_i\}}^{m_i^{\max} - m_i} \mathbb{P}(\text{arrivals}_i = a) \cdot \mathbb{P}_i(p_i = m_i + a - m_i' | a + m_i) \quad (10)$$

$$+ \sum_{a=m_i^{\max} - m_i + 1}^{\infty} \mathbb{P}(\text{arrivals}_i = a) \cdot \mathbb{P}_i(p_i = m_i^{\max} - m_i' | m_i^{\max}) \quad (11)$$

The equality on the second line holds because of the formula of total probability, considering all the possible discrete events of having $a$ requests arriving in the considered time-slot, for $a = \max\{0, m_i' - m_i\}, \ldots, \infty$. To obtain the last equality, we have to consider that, thanks to (4)

$$0 \leq p_i = \begin{cases} m_i + a - m_i' & \text{if } a + m_i \leq m_i^{\max} \\ m_i^{\max} - m_i' & \text{otherwise.} \end{cases} \quad (12)$$

which also implies, by simple calculation, that $a \geq m_i' - m_i$. This justifies why we start the summation from $\max\{0, m_i' - m_i\}$ instead of 0.

### D. Cost

The NO aims to minimize an overall cost, which takes into account the tradeoff between the QoS experienced by each SP traffic and the cost for providing servers, which consists in the energy consumption, along with the related monetary cost, for the NO. The system cost is computed as $C \triangleq \sum_{i=1}^{N} C_i$, where $C_i$ is the cost related to the $i$-th SP, calculated in each time-slot:

$$C_i \triangleq c_i^m \cdot m_i + c_i^s \cdot s_i + c_i^l \cdot \mathbb{E}(l_i) + c_{\Delta_s^+} \cdot \Delta_{s_i}^+ + c_{\Delta_s^-} \cdot \Delta_{s_i}^- \quad (13)$$

where

- $c_i^m$ is the unitary cost to have a waiting request in the queue for one time-slot;
- $m_i$ is the number of requests waiting in the queue;
- $c_i^s$ is the unitary cost to have a running server during one time-slot;
- $s_i$ is the number of servers allocated to the SP;
- $c_i^l$ is the unitary cost to have a lost request;
- $\mathbb{E}(l_i)$ is the expected value of lost requests;
- $c_{\Delta_s^+}, c_{\Delta_s^-}$ are the costs of switching on/off one server, respectively;
- $\Delta_{s_i}^+, \Delta_{s_i}^-$ is the number of servers switched on/off in the current time-slot.

Via the cost coefficients, all terms in the cost are converted to a monetary metric. Note that, for what concerns the lost requests (the ones that arrive when the queue is full), we decided not to insert them into the state, which would have caused the state space to explode. We take them into account directly in the cost function as an expected value. In the numerical results we will show that this approximation does not impede our policies to work in practice. Note also that, while we use the expected number of losses in the MDP policy computations, we report in the plots of the numerical results the actual losses measured in the simulations.

### E. Proposed allocation policy

As we want to take into account priorities between SPs in the system (based on their traffic types, QoS requirements and/or willingness to pay for the resources), we can apply MDP to each SP "in sequence", going from the highest to the lowest priority SP.[1] This does not guarantee us cost-optimality for the whole system but allows us to enforce cost-optimality within each SP, along with the scalability advantage, since the state and action space of each MDP $i$ (corresponding to single SP $i$) is much smaller than the one corresponding to the entire system at once (Sec. II-C). Note that in case of shortage of resources, SPs with lower priority will unavoidably be the ones suffering from degraded performance, or even get blocked.

The policy we propose is based on the observations above. We call it *ConservativeMDP*. It combines multiple MDPs, one per SP, as follows. The SPs are divided into two groups: (i) a group with statically allocated servers and (ii) a group with dynamically allocated. This distinction corresponds to serving SPs with high QoS requirements (static group) versus SPs with lower QoS requirements (dynamic group); the second group allows more energy saving.

Assume that the SPs are ordered from highest to lowest priority. The sub-policy related to the static group is obtained by (i) calculating the optimal MDP policy for SP 1, (ii) taking the maximum number of servers $s_1^{\max}$ used in such a policy, (iii) statically fixing the number of servers of SP 1 as $s_1 = s_1^{\max}$, (iv) calculating another MDP policy for SP 2 considering that now the available servers in the system are $s^{\max\text{'}} = s^{\max} - s_1^{\max}$ and statically fixing the number of servers in SP 2 as $s_2 = s_2^{\max}$. We proceed similarly for the other SPs, in the order of priority.

The sub-policy related to the dynamic group is obtained by (i) calculating the optimal MDP policy for SP 1, (ii) reserving the maximum number of servers $s_1^{\max}$ dictated by such MDP to SP 1 even if these are not always used all, (iii) calculating the optimal MDP policy for SP 2, considering that the only servers available are now $s^{\max\text{'}} = s^{\max} - s_1^{\max}$. We proceed in the same manner for the other SPs, in the order of priority.

## III. Numerical Results

In order to evaluate the performance of the proposed policy, we developed a Python simulator. We used MDPToolbox [4] to calculate the MDPs. The code used to obtain the following results is available as open-source [2].

We assume 1-second slot duration. The plots in this work represent the average of 40 simulations each with 10K time-slots, corresponding to 2.8 hours. Using cost parameters inspired by reality (table I).

[1] Note that in case multiple SPs have the same priority and requirements, we could consider them as a single SP, in terms of policy application.

TABLE I
REFERENCE VALUES FROM [3] (SEC. 1.1)

| | |
|---|---|
| Server cost | 300$ per year |
| Waiting request cost | $6.2 * 10^{-}6$$ |
| Lost request cost | $6.2 * 10^{-}6$$ |

TABLE II
REQUIREMENTS

| Requirements | Services | |
|---|---|---|
| | *SP1: High Priority* | *SP2: Low Priority* |
| Latency | $< 60$ ms [5] | 5 sec [6] |
| Loss probability | $< 0.1\%$ [5] | "1 percent" [7] |

We compare our *ConservativeMDP* policy with a *Baseline* consisting in a static policy that equally divides all available servers among SPs, without using any MDP.

We consider a NO infrastructure with two SPs. We assume SPs 1 and 2 deploy services of different types: SP 1 has stringent time and reliability constraints (small response time and low probability of unserved/lost requests), as in Table II, while SP 2 has less stringent delay constraint and loss probability.

We may assume that in such a case SP 1 has a more expensive contract, which is equivalent to saying that every request correctly processed in SP 1 represents a higher revenue for the NO and that a failure to process SP 1 requests induces a higher cost for the NO than the failure to serve requests of SP 2. We model this situation by giving greater weight to the cost coefficients of SP 1, as shown in Table III.

We considered several scenarios with different arrival histograms but for reasons of space we only show results with arrivals histogram generated from a Gaussian window with $\sigma = 2$. The performance trends shown next remain the same for the other arrival histograms. Table IV shows all the parameters considered for the simulation. Note that we set a small queue size for SP1, as a longer queue would be useless, since requests being in the tail would not meet the stringent latency constraint.

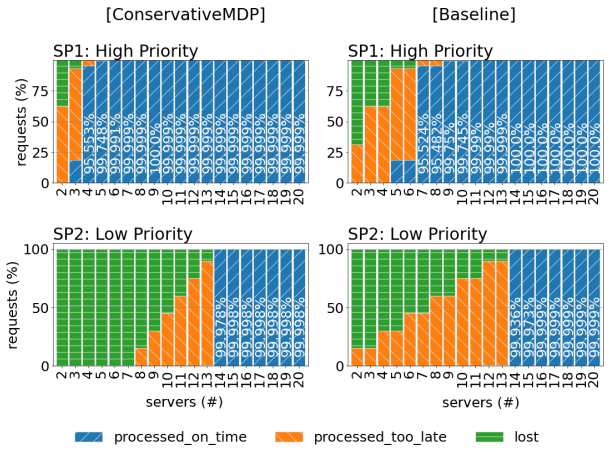


Fig. 1. Performance comparison between ConservativeMDP and Baseline

TABLE III
COST COEFFICIENTS

| Cost Coefficients ($) | | SPs | |
|---|---|---|---|
| **Description** | **Symbol** | **S1: High Priority** | **S2: Low Priority** |
| Waiting request | $c_i^m, i = 1, 2$ | 3.1e-5 | 6.2e-6 |
| Running server | $c_i^s, i = 1, 2$ | 9.64e-6 | 9.64e-6 |
| Lost request | $c_i^l, i = 1, 2$ | 3.1e-5 | 6.2e-6 |
| Server switch on | $c_{\Delta_s^+}$ | 9.64e-6 | 9.64e-6 |
| Server switch off | $c_{\Delta_s^-}$ | 1.0e-8 | 1.0e-8 |

TABLE IV
SIMULATION PARAMETERS

| | |
|---|---|
| $H_1^{\text{arr}}$ | Gaussian window, $\sigma = 2$, support from 0 to 16 |
| $H_1^{\text{cap}}$ | [0, 0, 0.5, 0.5] |
| Queue size of SP 1 $m_1^{\max}$ | 16 |
| $H_2^{\text{arr}}$ | Gaussian window, $\sigma = 2$, support from 0 to 20 |
| $H_2^{\text{cap}}$ | [0, 0.5, 0.5] |
| Queue size of SP 2 $m_2^{\max}$ | 100 |

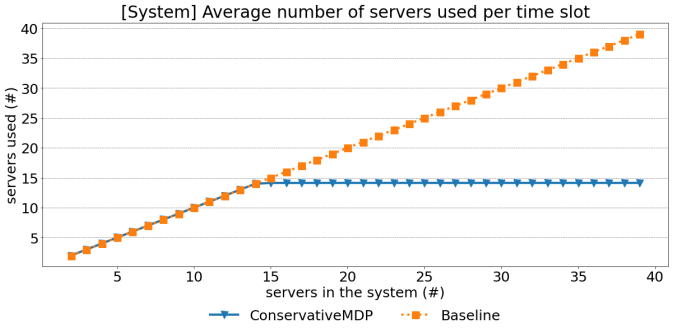


Fig. 2. Average server utilization per time slot of the overall system

In Fig. 1 we represent the (i) requests processed in time, (ii) the ones whose processing finished after the deadline of Table II (too late) and (iii) the requests lost, as the relative queue was already full. It is clear that ConservativeMDP meets the requirements, even with scarce resources, e.g., 14 available servers. At the same time, it keeps the energy cost and server utilization low (Fig. 2), by opportunistically turning off the servers for the low priority SP, when they are not strictly needed.

## IV. CONCLUSION

We considered in this paper dynamic allocation of virtualized computational resources, owned by a Network Operator (NO) at the edge. The NO needs to allocate such resources to several 3rd party Service Providers (SPs) each with different QoS requirements. We devised an MDP-based allocation policy that minimizes system costs, including energy consumption, while meeting SP QoS requirements. Our policy implements priorities between different SPs and is solved in a computationally efficient manner.

## V. ACKNOWLEDGEMENT

This work was partially funded by Beyond5G, a project of the French Government's recovery plan "France Relance" and carried out in the Plateforme Très Haut Débit (THD), a Fiber-To-The-Home platform of Télécom SudParis.

## REFERENCES


[1] Araldo, A., Di Stefano, A., and Di Stefano, A. (2020). Resource Allocation for Edge Computing with Multiple Tenant Configurations. ACM/SIGAPP Symposium On Applied Computing.

[2] A. Spallina, Multi-Tenant Resource Allocation with MDP: GitHub Repository, https://github.com/AlessandroSpallina/resource-allocation-mdp, 2021

[3] M. Bayati, Power Management Policy for Heterogeneous Data Center Based on Histogram and Discrete-Time MDP, Electron. Notes Theor. Comput. Sci. 337 (2018) 5-22.

[4] I. Chadès, G. Chapron, M-J. Cros, F. Garcia, R. Sabbadin, MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems – Ecography 37: 916–920 (ver. 0), 2014.

[5] K. Antonakoglou et al., On the Needs and Requirements Arising from Connected and Automated Driving, Journal of Sensor and Actuator Networks 9.2 (2020): 24.

[6] Sun, L., Zong, T., Wang, S., Liu, Y., and Wang, Y. (2021). Towards Optimal Low-Latency Live Video Streaming. IEEE/ACM Transactions on Networking.

[7] J. Wu, B. Cheng, C. Yuen, Y. Shang and J. Chen, Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks, IEEE Transactions on Mobile Computing, vol. 14, no. 4, pp. 688-701, 1 April 2015.