Distributed algorithms for multi-resource allocation

Francesca Fossati, Stephane Rovedakis, Stefano Secci

Distributed algorithms for multi-resource allocation

Francesca Fossati, Stéphane Rovedakis, Stefano Secci, *Senior, IEEE*

***Abstract*—Novel network infrastructures require the distribution of computing and network resource control to meet stringent requirements in terms of latency, reliability and bitrate. 5G systems bring a key novelty in systems design that it the 'network slice'as a new resource provisioning entity. A network slice is meant to serve end-to-end services as a composition of different network and system resources as radio, link, storage and computing resources. Conventionally, each resource is managed by a distinct decision-maker, platform, provider, orchestrator or controller. Naturally, centralized slice orchestration approaches are proposed in the literature, where a multi-domain orchestrator allocates the resources, for instance using a multi-resource allocation rule. Nonetheless, while simplifying the algorithmic approach, centralization can come at the expense of scalability and performance. In this paper, we propose new ways to distribute the slice multi-resource resource allocation problem, using cascade and parallel resource allocations that are functionally compatible with novel software platforms. We also show how to adapt the proposed algorithms to make them able to guarantee service level agreements on the minimum resource needed, and to take into account deadline priority policy scheduling. We provide an exhaustive analysis of the advantages and disadvantages of the different approaches, including a numerical analysis for a realistic setting.**

I. Introduction

Starting with the fifth-generation (5G) systems, communication networks need to meet novel diverse requirements. In 5G, mobile services are categorized in three main classes: enhanced mobile broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC) and massive machine type communications (mMTC) - characterized by bandwidth, latency, frequency and reliability requirements. To guarantee these services are adequately provisioned over an end-to-end infrastructure covering multiple resource domaines, the concept of network slice was formalized by 5G standardization bodies [1] - going beyond former application-level notions of network slices not encompassing multiple resource allocation.

The network slice, as meant starting with 5G systems, is an independent and logically-isolated end-to-end virtual network running on a shared physical infrastructure, aiming to provide the customers required service or vertical corresponding to different business domains such as health care, transport, smart office, agriculture, industry, automotive etc [2]. Hence a network slice spans different parts of the network as the access, transport, core and data-center segments, combining networking, computing and storage programmable resources [3]. The interest towards network slicing is motivated by the increasing programmability of the Radio Access Networks (RANs) as well as of the core elements, also thanks to the novel technologies such as Software-Defined Networking/RAN (SDN/SD-RAN) and Network Function Virtualization (NFV) [1].

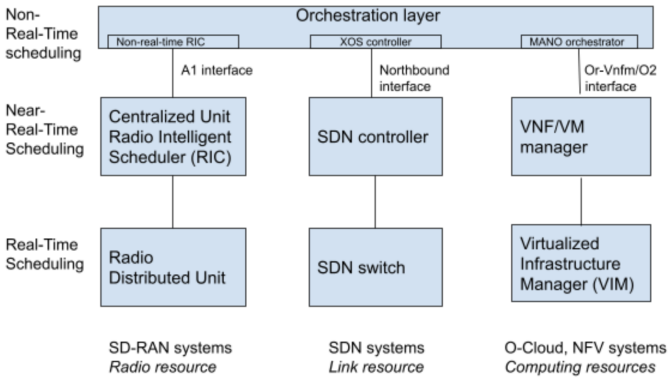


Fig. 1: Multi-resource subsystems in virtualized access networks.

Provisioning resources along an end-to-end path is therefore a multi-resource allocation problem. In the literature, different multi-resource allocation techniques exist, such as the Dominant Resource Fairness (DRF) [5] and the generalization of single-resource allocation rules ones [6]; such allocation rules are based on a centralized approach. In network slicing, a centralized approach implies the presence of a multi-domain orchestrator able to manage heterogeneous resources. Nowadays each part of the network is managed separately, as depicted in Fig. 1, and the presence of this kind of orchestrator may not be viable in practice for large-scale multi-domain deployments because resources can belong to different resource providers; e.g., the radio resource can be managed by radio operator, the cloud resource by a cloud service provider, the network link resource by an Internet service access provider.

In this paper we provide an extension of the preliminary work [7] toward the distribution of network slice resource allocation. We are interested in investigating distributed algorithms able to allocate slices. In particular, we propose three algorithms: two following a cascading approach and one a parallel approach.

Our reference technical framework is the one of novel virtualized access networks, depicted in Fig. 1. Large efforts are currently devoted to the design of virtualized/Software-Defined Radio Access Network (vRAN/SD-RAN) systems, such as Open RAN [4]. In these systems, a radio intelligent controller (RIC) can coordinate with other resource controllers (SDN link controllers, NFV/MEC orchestrators and resource managers) in the resource allocation and scheduling decisions. Such an interaction is envisioned as being done via an orchestration layer, using standard interfaces already identified in

Francesca Fossati is with CentraleSuplec, Gif-sur-Yvette, France. francesca.fossati@l2s.centralesupelec.fr

Stéphane Rovedakis and Stefano Secci are with Cnam, Cedric, 75003 Paris, France. Email: {firstname.lastname}@cnam.fr

A preliminary version of the content of this article appears in [7].

This work was funded by the Agence Nationale de la Recherche (ANR) MAESTRO-5G (ANR-18-CE25-0012) project.

corresponding sub-systems open-source projects. In particular, three level of scheduling are functionally identified: real-time (running at traffic processing device), near-real-time (running at resource controllers) and non-real-time (running at the multi-resource orchestration level). The algorithms we propose in this paper have the potential to be integrated to these systems, running at the near-real-time controller level.

We compare the approaches quantitatively (time complexity, message overhead, latency budget) and qualitatively (advantages, disadvantages). We then propose and test scheduling algorithms dealing with run-time constraints. The novel contributions of this work compared to [7] are:

- the generalization of the distributed algorithms in case of an arbitrary number of providers;
- the design of a priority-and-deadline-based scheduling algorithm to consider Service Level Agreement (SLA) targets;
- a complete analysis and demonstration of the delay budget and message complexity figures.

The paper is organized as follows. Section II provides a background on well-known single- and multi-resource allocation rules, along with a background on recent network slice allocation works. In section III we describe the three proposed algorithms, called Cascading Resource Allocation (CRA), Ordered Cascading Resource Allocation (OCRA) and Parallel Resource Allocation (PRA). Section IV provides the evaluation of the algorithms. Section V proposes a scheduling approach to meet SLA constraints. Finally section VI concludes the paper.

II. Background

We give an overview of common single and multi-resource allocation rules and related state of the art in network slicing.

A. Single and multi-resource allocation

Resource allocation problems can be divided into two types: *single-resource* ones, when only one resource has to be shared among the users, and *multi-resource* ones, when there are at least two resources to share.

Being $N = \{1, \cdots, n\}$ the set of users, a single resource allocation problem can be modeled by a pair (d, r) where d is a vector containing the demands of the n users (or tenants) and r is the amount of available resource that has to be shared among the users. An allocation a is a n-dimensional vector where a_i is the amount of resource given to user $i \in N$. The allocation has to satisfy: (i) non-negativity ($a_i \geq 0$), (ii) demand boundedness ($a_i \leq d_i$) and (iii) efficiency ($\sum_{i=1}^{n} a_i$=r) [8].

Let now $N = \{1, ..., n\}$ be the set of users and let $M = \{1, ..., m\}$ be the set of m available resources, then a multi-resource allocation problem can be modeled as a pair (D, R). R represents the vector of the available resource amounts and D is the matrix of demands ($d_{ij} \in D$ is the quantity of resource $j \in M$ demanded by user $i \in N$). The allocation matrix A is given by assigning to user i for resource j the amount a_{ij}=$d_{ij} \cdot x_i$, where $x = (x_1, ..., x_n)$ - with $0 \leq x_i \leq 1 \ \forall i \in N$ - is the vector of the ratios of resource allocated to each tenant. In the following, we assume that a feasible vector of ratios x must belong to the admissible region $\mathcal{F}$ s.t. $\sum_{i \in N} x_i \leq 1$ (so, a feasible allocation is s.t. $\sum_{i \in N} a_{ij} \leq r_j, \forall j \in M$).

We now summarize the most common allocation rules, which do solve an actual problem when the resource is scarce, i.e., when $\sum_{i \in N} d_i > r$ in case of single resource, or when it exists at least one resource j for which $\sum_{i \in N} d_{ij} > R_j$ in the case of multiple resources. Existing rules are as follows:

Weighted proportional rule [9]: computes the allocation that maximizes the $\sum_{i=1}^{n} p_i \log a_i$. When the weight p_i is equal to 1 for each user, the solution is such that increasing the allocation of a user, at least another user gets a loss in proportion larger than the gain of the previous user. When the weights are chosen equal to the demand d_i for each user i, the allocation assigns the same proportion of demand to each user, maximizing the fairness Jain's index [10].

Max-Min Fair (MMF) rule [11]: is an egalitarian solution, privileging the weak users (with small demands). It is calculated in a recursive way maximizing the minimum allocation, then the second lowest allocation, and so on. Ordering the users according to their increasing demand, i.e., $d_1 \leq d_2 \leq \cdots \leq d_n$, then MMF allocation for user i is given by $MMF_i(d, r) = \min\left(d_i, \frac{r - \sum_{j=1}^{i-1} MMF_j(d,r)}{n-i+1}\right)$.

α-fair rule [12]: is a family of allocation rules that generalizes the MMF and proportional rules. It is obtained by maximizing $\sum_{i=1}^{n} \frac{a_i^{(1-\alpha)}}{1-\alpha}$, where $\alpha \geq 0$ [1].

Mood value rule [13], [14]: is a solution conceived for complete awareness situation when users have full knowledge of the other users demand and of the available resource. Each user i can compute its minimal right min_i (what remains if all the other users are fully satisfied) and maximal right max_i (its own demand or the available resource, if the demand overcomes it), and the allocation is computed as $min_i + m(max_i - min_i)$ where m in [0,1] is the ratio between what remains when users get the minimum and $\sum_{i=1}^{n}(max_i - min_i)$.

Dominant Resource Fairness (DRF) [5] rule: generalizes the MMF rule for a multi-resource context. The allocation is the solution of the following problem:

$$\begin{aligned} &\text{maximize} \quad x \\ &\text{subject to} \quad ds_i x_i = ds_j x_j, \quad \forall i, j \in N \end{aligned} \tag{1}$$

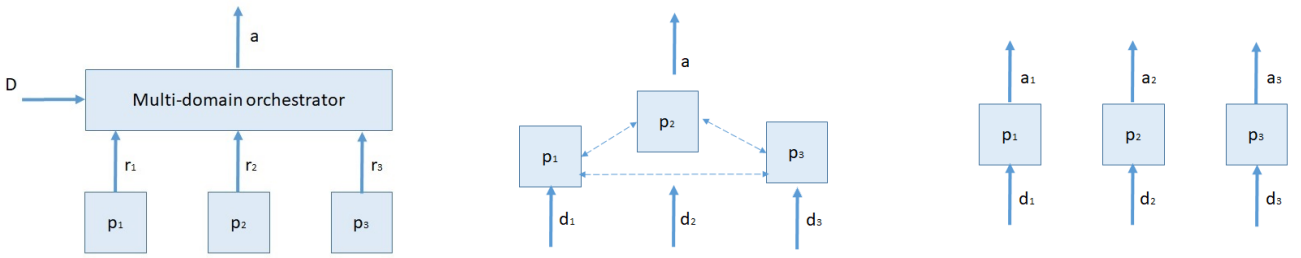
where $x \in \mathcal{F}$, and $ds_i = \max_{k \in M}\{\frac{d_{ik}}{r_k}\}$ is called dominant share of user i.

The above described allocation rules (single and multi-resource) are Pareto efficient.

Definition 1. An allocation a is Pareto efficient if for any other allocation a' we have $a'_{kj} > a_{kj}$ for some $k \in N$ and $j \in M \implies a'_{k'j} < a_{k'j}$ for some $k' \in N$.

Other multi-resource rules: A detailed description of multi-resource allocations can be found in [16], [6].

[1] If $\alpha = 1$ the solution coincides with the proportional one, if $\alpha = 2$ with the minimum delay potential allocation, that is the allocation obtained minimizing the total potential delay $\sum_{i=1}^{n}(\frac{1}{x_i})$ and if $\alpha \to \infty$ with the MMF rule [12].



(a) Centralized algorithm (b) Distributed algorithm (c) Decentralized algorithm

Fig. 2: Diagrams of multi-resource allocation approaches for an example with three providers p_i providing one resource r_i, with $i = 1, 2, 3$. D is a $n \times 3$ demand matrix, where n is the tenant number and d_i, with $i = 1, 2, 3$ is a demand vector for the resource i, a is the $n \times 3$ allocation matrix and a_i, with $i = 1, 2, 3$ is an allocation vector for the resource i.

A family of allocation rules, generalizing well-known single and multi-resource problems is the MURANES [6]. They are obtained solving the following problem:

$$\begin{aligned} &\text{maximize} \quad OWA(v) \\ &\text{subject to} \quad x \in \mathcal{F}, 0 \leq x_i \leq 1, \forall i \in N \end{aligned}$$

where $OWA(v) = \sum_{j=1}^{n} v_{(j)}$, $v_{(j)}$ is the j-th smallest element of $(v_1, ..., v_n)$, the input vector v is the (weighted) satisfaction vector[2] and $\mathcal{F}$ provides the region of acceptable solutions. See [6] for details. In addition to those already presented, let us mention: (i) the asset fairness [5], aiming at equalizing the resource allocated to each users, (ii) the Nash product that maximizes the product of the percentage of resource to allocate [5] and (iii) the Bottleneck-Based Fairness (BBF) equalizing the share received on the bottleneck resource [15].

B. Centralized vs distributed multi-resource allocation

All the described multi-resource allocation rules are centralized, i.e. they are meant to be run by a multi-domain orchestrator (e.g., at the non-real time orchestration layer in Fig. 1) that has a view on all the resources and receives the demands of the users, as represented in Fig. 2a. Centralized approaches imply that there is a relation between the resource; in particular, for all the proposed allocation in previous paragraph, a linear dependency between resources is considered.

Alternatively to centralized approaches, distributed approaches can be considered (e.g., running at the near-real time scheduling layer at resource controllers as in Fig. 1). With these approaches it is possible to let providers calculating their own allocation without sharing confidential information as the quantity of available resource. Naturally, the presence of other providers with their own allocation, can provide not efficient or unfeasible solutions, so a degree of collaboration between providers is necessary to obtain the final allocation, acceptable for all them (Fig. 2b).

A third approach that is possible to follow is to let allocate resources of each provider separately, without any information exchange among resource controllers (Fig. 2c). In this case, that we call de-centralized approach, we can have a resource-waste problem as described in [6]. For this reason, this type of approach is not considered in the following.

In this paper, we compare centralized and distributed algorithms, following the first two approaches. It is worth noting that combinations of the three described approaches can be conceived as well, in particular to deal with situations in which the hypothesis of linear dependency between groups of resources does not hold.

C. Resource allocation in network slicing

Recent works address resource allocation problems in network slicing from many points of view. Categorizing them by the type of resource, some such as [17] deal with the allocation of network link and computing resources, and others that address the radio allocation [18], [19].

Many objectives are adopted for the computation of a slice allocation. Some examples are (i) the maximization of the slice customers profit [20], (ii) the maximization of network revenue [21] or (iii) the achievement of a desirable fairness across the network slices of different tenants and their associated users [18].

Different mathematical methods are used. A game theoretic approach is often adopted; in [22], where the authors propose a network slice game as a non-cooperative game, with existence of a Nash equilibrium under appropriate hypothesis; in [19] the network slicing game is a cooperative game, but solved with a distributed algorithm not to force the mobile network operator to reveal private information; in [21] and [23], where the resource allocation is the outcome of an auction. Instead, a utility optimization approach, with different objectives, is used in [17], [18] and [20].

Let us position our work with respect to these works.

- We consider multi-resource allocation problems in order to provide end-to-end network slices, and not only spectrum sharing problems as in [18], [19].
- We provide distributed algorithms, i.e., the decision-making is the result of a distributed set of computations at different nodes, in order to let each resource provider play a role in the slice provisioning, differently from the above-mentioned works.

[2]The considered (weighted) satisfaction vectors are x, $ds \cdot x$, ps and $ds \cdot ps$ where x is the classical satisfaction, i.e. the percentage of resource allocated to a user, $ds \cdot x$ is the weighted classical satisfaction that takes inspiration from the DRF allocation rule, ps is player satisfaction the correct way to measure the satisfaction in full knowledge environment [13],[14] and $ds \cdot ps$ is the weighted player satisfaction considering the dominant share to weight the ps satisfaction.

- Each network slice has a resource demand vector for each resource that has to be allocated, similarly to [23].

III. Distributed multi-resource allocation

We propose three algorithmic approaches to address the multi-resource allocation problem in a distributed way. The first two follow a cascading behavior while the third one exploits parallelism. In this section we analyze the slice resource allocation process under the three approaches, describing advantages and disadvantages of each one. Note that while in this section and the following one we focus on static distributed resource allocation approaches, in Section V we develop and evaluate dynamic scheduling variants.

A. Problem modelling

Let $N = \{1, ..., n\}$ be the set of tenants, $M = \{1, ..., m\}$ be the set of available resources and $P = \{1, ..., p\}$ be the set of resource providers. We realistically consider $|P| \leq |M|$, that is, an ex-ante per-resource provider competition may have taken place beforehand to select the set of providers involved in the slice resource allocation and provisioning. It follows that in our problem we have only one provider for each resource, and each of them can provide more than one resource. The allocation problem is represented as a triplet (D, R, γ), where D is a $n \times m$ matrix with d_{ij} equal to the quantity of resource $j \in M$ demanded by tenant $i \in N$, $R = (r_1, ..., r_m)$ is a vector of positive numbers r_j equal to the amount of each available resource $j \in M$, and γ is a n-dimensional vector containing the priority index of the service required by tenants.

In this work, the priority index γ is linked to the latency of the service. Services requiring low latency have high priority and a low value of γ, those tolerant to higher latency have lower priority and the correspondent value of γ is high. E.g., considering the three classes of service formalized for the 5G, following what is recommended in [24], the importance of latency requirement is high for URLLC services, which refers to wireless connection with low latency, medium for eMBB services, which needs high data bandwidth and moderate latency, and low for mMTC services because they focus on massive objects connectivity, with no strict latency requirements [25]. For this reason, at first instance, we consider 3 priority levels characterizing the three reference 5G classes of services.

Another important aspect to model in network slice resource allocation is the relation between allocated resources. As already assumed in previous works [6], [17], [26], the relationship between resources can be safely approximated with a linear one; this means that if a user asks for 10 Gbps, 40 CPU and 160 GB and it receives only 5 Gbps then the cloud resource provider has to allocate 20 CPU and 80 GB because if the allocation is superior, the cloud resource is wasted, while if inferior, the link resource is wasted.

Let the allocation outcome be represented by a matrix A with components $a_{ij} = d_{ij} \cdot x_i$ where $x = (x_1, ..., x_n)$, $0 \leq x_i \leq 1 \ \forall i \in N$, is the vector of the percentage of demand allocated to each tenant. The allocation is not trivial if it exists a resource $j \in M$ such that $\sum_{i=1}^{n} d_{ij} > r_j$ because the resource is not sufficient to fully allocate the demands of the users (i.e. the resource is congested - in the trivial case the resource provider can allocate the demand and $x = (1, \cdots, 1)$). The three algorithms proposed in next subsections take into account that resources can be congested.

Algorithm 1 Priority-aware allocation rules logic

Input: R, D, N, M, γ
Output: x

for $pr = 1 : s$ **do**
  S=set of user with $\gamma = pr$
  Q=set of user with $\gamma > pr$
  if $\sum_{i \in S} d_{ij} \leq r_j, \forall j \in M$ **then**
    x=ones($|S|$)
  else
    x_S= solution of the selected allocation rule
    x_Q=zeros($|Q|$)
    exit for loop
  end if
end for

The congestion level (μ) of a resource provider p is defined as the ratio between the sum of the demands for its resource(s) and the available quantity of resource(s), i.e. $\mu_j = \frac{\sum_{i=1}^{n} d_{ij}}{r_j}$, if it provides only one resource $j \in M$. Contrarily, if it provides more than one resource, the congestion level is the maximum between the level of each resource it provides. If $\mu_p > 1$ the resources provided by the provider p are congested. We can notice that the congestion level of a resource provider is used in the proposed algorithms to avoid or reduce the sharing of sensible information by resource providers, e.g., quantity of available resources.

Example 1. Let us consider the problem (D, R, γ) with $R = (100, 30, 600, 80)$ and $D = \begin{bmatrix} 20 & 10 & 160 & 40 \\ 20 & 25 & 488 & 64 \\ 30 & 10 & 160 & 40 \end{bmatrix}$ and $\gamma = (1, 1, 1)$. The resources are radio (in number of resource blocks RB), link (in Gbps), RAM (in GB) and CPU (in number of vCPU). The resource providers are 3: radio, network link and cloud providers.

The congestion level is: $\mu_1 = \frac{20+20+30}{100} = 0.7$, $\mu_2 = \frac{10+25+10}{30} = 1.5$, $\mu_3 = max\{\frac{160+488+160}{600}, \frac{40+64+40}{80}\} = 1.8$.

Each resource provider has to take into account the priority index so that the allocation rule described in Section II has to be adapted to the context. In this work we consider a simple algorithm (Algorithm 1) to adapt the allocation rules. We suppose that the priority index takes integer value from 1 to s, where s is the priority index of the lower priority required service, and lower value of γ corresponds to higher service priority. Algorithm 1 firstly attempts to fully satisfy users with the highest priority, and then the ones with lower priorities. Note that, alternatively, any other algorithm providing Pareto efficient solution could be used as well.

B. Cascading Resource Allocation (CRA)

The first algorithm we propose follows a cascading approach, i.e., each resource provider sends to the following one the information about its allocation, and passing through all the providers the allocation is adjusted taking into account the

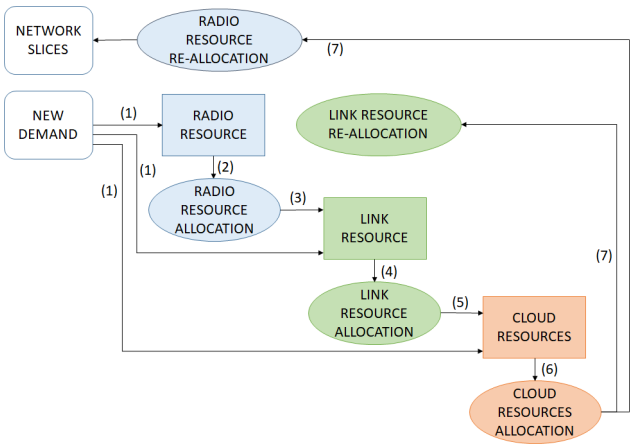


Fig. 3: CRA algorithm.

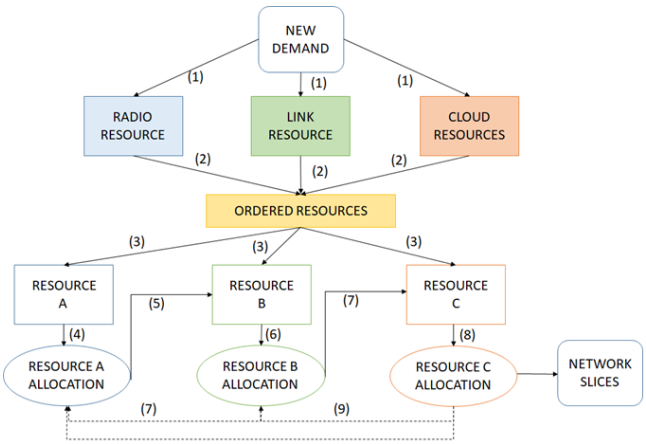


Fig. 4: OCRA algorithm. The steps not always necessary are drawn in dashed line.

Algorithm 2 CRA

Input: R, D, N, M, γ, P
Output: x

Each provider $k \in P$ receives D_k, i.e. the submatrix of demands for the resources provided by k
The first provider calculates x
The first provider sends x to the second provider
for $k = 2 : p$ **do**
  if $\sum_{i=1}^{n} d_{ij} x_i > r_j$, for at least one j provided by k **then**
    Provider k updates x (old values of x_i are upper bound)
  end if
  if $k < p$ **then**
    Provider k sends x to provider $k+1$
  else
    Provider k sends x to all the other providers
    Providers $1, ..., p-1$ reallocate the resources
  end if
end for

Algorithm 3 OCRA

Input: R, D, N, M, γ, P
Output: x
Each provider $k \in P$ receives D_k, i.e. the submatrix of demands for the resources provided by k
Each provider $k \in P$ calculates the congestion level μ_k and sends it a multi-domain orchestrator
Each provider $k \in P$ receives the new order of the resources calculated by the multi-domain orchestrator
The first provider calculates x
The first provider sends x to the second provider
for $k = 2 : p$ **do**
  if $\sum_{i=1}^{n} d_{ij} x_i > r_j$, for at least one j provided by k **then**
    Provider k updates x (old values of x_i are upper bound)
    Provider k sends x to all the $k-1$ previous providers
    Providers $1, ..., k-1$ reallocate the resources
  end if
  if $k < p$ **then**
    Provider k sends x to provider $k+1$
  end if
end for

congestion level of each resource and the allocation proposed by the previous provider. Algorithm 2 describes the CRA approach for the general case with p providers; the steps of the algorithm are labeled within parenthesis. The order follows the natural end-to-end order of the resources, as for the radio-link-cloud three resources scenario presented in Figure 3.

We consider now and later as reference scenario for the examples the one with 3 resources providers ($P = \{1, 2, 3\}$) providing radio, link and cloud resources. To make the notation clearer, in this case we use the subscript r for radio ($p = 1$), l for link ($p = 2$) and c for cloud ($p = 3$).

Example 2. Let us consider the same problem (D, R, γ) of Example 1. The algorithm's steps are:

(1) The radio resource provider receives the demand vector $(20, 20, 30)$, the link resource provider receives the demand vector $(10, 25, 10)$ and cloud resource provider receives the demand matrix $\begin{bmatrix} 160 & 40 \\ 488 & 64 \\ 160 & 40 \end{bmatrix}$.
(2) The radio resource provider calculates the allocation. In this case there is no congestion so $a_r = (20, 20, 30)$ and $x_r = (1, 1, 1)$.
(3) The link resource provider receives the vector x_r.
(4) The link resource provider calculates the allocation. In this case there is congestion so x_r is not an admissible solution. The provider uses an allocation rule; if it is for example the MMF one, the allocation is $a_l = (10, 10, 10)$ and $x_l = (1, 0.4, 1)$.
(5) The cloud resource provider receives the vector x_l.
(6) The cloud resource provider checks if x_l is admissible:
  - $160 \cdot 1 + 488 \cdot 0.4 + 160 \cdot 1 \overset{?}{<} 600 \rightarrow$ yes
  - $40 \cdot 1 + 64 \cdot 0.4 + 40 \cdot 1 \overset{?}{<} 80 \rightarrow$ no

  Due to the fact that the proposed x_l is not admissible; the cloud provider calculates a new allocation, taking into account that for each user i the upper bound for x_{c_i} is x_{l_i}. For example using the DRF rule we get $x_c = (0.68, 0.4, 0.68)$ and $a_c = \begin{bmatrix} 108.8 & 27.2 \\ 195.2 & 25.6 \\ 108.8 & 27.2 \end{bmatrix}$.
(7) The cloud resource provider sends the vector $x_c = (0.68, 0.4, 0.68)$ to the link and radio resource providers that re-allocate the resources obtaining $a_r = (13.6, 8, 20.4)$ and $a_l = (6.8, 10, 6.8)$.

C. Ordered Cascading Resource Allocation (OCRA)

Algorithm 4 PRA

Input: R, D, N, M, γ, P
Output: x
Each provider $k \in P$ receives D_k, i.e. the submatrix of demands for the resources provided by k
Each provider $k \in P$ calculates the value of x
Each provider sends the allocation vector to each of the other providers
A consensus algorithm provides the final value of x

In the presence of a multi-domain orchestrator that is able to schedule how resource allocation takes one can partially avoid resource re-allocation. Before each decision is taken, the orchestrator asks or receives the congestion level of the resources provided by the p providers and re-order the providers. This cannot guarantee to bypass the re-allocation for all resources, but it can strongly reduce its impact on the solution (see Example 3). The algorithm is described by Algorithm 3, and is depicted in Figure 4 for the radio-link-cloud scenario.

Example 3. Let us consider the same problem (D, R, γ) of Example 1. The algorithm's steps are:

(1) Each resource provider receives the demand vector/matrix.
(2) Each resource provider calculates the congestion level: $\mu_r = 0.7$ $\mu_l = 1.5$, $\mu_c = 1.8$.
(3) The multi-domain orchestrator sends the resources order to the resource providers. The first resource to be allocated is the cloud followed by the link and the radio.
(4) The cloud resource provider calculates the allocation, for example using the DRF: $x_A = (0.67, 0.412, 0.67)$, $a_c = \begin{bmatrix} 107.2 & 26.8 \\ 201.1 & 26.4 \\ 107.2 & 26.8 \end{bmatrix}$
(5) The link resource provider receives the vector x_A.
(6) The link resource provider checks if x_B is admissible: $10 \cdot 0.67 + 25 \cdot 0.412 + 10 \cdot 0.67 \overset{?}{<} 30 \rightarrow$ yes.
The allocation is: $x_B = x_A$, $a_l = (6.7, 10.3, 6.7)$.
(7) The radio resource provider receives the vector x_B.
(8) The radio resource provider accepts the proposed x_B because the resource is not congested. The allocation is: $a_r = (13.4, 8.24, 20.1)$.
(9) Step not necessary because no resource re-allocation.

It is worth noting that with this algorithm one cannot always avoid re-allocation. In fact if, in Example 3 we increase the value of d_{22} from 25 to 30, the order of the resource, based on the congestion level, remains the same ($\mu_l = 1.67$), but if the cloud provider proposes the allocation $x_A = (0.2, 1, 0.2)$, the link provider cannot accept it because $10{\cdot}0.2{+}30{\cdot}1{+}10{\cdot}0.2 > 30$. This shows that the re-allocation is not always avoided with this algorithm, but at least its negative impact is decreased.

D. Parallel Resource Allocation (PRA)

In the previous proposed algorithms the computation of the resource allocation is done following a weakly distributed manner. Indeed, the resource allocation is computed according to a defined sequence among the resource providers, which implies a high dependency and a low collaboration degree between providers. Thus, the computation time required by these algorithms is dominated by the resource provider with the highest response time. To limit the impact of such a situation, we design a fully-distributed algorithm, named Parallel Resource Allocation (PRA), which allows to increase the level of parallelism to compute the allocation and to reduce the computation time. Contrary to the two preceding algorithms, the idea of the PRA approach is to allow each provider to compute its own allocation, then all the resource providers exchange their allocation and use a distributed consensus approach [27] to obtain the final allocation.

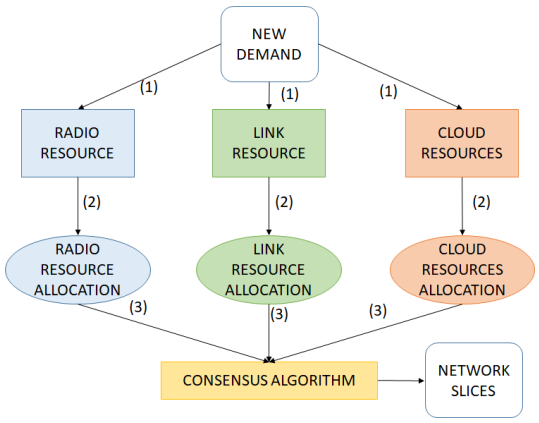


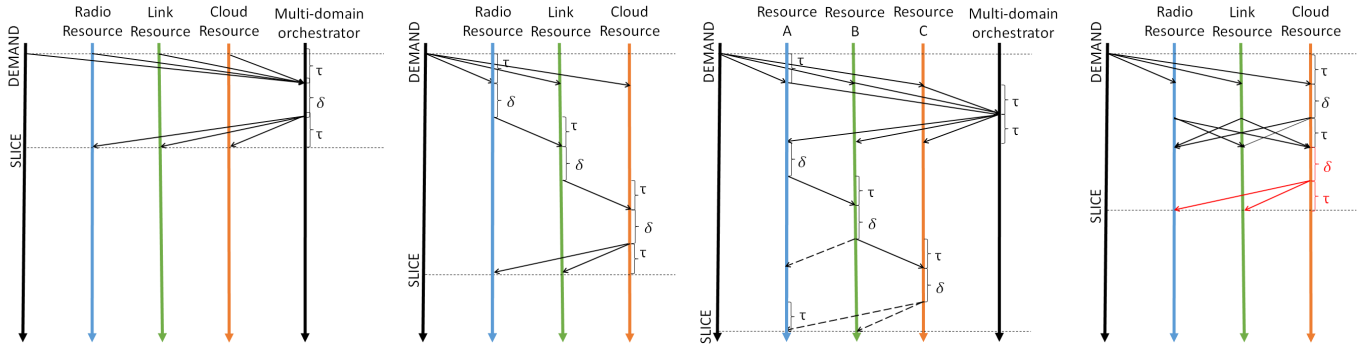
Fig. 5: PRA algorithms; PRA-1 using the 1-phase consensus algorithm, PRA-2 using the 2-phase consensus algorithm.

PRA is described in Algorithm 4, and is depicted in Figure 5 for the radio-link-cloud scenario.

We develop the PRA idea with two different consensus algorithm variants. The first one has the property of being fast, but it does not guarantee to saturate at least one of the congested resources, so it is not Pareto efficient as we prove later (Section IV-B). The second one introduces an additional information exchange to the process, but it guarantees to saturate at least one of the congested resources.

The first consensus algorithm, called PRA-1, is a 1-phase algorithm; each resource provider diffuses to all the other ones the value of x, and the allocations are obtained in the following way: $(\min_{k \in P} x_{k_1}, \cdots, \min_{k \in P} x_{k_n})$. The non-saturation of the resources can happen when there exists at least one user for which the dominant resource, i.e., the resource in percentage most requested by the user, is not the one with higher congestion level (see Example 4).

The second algorithm, called PRA-2, is a 2-phase algorithm; each resource provider diffuses (i) the value of the allocation, (ii) the congestion level and (iii) the resource share of each resource it provides for each user, i.e., $rs_i = \{\frac{d_{ij}}{r_j}\}\ \forall i \in N$ and for each resource j it provides. These information are used in order to limit the exchange of sensible information for resource providers (e.g., quantity of available resource). The provider with the most congested resource can identify itself and calculate the value of x using a multi-resource approach. In fact, the information about the resource share allows the provider



(a) Centralized (b) CRA (c) OCRA (d) PRA

Fig. 6: Involved signaling for the centralized algorithm and the proposed distributed ones, for the radio-link-cloud scenario, as a function of time, under the hypothesis of upper bound on transfer times (τ) and equal allocation computing times (δ). The dashed arrows indicate not necessary steps, and the red arrows correspond to extra steps of the 2-phase consensus algorithm.

to take into account the capacity constraints; moreover the optimization objective is decided by the provider following its fairness goal. We can notice that the calculus of the allocation by each provider is thus not necessary and can be avoided to decrease the delay budget. PRA-2 guarantees a Pareto efficient allocation as proven later.

Example 4. Let us consider the problem (D, R, γ) of Example 1. The value of x calculated in a parallel way is x_r=$(1, 1, 1)$ for the radio resource, x_l=$(1, 0.4, 1)$ for the link resource using the MMF allocation rule and x_c=$(0.67, 0.412, 0.67)$ for the cloud resource, using the DRF allocation rule.

Using the PRA-1, each resource provider allocates the resources using $x = (0.67, 0.4, 0.67)$. The allocations are: $a_r = (13.4, 8, 20.1)$, $a_l = (6.7, 10, 6.7)$, $a_c = \begin{bmatrix} 107.2 & 26.8 \\ 195.2 & 25.6 \\ 107.2 & 26.8 \end{bmatrix}$ and the resource used is $(41.5, 23.4, 409.6, 79.2)$. This shows that the saturation of the resources is not guaranteed when we use the 1-phase algorithm. In fact, for user 2 the dominant resource is the link resource but the resource with higher congestion level is the cloud one.

With PRA-2, the three resource providers diffuse the following information:

- $x_r = (1, 1, 1)$, $rs_r = (0.2, 0.2, 0.3)$, μ_r=0.7.
- $x_l = (1, 0.4, 1)$, $rs_l = (0.33, 0.83, 0.33)$, μ_l=1.5.
- $x_c = (0.67, 0.412, 0.67)$, $rs_c = (0.5, 0.81, 0.5)$, μ_c=1.8.

The cloud resource is the most congested one and the cloud provider calculates the value of x. For example, if it chooses to use a proportional approach (equalizing the x of each tenant), the solution is $x = (0.556, 0.556, 0.556)$, $a_r = (11.12, 11.12, 16.68)$, $a_l = (5.56, 13.9, 5.56)$, $a_c = \begin{bmatrix} 89 & 22.2 \\ 271.3 & 35.6 \\ 89 & 22.2 \end{bmatrix}$.

IV. Performance evaluation

In this section we provide a qualitative and quantitative analysis of the proposed algorithms. Section IV-A provides an analysis in terms of delay budget, Section IV-B highlights advantages and disadvantages of each algorithm, and in Section IV-C we numerically compare the algorithms.

Algorithm	Best case	Worst case	Message complexity
Centralized	$2\tau + \delta$	$2\tau + \delta$	$2p + 1$
CRA	$(p+1)\tau + \delta$	$(p+1)\tau + p\delta$	$3p - 2$
OCRA	$(p+2)\tau + \delta$	$(p+3)\tau + p\delta$	$[4p - 1, \frac{p^2+7p}{2} - 1]$
PRA-1	$2\tau + \delta$	$2\tau + \delta$	p^2
PRA-2	$3\tau + 2\delta$	$3\tau + 2\delta$	$p^2 + p - 1$

TABLE I: Delay budget and message complexity - General case with p resource providers.

A. Delay budget

We are here interested in estimating the delay budget of each algorithm, i.e., the global time between the submission of a slice demand and the moment in which the computation of the resource allocation is done. Delay contributions in the process are the transfer time covering the transmission delay and the propagation delay for each message and the allocation computation time. Time for checking if an allocation x is admissible can be considered negligible. We also assume in the following the transmission delay to be negligible, given the likely short message size in stake, so the transfer delay corresponds to the propagation delay.

Figure 6 shows delay budget diagrams for the three proposed algorithms, and an arbitrary centralized approach where a multi-domain orchestrator receives the tenants demand and computes the allocation as a one-shot operation. Under the simplification that propagation delays are all roughly equal to a value τ and all allocation computing times are equal to δ, we obtain the estimation of the delay budget in Table I for the general case with p resource providers. We report the value of the delay budget in the best and worst case; these two values do not coincide in case of cascading approaches: the best case is the one in which only one allocation is calculated and is admissible for all the other resource providers, while the worst one is in case an allocation has to be calculated by each resource provider.

Clearly the centralized approach is the one with lower delay budget together with the first distributed approach. The algorithm closer to the centralized approach is PRA-1. Cascading approaches have a higher figure; note that while for the centralized and PRA approaches the value of delay budget does not depend on the number of resource providers p, for cascading approaches it does. Figure 7 compares the delay

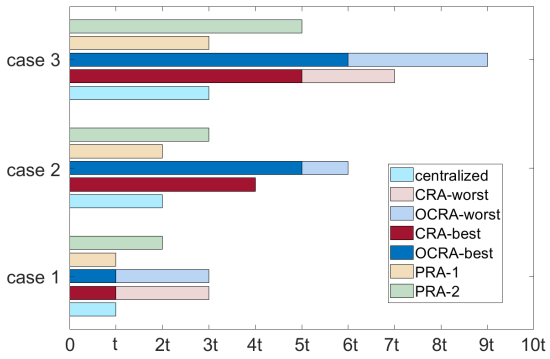


Fig. 7: Comparison of delay budgets with $p = 3$. Case 1: $\tau \ll \delta$, $t = \delta$. Case 2: $\delta \ll \tau$, $t = \tau$. Case 3: $\tau = \delta = t$.

Algorithm	Advantages	Disadvantages
Centr.	Low delay budget	Multi-domain orchestrator High confidentiality disclosure
CRA	No multi-domain orcherstrator	Re-allocation
OCRA	Rarely re-allocation	High delay budget Multi-domain orchestrator
PRA-1	No multi-domain orchestrator Low delay budget Independent radio allocation	Pareto efficient solution not guaranteed High message complexity
PRA-2	No multi-domain orchestrator Low delay budget Independent radio allocation	High message complexity Low confidentiality disclosure

TABLE II: Pros vs cons of studied algorithms.

budget of all the approaches with 3 resource providers, in three different cases: (case 1) the propagation delay is negligible with respect to the computing time, i.e., $\tau \ll \delta$, (case 2) the reverse case, i.e., $\delta \ll \tau$ and (case 3) the two times are comparable - we plot the case $\tau = \delta$.

B. Pros and cons

Let us draw advantages and disadvantages of the different algorithms. Table II summarizes the following observations.

Choosing a centralized approach we have the advantages of a low delay budget in the creation of the slice, due to the fact that the decision is taken atomically, by a single entity. Meanwhile the fact of having centralization at a multi-domain orchestrator can be seen as an obvious drawback in terms of reliability and scalability from the one hand, and confidentiality from the other hand as each provider has to share possibly sensible information, as for example the quantity of resource available in its domain. The presence of a multi-domain orchestrator is also necessary for the OCRA approach, to order the resource providers. In this case it has only a function of dispatcher (note that for OCRA it is however possible to avoid the presence of the multi-domain orchestrator using a distributed approach to exchange the information about the resources congestion level, however impacting performance). All the other approaches have the advantage of not needing such a centralized orchestrator.

Concerning cascading approaches (CRA, OCRA) they have the disadvantage of re-allocating resources during the slice provisioning; this is expected to be highly reduced with the OCRA approach.

Allocation rule	No re-allocation	One re-allocation	Two re-allocation
MMF	82.7%	17%	0.3%
Mood value	100%	0%	0%
Proportional	100%	0%	0%

TABLE III: Occurrence of re-allocations with the OCRA algorithm using common single-resource rules.

Allocation rule	Percentage of non-optimal solutions
MMF	57%
Mood value	72%
Proportional	56%

TABLE IV: Percentage of non-Pareto efficient solutions using the PRA-1 algorithm.

Advantages of parallel approaches are (i) the low delay budget, due to the simultaneously computation of the allocation and diffusion of the information, and (ii) the possibility to independently allocate some resources. For example, this can be useful for the radio resource for which the hypotheses of linear dependency with the other resources may appear less acceptable with some radio scheduling protocols.

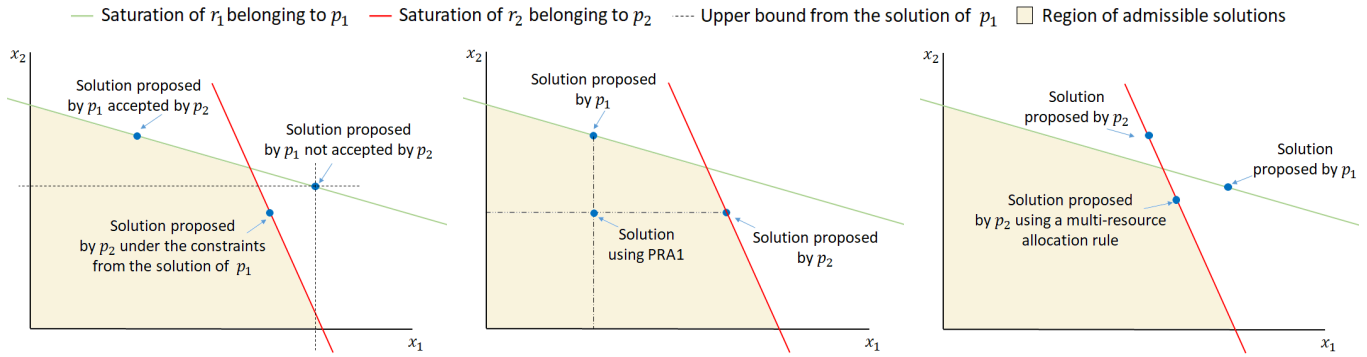
If distributed approaches have good behavior in terms of delay budget compared to the cascading ones, considering the number of messages that have to be exchanged, the judgment is reversed. From Table I, we can see that the number of exchanged messages grows quadratically with the number of providers p. In case in which $p = 10$ the number of the exchanged messages is between 21 and 30 for the centralized and cascading approaches, while it is 100 and 109 for the two distributed ones. This is the price to pay when we distribute the calculus of the allocation to avoid a single point of failure.

Among the disadvantages, for the PRA-2 we find the low confidentiality disclosure; in fact, providers are forced to exchange much more information compared to the PRA-1 and the centralized approaches. A drawback for the PRA-1 approach is the possibility to get a solution that is not Pareto efficient. In this respect, we can state the following Theorem.

Theorem 1. CRA, OCRA and PRA-2 algorithms provide Pareto-efficient solutions.

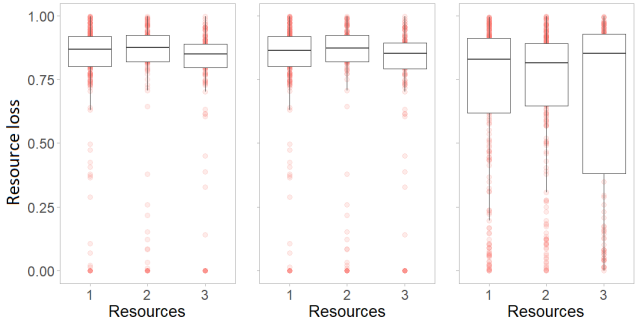
Proof. CRA and OCRA algorithms provide Pareto-efficient solutions because the allocation coincides with the one proposed by one provider that selects a Pareto efficient allocation rule. The PRA-2 algorithm provides an allocation that saturates one resource because it is calculated using a multi-resource allocation rule, so it provides a Pareto efficient solution. The algorithm PRA-1, using the minimum value for each component allows the increasing of the allocation of one tenant without decreasing the one of the other. Let us consider the example 4. If we increase the allocation of tenant 2 from 0.4 to 0.412 we obtain the allocation proposed with the OCRA in Example 3. Thus, it is possible to increase the allocation of tenant 2 without modifying the one of the others (allocation not Pareto efficient). □

Figure 8 shows an example with two tenants and two resources provided by two different providers. We can see how algorithms works and that the Pareto efficiency is guaranteed for the cascade approaches and for the PRA-2 algorithm, while it is not for the PRA-1 algorithm because the solution is inside



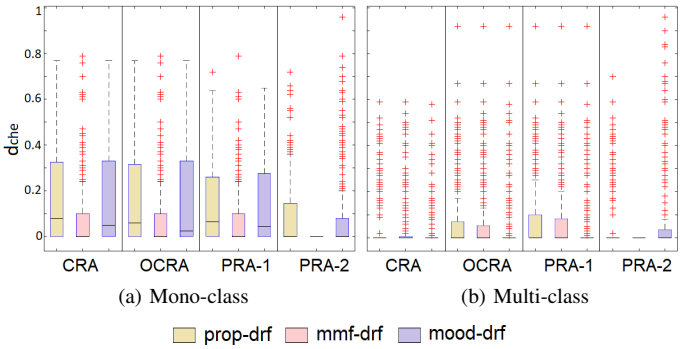
(b) CRA - OCRA (c) PRA-1 (d) PRA-2

Fig. 8: Pareto efficiency for an example with 2 tenants and 2 resources provided by p_1 and p_2. Hypothesis: in the OCRA plot p_1 more congested than p_2 and in the PRA-2 plot p_2 more congested than p_1.



(a) MMF (b) Proportional (c) Mood value

Fig. 9: Percentage of resource loss.



(a) Mono-class (b) Multi-class

Fig. 10: Chebyshev distance.

the region of the admissible solutions and not on the Pareto frontier.

C. Numerical analysis

We present a numerical analysis to measure (i) the occurrence of reallocation using the OCRA algorithm, (ii) the occurrence of inefficient solutions for the PRA-1 algorithm, and (iii) the distance of the proposed distributed approaches from the centralized one. The analysis for (i) and (ii) is done considering services with the same priority. In our knowledge there are no previous works considering a congestion scenario so the results show the difference between the 4 distributed approaches and they are compared to centralized ones.

1) Occurrence of re-allocation: The aim here is to understand if there is a real gain using an ordered approach, i.e., if the re-allocation of the resources is reduced and consequently the delay budget induced by allocation computation. We generate 300 problems with three tenants, three resources belonging to three providers, randomly associating a level of congestion between 0.1 and 2 for each provider. Table III shows the results of the simulations when all providers use the same allocation rule (Proportional, Mood value, MMF). We can see that there is a real gain in using an OCRA approach because with the proportional allocation and the mood value we have no re-allocations, while with the MMF there are situations in which one re-allocation is needed, but two are needed only for a negligible number of cases.

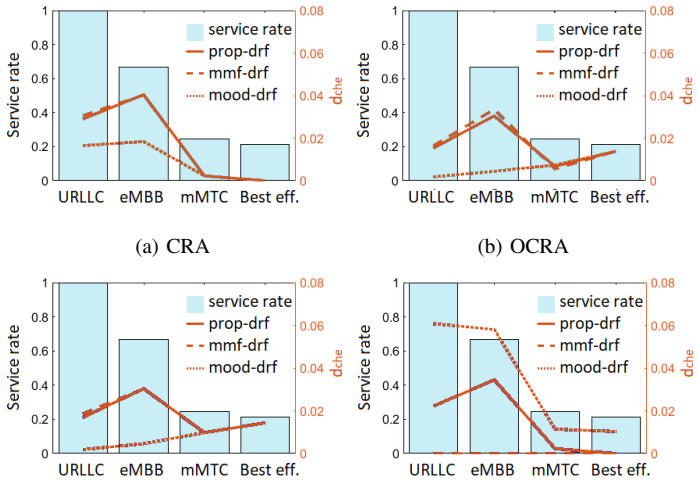
2) Percentage of inefficient solutions in PRA-1: We test here the efficiency of the solutions when we use the PRA-1 algorithm. Using the same data generated for the previous simulations, we calculate the percentage of time in which the PRA-1 algorithm does not provide a Pareto-efficient solution (Table IV) and we estimate how much is the loss for the tenants in term of resources (Fig. 9). Clearly, PRA-1 has high probability to provide allocations that are not Pareto-efficient. When providers use the same allocation rule, more than half of the time the produced allocation is not Pareto efficient. Furthermore the resource loss is high. The median value in percentage, obtained in the boxplot (Fig. 9) belongs to the interval of $[0.8, 0.9]$. We can also observe that the higher resource loss is experienced with the Mood value allocation.

3) Distance from a centralized approach: We firstly introduce a measure of the distance between a centralized approach and a distributed one. A simple measure we can consider is the Chebyshev distance (or L_∞ metric) defined as follows.

Definition 2. The Chebyshev distance between two vectors y_1 and y_2 is $d_{che} = max_i|y_{1_i} - y_{2_i}|$.

In our case, considering a solution vector obtained with a centralized approach and one with a distributed one, the measure indicates the gain (or loss) of the user that obtains the maximum gain (or loss) when a distributed approach is used. This measure provides an estimation of the satisfaction (unsatisfaction) of the users in adopting a distributed approach.

We developed an ad-hoc simulator in Python that we make available in [33] for verification and reproducibility. We sim-



(a) CRA (b) OCRA

(c) PRA-1 (d) PRA-2

Fig. 11: Chebyshev distance average and service rate.

ulate 200 problems with five tenants, taking inspiration from Amazon EC2 instances [28] (the same considered instances as in [6]); we report in the supplementary materials the detailed instance table: we select those templates with different 'instance type' ('General Purpose', 'Computer Optimized', 'Memory Optimized', 'Accelerated Computing'and 'Storage Optimized') and we consider three resources belonging to two providers (CPU and memory for the cloud provider, link capacity in Gbps for the network link provider), a level of congestion between 0.1 and 1.5 for each provider, and both the case in which the tenants have the same priority and belong to the same class (single-class) and the case in which the tenants have different priorities and belong to different classes (multi-class). In the second case we arbitrarily associate to the different Amazon templates a type of service as follows: URLLC to Accelerated Computing instance, eMBB to Computer and Memory Optimized instance, mMTC to Storage Optimized instance, and best effort to General Purpose instance. The considered services, defined accordingly to 5G standards, are differentiated accordingly to the demand and the priority index.

Figure 10 shows the boxplot of the distance for each algorithm, using different combinations of allocation rules, when the centralized approach uses the DRF rule. When the priority is the same for each tenant there are users that can gain or loose a lot when the providers adopt as distributed approach the CRA, OCRA and PRA-1. In the single-class case, the distance is reduced using the PRA-2 approach because the proposed allocation is calculated as a multi-resource allocation taking into account the information provided by each provider. In the multi-class case we notice a performance improvement of the distributed algorithms. In fact, in this case, the differences emerge only for the group of tenants belonging to the same priority class for which the remaining resource, after that tenants with higher priority are fully served, is not enough. In this case due to the small cardinality of the subset of users belonging to the same class, there is a high probability that the distributed solution is close to the centralized one.

We then consider the distance measure inside each group of services and the service rate, i.e., the percentage of requests processed (Figure 11). The distributed algorithm always serves the users with higher priority and the service rate decreases with the service priority. On average, the distance increases decreasing the service priority, but a decrease of the distance is possible because (i) services with low priority have high probability not to be served both with the centralized and distributed approaches (Figure 11) and (ii) as already said, if the cardinality of the last served group is small the distributed and centralized solutions can be close. We can also observe that: (i) CRA has better behavior in terms of distance than OCRA algorithm for best-effort services, vice-versa for the URLLC and eMBB services, (ii) although PRA-1 is not Pareto efficient, it achieves Chebyshev distances close to the cascade approaches and (iii) PRA-2 algorithm returns allocations which are the closest to the centralized algorithm in terms of distance for the MMF and DRF allocations and it has the worst results when using the mood-DRF allocation.

V. Integration of SLA and Priority-Based Scheduling

In section III we proposed different distributed algorithms to slice the network in a given time frame. We want now to enrich these algorithms with policies taking into account Service Level Agreement (SLA) constraints [1]. In the literature we can find cloud scheduling algorithms that meet tenants' Quality of Service (QoS) requirements. Examples are: (i) the work in [29] where a deadline job scheduling algorithm is proposed considering the current status of the system and the job execution cost model; (ii) the work in [30] proposing the Earliest Deadline First (EDF) that is a dynamic priority real-time scheduling algorithm that considers time constraints of tasks in scheduling for execution and (iii) the work in [31] proposing an algorithm called Earliest Maximal Waiting Time First (EMWTF), that enhances the EDF algorithm. None of these works provide an algorithm able to provide a fair allocation over the time considering all our constraints (SLA constraints, priority, deadline and service time frame).

In this section we discuss (i) how to guarantee the continuity of the service, i.e., when a tenant is served, it has to be served for the required time and (ii) how to guarantee a minimum capacity, i.e., the minimum quantity of resource specified in the contract between the provider and the consumer. The main aim is to guarantee time-fair allocations, i.e., to serve with a similar service availability rate users asking for services of equal priority.

Under this setting, given a time frame t, the resource allocation problem is a tuple $(D_t, D_t^m, \gamma_t, \nu_t, \tau_t, \lambda_t, R_t)$ where:

- D_t is the demand matrix,
- D_t^m is the matrix containing the minimum values of resource to allocate to each tenant defined in the SLA,
- γ_t is a vector containing the priority index of each tenant,
- ν_t is a vector containing the availability rate of each tenant, i.e. the percentage of time the tenant was served, with at least a minimum resource amount,

- τ_t is the number of time frames the demand is for[3]
- λ_t is the deadline, i.e. the number of time frames the tenant can wait before being served,
- R_t is the available resource[4].

We link the priority index to the latency of the service: if the service requires a low latency its priority is higher, otherwise it is lower. For instance URLLC services can be characterized by higher priority indexes compared to eMBB and mMTC services because they are low-latency services.

The structure of the algorithms remains similar but we have to define a users delay policy (V-A), how to allocate the resources, under the constraint of guaranteeing a minimum share of resource (V-B) and how the cascade (V-C1) and parallel algorithms work (V-C2). Finally we propose a SLA and deadline priority algorithm (V-D).

A. User delay policy

When a provider p is not able to satisfy the minimum allocation of each tenant ($\sum_{i=1}^{n} d_{ij}^m > R_j$, for at least one resource j provided by p), it has to remove one or more tenants, putting them in hold for possible servicing in the next time slot. Different user removal/delay policies can be proposed. Here we propose one that takes into account both the user priority index and the current availability rate of each tenant, already used in [32], for its desirable property of guaranteeing fairness over the time.

The idea is that, to order users, we should firstly consider the priority index. Tenants with high priority have to be removed only if the tenants with lower priority do not exist or are already been removed. This can match operational constraints because high priority services are the ones asking for low latency, and tenants asking for this type of services have to be served as fast as possible.

One time users are ordered depending on the priority index, the removal/delay policy considers the value of ν. The highest is the value of ν, the highest is the percentage of time the tenants were served in past time slots. It follows that users with same priority have to be ordered by descending values of ν, in order to guarantee a sort of time-fairness. An example of users re-ordering, considering the described policy is given in Example 5.

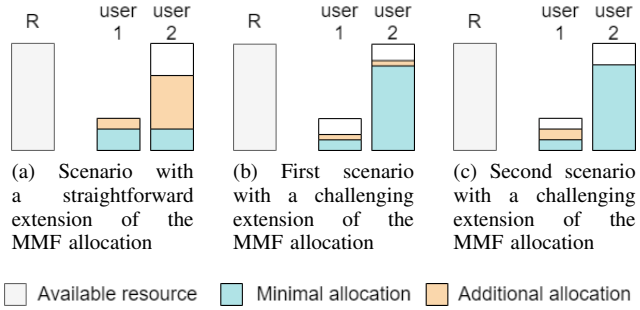
Example 5. Let us consider five tenants with priority vector $\gamma = [1, 2, 1, 3, 1]$, where lower value of γ means higher priority, and availability vector $\nu = [0.1, 1, 0.5, 0.9, 0.3]$. The ordered users vector, following the policy proposed is $[4, 2, 3, 5, 1]$.

The users re-ordering has to be done by each provider at the beginning of each time frame and in case the expiration time of a tenant falls in currently time slot, providers automatically set the priority of the user as the highest one, trying to avoid the delaying of the tenant (as further explained in Section V-D).

Once the order of the users to remove is established, each provider tests if the minimum of the resources it provides can be allocated, if not it excludes a user following the order described above and it checks if in the group of the excluded one there is one or more one that can be re-introduced. The users' re-introduction order is inverse with respect to the elimination one.

[3]We consider that during the service time frame τ the demand of the user is fixed. In practice, it can change across service time frames so as to take into account dynamic resource demands.

[4]We consider the available resource as fixed in a given time frame, knowing that the resource available could also vary at different service time frames, to cope for the fact that externalities can change the actual available resource at a given service time frame.



(a) Scenario with a straightforward extension of the MMF allocation

(b) First scenario with a challenging extension of the MMF allocation

(c) Second scenario with a challenging extension of the MMF allocation

Fig. 12: Different resource allocation scenarios. The MMF extension is challenging, for example, if users with high demands have high minimal demands as in (b) and (c). The resource left after the allocation of the minimal demand has to be divided equally between users (b) or it has to be allocated to the one with smaller demand (c).

B. Resource allocation with minimum capacity constraints

We now discuss how resource allocation rules can be modified to take into account the user's minimum demand.

Concerning the rules obtained as solutions of optimization problem, it is sufficient to add a constraint regarding the value of the percentage of resource to allocate x, or alternatively on the value of the minimal allocation. For single-resource allocations (weighted proportional, α-fairness) we can modify the demand boundness constraints ($a_i \leq d_i$) adding a lower bound: $d_i^m \leq a_i \leq d_i$; for multi-resource allocation the boundness constraints ($x_i \leq 1$) becomes $x_i^m \leq x_i \leq 1$ with $x_i^m = max_j\left(\frac{d_{ij}^m}{d_{ij}}\right)$, when each provider considers the resources j it provides.

The re-definition of the mood value and the MMF allocation rules, in case of minimal demand to guarantee, is less intuitive. The MMF allocation is not solvable as an optimization problem and it is not clear how the problem with the minimal demand to satisfy can be solved. In particular Figure 12 shows how, using the hydraulic interpretation of the MMF allocation, the re-definition problem is challenging. This open problem represents an interesting possible future work.

The mood value instead can be interpreted as the results of an optimization problem [13], [14], after that the minimal allocation, calculated considering the other users demands, is allocated to each user. Under the hypothesis that the sum of the minimal demands is inferior to the available resource (always respected otherwise we remove one or more users), the mood value allocation can be calculated as follows.

Definition 3. Let (d^m, d, R) be an allocation problem, where d^m is the vector of the minimal demands such that $\sum_{i=1}^{n} d_i^m \leq$

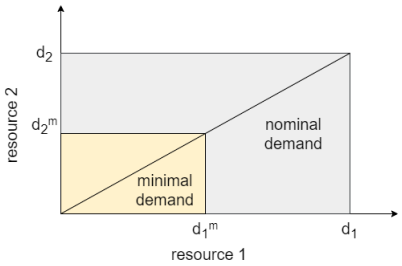


Fig. 13: Bi-dimensional representation of a nominal and minimal demand for two resources, when the linear dependency between resources is the same for the two types of demands.

R, then the *Refined mood value* is given by $a_i^m = min_i^* + m^*(max_i - min_i^*)$, where: $min_i^* = \max\{d_i^m, min_i\}$, $max_i = \max\{d_i, R\}$, $m^* = \frac{R - \sum_{i=1}^n min_i^*}{\sum_{i=1}^n max_i - \sum_{i=1}^n min_i^*}$.

C. Cascade and parallel algorithms enhancements

Before adapting the two types of algorithms proposed in section III for long term allocations, let us further elaborate on a hypothesis of our problem. We assume that the minimal demand of each user maintains the same linear dependence between the resources expressed by the nominal demand. As we can see from Figure 13, this implies that for each tenant the portion of the nominal demand demanded as minimal demand is the same for each resource. Consequently, each user, instead of submitting the minimal demand for each resource, can simply submit the demand ratio vector needed to receive the minimal allocation for each resource. Thus, alternatively to the demand matrix D^m, we can consider a vector d^m, where each component gets value between 0 and 1 and that represents the percentage of demand each user requires as minimal one[5].

We are now ready to describe how cascade and parallel algorithms work at each instant of time, taking in consideration the user delay policy and the guarantee on the minimal allocation. In fact, differently from the classical algorithms, now, when a provider proposes an allocation it takes into account the minimal allocation, hence it uses allocation rules described in V-B. We first present hereafter the enhanced algorithm, and then we explain the general algorithm over the time, considering the redefined cascade and parallel algorithms and the deadline priority policy.

1) Cascade algorithms (sCRA, sOCRA): The two enhanced cascade algorithms, we name sCRA and sOCRA, work similarly to the static case but, in order to provide feasible allocations, providers need to know which users can be served by every provider. This can be done by adding a pre-processing step at the beginning of the classical CRA algorithm, or while providers send their congestion for the OCRA algorithm.

If we do not add the pre-processing step in the sCRA algorithm the budget-delay can excessively increase because each time a provider k delays a user, served by the previous ones, this information has to be sent to the first provider that has to update the allocation and consequently each of the following provider had to re-calculate the allocation. Adding this preliminary step, the sCRA algorithm has the same delay-budget of the sOCRA and, as the sOCRA, it needs the presence of a multi-domain orchestrator. Consequently the sCRA loses all the advantages over the sOCRA and therefore we consider only the sOCRA algorithm next.

2) Parallel algorithms (sPRA-1, sPRA-2): In parallel algorithms the notion about which user is served and which is delayed is learned after that each provider diffuses the allocation. In fact if a user cannot be served, the allocation is zero. It follows that sPRA1 and sPRA2 work as the classical parallel algorithms.

[5]In practice, the minimum service to be guaranteed is established by the SLA, which is a contract between the resources providers and the users so it is possible that, for different resources, different minimum values are negotiated.

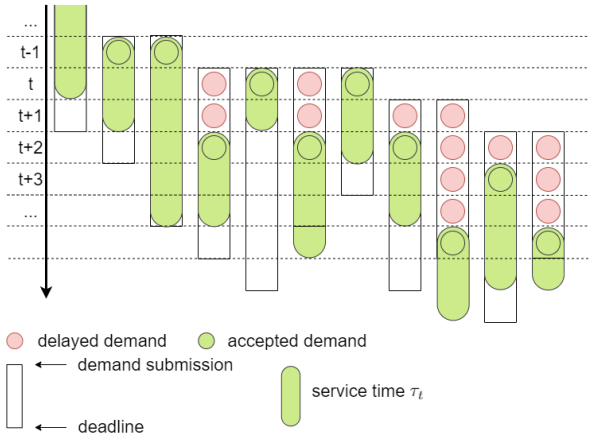


Fig. 14: Example of SLA-deadline priority algorithm dynamic.

service time frame τ	URLLC	$1+(1-0.2)^{x-1}0.2$
	eMBB	$1+(1-0.01)^{x-1}0.01$
	mMTC	$1+(1-0.8)^{x-1}0.8$
	Best effort	$1+(1-0.01)^{x-1}0.01$
Deadline λ	URLLC	2
	eMBB	100
	mMTC	50
	Best effort	100
Priority γ	URLLC	1
	eMBB	2
	mMTC	3
	Best effort	4

TABLE V: Simulation settings.

D. SLA and deadline priority algorithm

We here describe how to integrate the re-defined cascade and parallel algorithms with a general SLA and deadline priority scheme.

At each instant of time t the algorithm can be decomposed in the following phases. Let N_a be the set of users that have started to be served at time t, N_n the set of new tenants submitting their first demands at time t and N_d the set of users whose demands were delayed at time $t-1$ [6].

1) *Resource availability update*: a group of N_a users with demand matrix and minimal demand matrix respectively D_a and D_a^m has to be served because accepted in a previous instant of time, with a service time frame still not expired. Each resource is consequently updated as follows $R_j^* = R_j - \sum_{i \in N_a} D_{a_{ij}}^m, \forall j \in M$, i.e., considering

[6]Note that we avoid the subscript t for the sake of simplicity.

that at least a minimal demand for each user has to be guaranteed. This is possible if the available resource of the provider does not change, due to the fact that in the previous instant of time $t-1$ demands were served – in some cases also with other users undergoing service expiration at time $t-1$[7].

2) *Distributed allocation*: Cascade or parallel algorithms described in section V-C1 and V-C2 provide the distributed allocation. The user delay policy is considering only users delayed in a previous step N_d and the new group of tenants N_n arriving at instant t.
3) *Demand, priority and availability rate update*: After the resource is allocated:
   - the users with the service expired at time t are removed from N_a,
   - the users served for the first time at the t instant are added to N_a,
   - the users submitting a demand at t but not served are added to N_d.

In order to try to serve also users with low priority, if the deadline of a tenant demand falls at time $t+1$, the priority is increased to the highest one[8]. In this way the tenant has higher probability not to be delayed and consequently served before the expiration of its request. If, despite the increased priority, the tenant cannot be served, its availability rate is updated.

Figure 14 gives an example of algorithm dynamics. We can notice that each user can be accepted or delayed in the service time frame between the demand submission and the deadline, once a demand of a tenant is accepted it is served for a number of instants of time equal to the service time frame τ_t. The acceptance or delay of the demand depends, as already explained, on the resource R_j^* and the delay policy used by the providers.

To conclude, let us highlight that in order to account for dynamic resource demand, we can model different demands for a same user over different service time frames, including as well a change in priority over time to favor servicing demands in due time. Furthermore in practice, the available resource can vary across service time frames, with no actual impact on the algorithm which can adapt to changing available resource parameters over time.

E. Numerical analysis

We provide a numerical analysis of the sOCRA, sPRA1 and sPRA2 algorithms. We simulate 1000 time instants, with demands arrival following a Poisson distribution with mean 3. The demands are generated as done in Section IV-C3 while the minimum demand is set to 40% of the demand for each user. Table V shows the settings of the simulations. The four considered services are differentiated accordingly to the demand, the priority index, the deadline and the service time frame. The service time frame is simulated using a geometric distribution with different parameters depending on the service type. We consider four experiments varying the quantity of available resource. The value of the resource, linked to the average demand of each resource, is chosen as follows: $R^1 = 3Colavg(\mathbf{D})$, $R^2 = 5Colavg(\mathbf{D})$, $R^3 = 10Colavg(\mathbf{D})$, $R^4 = 20Colavg(\mathbf{D})$, where $Colavg(\mathbf{D})$ is the average by column of the simulated demands in the dataset $\mathbf{D}$.

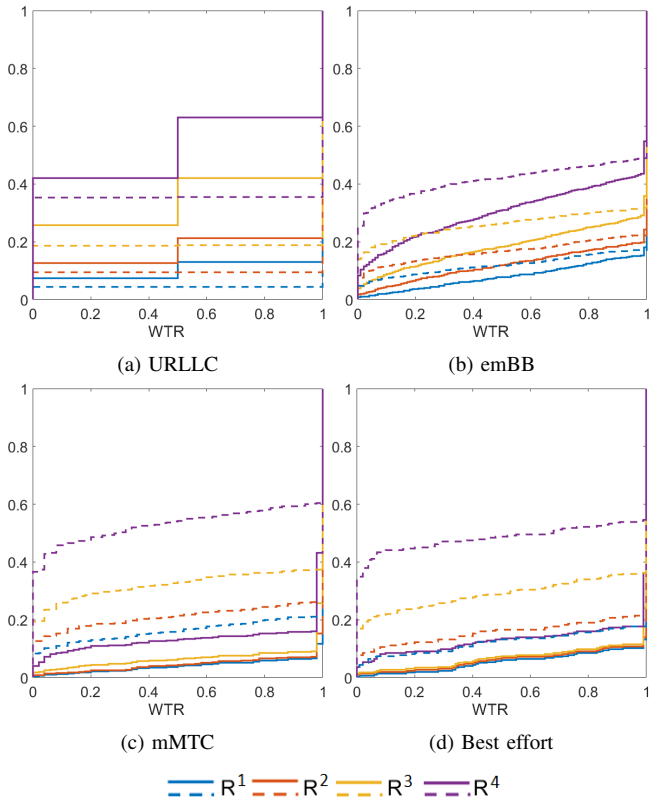


Fig. 15: WTR CDF - The dotted line is for the baseline algorithm and the straight one for our proposal.

We consider two metrics to analyze the performance of our algorithm. The first one is the *Waiting Time Ratio (WTR)* calculated as the ratio of the waiting time over the deadline, that is the maximum waiting time of a tenant. The second metric is the *Acceptance Ratio* that is the ratio of the number of users served, independently of when (soon or at an instant of time close to the deadline), over the number of users demanded resources. We compare our proposed scheduling algorithm to a baseline one where the selection of the user to serve is random instead of following the user delay policy. We want to emphasize that the analysis does not go into the value of the allocation provided using the cascade or parallel algorithms as we want to assess the effectiveness in using an enhanced user selection policy.

Figure 15 shows the Cumulative Distributed Function (CDF) of the WTR for the two algorithms, as a function of the available resource. If the waiting time is zero, i.e., the tenant is served when it submits the demand, then the WTR is equal to 0 while if WTR=1 the tenant is not served. As one would expect, the number of served users, both with the baseline and our algorithm, is increasing if the available resources

[7]In case the providers decrease their available resource it is possible that the allocation for some of these users fall below the minimum threshold given by the SLA

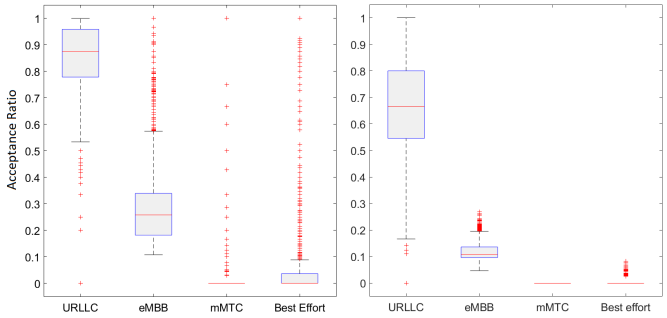
[8]Other strategies can be used to try to serve also users with low priority before their demand expire. For example, the priority can be increased in a gradual way (passing from the tenants user level of priority to the upper one) each given number of instants of time.

		Centralized	CRA		OCRA		PRA-1	PRA-2
			Best case	Worst case	Best case	Worst case		
Actions	Demand submission (τ)	1	1	1	1	1	1	1
	Solution submission (τ)	1	p	p	$p-1$	p	1	2
	Other information submission (τ)	-	-	-	2	2	-	-
	Allocation computation (δ)	1	1	p	1	p	1	2
	Total delay budget	$2\tau+\delta$	$(p+1)\tau+\delta$	$(p+1)\tau+p\delta$	$(p+2)\tau+\delta$	$(p+3)\tau+p\delta$	$2\tau+\delta$	$3\tau+2\delta$

TABLE VI: Number of actions required to reach the solution and total delay budget.

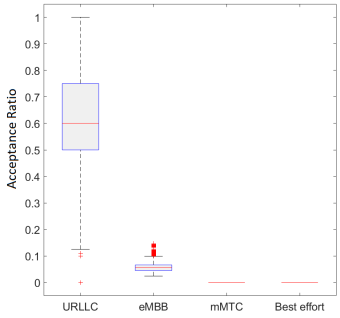
		Centralized	CRA	OCRA		PRA-1	PRA-2
				Best case	Worst case		
Communication	tenant $\rightarrow$ multi-domain orchestrator	1	-	-	-	-	-
	tenant $\rightarrow$ provider	-	p	p	p	p	p
	provider $\rightarrow$ provider	-	$2(p-1)$	$(p-1)$	$(p-1)+\sum_{i=1}^{p-1} i$	$p(p-1)$	$p(p-1)+(p-1)$
	provider $\rightarrow$ multi-domain orchestator	p	-	p	p	-	-
	multi-domain orchestator $\rightarrow$ provider	p	-	p	p	-	-
	Message complexity	$2p+1$	$3p-2$	$4p-1$	$\frac{p^2+7p}{2}-1$	p^2	p^2+p-1

TABLE VII: Number of communications required to reach the solution and message complexity. Moreover, we provide in the supplementary materials an additional figure to show the message complexity as function of the number of resource providers.

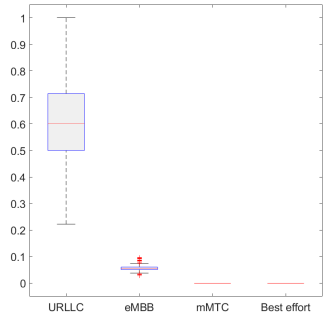


(a) congestion level: 0-25 %

(b) congestion level: 25-50 %



(c) congestion level: 50-75 %



(d) congestion level $>$ 75 %

Fig. 16: Boxplot of the Acceptance Ratio as function of the congestion level.

increase; for example, in the case of URLLC requests with $R^1 = 3Colavg(\mathbf{D})$ we are able to serve less than 20% of the tenants while with $R^4 = 20Colavg(\mathbf{D})$ we can serve around 35% of the tenants with the baseline algorithm and more than the 60% of the tenants with our algorithm. We highlight that the step-like behavior of the URLLC CDF is due to the fact that the deadline is equal to 2, and consequently the WTR can have only three values. Furthermore, we can observe the good performance of the algorithm into serving users with high priority. In fact, in case of URLLC and eMBB services the number of served users, independently of the waiting time, is higher with our algorithm while it is inferior for mMTC and Best effort services. In particular, the greater increase of served users with our algorithm happens thanks to the increase of the priority in the instant preceding the deadline. As a negative aspect it must be noted that the dotted line is above the straight line until the instant before the deadline thus the waiting times are longer with our approach.

In Figure 16 we group the results based on the congestion level and we plot the acceptance ratio. We can notice an expectable behavior, with higher acceptance ratio for services with higher priorities, for each congestion range, and decreasing acceptance ratio for each type of service when the congestion level increases. In particular, when the congestion is high (Fig. 16c and 16d), due to the lack of sufficient resources, tenants of low priority services as the mMTC and Best effort ones are rarely served.

VI. Conclusion

In this work we proposed algorithms to decentralize 5G slice resource allocation, using two cascading algorithms and two parallel algorithms. We extensively compared them, showing the advantages and disadvantages, also with respect to a centralized approach. In particular, we analyze the delay budget, the pros and cons in terms of reallocation, presence of a multi-domain orchestator, message complexity, level of confidentiality and possibility of allocate the radio resources independently. Finally by numerical simulations we (i) characterize the re-allocation events using the OCRA approach, (ii) estimate the percentage of inefficient solutions using PRA-1, and (iii) calculate the distance of distributed approaches solutions to the centralized one.

We then extended the algorithms proposing an algorithm to include Service Level Agreements for enforcing minimal allocation guarantees and time constraints. The numerical analysis shows that the proposed algorithm has good performances in terms of waiting time and acceptance ratio, increasing

the service rate of tenants with high priority compared to a baseline algorithm not considering the tenants priorities.

Future works on the subject can be on fault-tolerant distributed multi-resource allocation, and on the integration into ORAN/SDRAN systems of the proposed algorithms. Moreover, new fairness concepts and allocation rules for distributed multi-resource allocation may also be investigated.

APPENDIX

Proof of the delay budget and message complexity

Let us remember that the transfer times is τ, the allocation computing times is δ and that the actions bringing delay in the algorithms are:

- Demand submission: delay between the submission of the demand by the users and the receipt of the demand by the multi-domain orchestrator (centralized case) or the resource provider (distributed case);
- Solution submission: delay between the submission of the solution by one or more provider and the receipt of the solution by one or more provider;
- Other information submission: delay between the submission of other information type by one or more providers or by the multi-domain orchestrator and the receipt of the solution by one or more providers or by the multi-domain orchestrator;
- Allocation computation: delay required by the provider or multi-domain orchestrator to calculate the solution.

Table VI shows the number of each action necessary to allocate the resources with the considered algorithms and the budget delay resulting from this. To calculate the message complexity we need to count the exchanged messages, i.e., the number of arrows in Figure 6 in case of three providers. For a general case with p providers the results are proved in Table VII.

REFERENCES

[1] 5G Americas. "Network Slicing for 5G and Beyond". *White Paper*, 2016.

[2] NGMN. "5G white paper". *Next generation mobile networks*, 2014.

[3] 3GPP TS 22.261 V15.5.0, 5G; Service requirements for next generation new services and markets.

[4] O-RAN Use Cases and Deployment Scenarios, White Paper O-RAN Alliance, February 2020.

[5] A. Ghodsi, et al. "Dominant resource fairness: fair allocation of multiple resource types." *Proc. of USENIX NSDI 2011*.

[6] F. Fossati, S. Moretti, P. Perny, S. Secci. "Multi-Resource Allocation for Network Slicing", *IEEE/ACM Transactions on Networkig*, 28.3: 1311-1324, June 2020.

[7] F. Fossati , S. Moretti, S. Rovedakis, S. Secci. "Decentralization of 5G slice resource allocation", *2020 IFIP/IEEE Int. Symposium on Network Operations and Management Systems (NOMS)*.

[8] W. Thomson. "Axiomatic and game-theoretic analysis of bankruptcy and taxation problems: an update", *Math. Soc. Sciences* 74: 41-59, 2015.

[9] FP. Kelly, AK. Maulloo, DKH Tan. "Rate control for communication networks: shadow prices, proportional fairness and stability." *J. of the Operational Research society* 49.3, 1998.

[10] R. Jain, DM. Chiu, WR. Hawe. *"A quantitative measure of fairness and discrimination for resource allocation in shared computer system."* Vol. 38. Hudson, MA: East. Res. Lab., Digital Equipment Corporation, 1984.

[11] DP. Bertsekas, RG. Gallager, P. Humblet. *Data networks.* Vol. 2. New Jersey: Prentice-Hall International, 1992.

[12] J. Mo, J. Walrand. "Fair end-to-end window-based congestion control." *IEEE/ACM Trans. on Networking*, 8.5: 556-567, 2000.

[13] F. Fossati, S. Moretti, S. Secci. "A Mood Value for Fair Resource Allocations". *IFIP Networking 2017*.

[14] F. Fossati, S. Hoteit, S. Moretti, S. Secci. "Fair Resource Allocation in Systems with Complete Information Sharing". *IEEE/ACM Transactions on Networking*, 26.6: 2801-2814, Nov. 2018.

[15] Y. Etsion, T. Ben-Nun and D. G. Feitelson. " A global scheduling framework for virtualization environments." *IEEE Int. Symposium on Parallel and Distributed Processing*, 2009

[16] P. Poullie, T. Bocek, B. Stiller. "A survey of the state-of-the-art in fair multi-resource allocations for data centers." *IEEE Transactions on Network and Service Management*, 15.1: 169-183, 2018.

[17] M. Leconte, et al. "A resource allocation framework for network slicing." *IEEE INFOCOM 2018*.

[18] P. Caballero, et al. "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads." *IEEE/ACM Transactions on Networking (TON)*, 25.5: 3044-3058, 2017.

[19] Y. Xiao, M. Hirzallah, and M. Krunz. "Distributed Resource Allocation for Network Slicing Over Licensed and Unlicensed Bands." *IEEE Journal on Selected Areas in Communications*, 36.10: 2260-2274, 2018.

[20] G. Wang, et al. "Resource Allocation for Network Slices in 5G with Network Resource Pricing." *IEEE GLOBECOM 2017*, 2017

[21] M.Jiang, M. Condoluci, T. Mahmoodi. "Network slicing in 5G: An auction-based model." *IEEE ICC 2017*, 2017.

[22] P. Caballero, et al. "Network slicing games: Enabling customization in multi-tenant networks." *IEEE INFOCOM 2017*, 2017.

[23] H. Halabian. "Distributed Resource Allocation Optimization in 5G Virtualized Networks." *IEEE Journal on Selected Areas in Communications* 37.3: 627-642, 2019.

[24] M. Series. "IMT VisionFramework and overall objectives of the future development of IMT for 2020 and beyond." *Recommendation ITU: 2083-0*, 2015.

[25] P. Popovski, et al. "5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view", *IEEE Access 6*, 2018.

[26] S. Lee, et al. " Resource Management in Service Chaining." *IETF Secretariat, Intert-Draft*, 2016.

[27] G. Coulouris, J. Dollimore, T. Kindberg. "Distributed systems - concepts and designs (3rd ed.)." *Addison-Wesley*, p. 452, 2001.

[28] Amazon EC2 instances comparison: https://www.ec2instances.info.

[29] Liu, Li, et al. "Preemptive hadoop jobs scheduling under a deadline." *IEEE Eighth International Conference on Semantics, Knowledge and Grids.* , 2012.

[30] V. G. Abhaya, Z. Tari, P. Zeephongsekul, and A. Y. Zomaya. "Performance analysis of edf scheduling in a multi-priority preemptive m/g/1 queue, *IEEE Trans. on Parallel and Distributed Systems* 25.8: 21492158, 2014.

[31] Jia, Ru, et al. "A Deadline Constrained Preemptive Scheduler Using Queuing Systems for Multi-tenancy Clouds." *IEEE 12th International Conference on Cloud Computing*, 2019.

[32] F. Fossati, S. Moretti and S. Secci. "Multi-Resource Allocation for Network Slicing under Service Level Agreements", *Proc. of NoF 2019*.

[33] Simulation code (website): https://roc.cnam.fr/sim_tpds.zip (accessed on June 24, 2021).

Francesca Fossati is currently a postdoc at CentraleSuplec, Gif-sur-Yvette, France. She received her M.Sc. in mathematical engineering from Politecnico di Milano, Italy in 2015. In 2019, she obtained a Ph.D. at LIP6, Sorbonne Université. Her current research interests are about optimization and game theory, with applications to network resource allocation problems.

Stephane Rovedakis is associate professor at Cnam (Conservatoire national des arts et métiers), Paris, France. In 2009, he obtained a Ph.D. in Computer Science from University of Evry, France. His research interests concern the design of fault-tolerant and distributed algorithms to solve routing and resource allocation problems with applications in networks.

Stefano Secci is full professor of networking at Cnam (Conservatoire national des arts et métiers), Paris, France. He received the M.Sc. Degree in telecommunications engineering from Politecnico di Milano, Milan, Italy, in 2005, and a dual Ph.D. Degree in computer science and networks from Politecnico di Milano and Telecom ParisTech, France, in 2009. He was associate professor at LIP6, UPMC from 2010 to 2018. His current interests cover network automation. Webpage: https://cedric.cnam.fr/~seccis.

Distributed algorithms for multi-resource allocation
Supplementary Materials annex

Francesca Fossati* Stéphane Rovedakis† Stefano Secci†

1 Message complexity

We provide in the following a supplementary figure to show the message complexity as function of the number of resource providers.

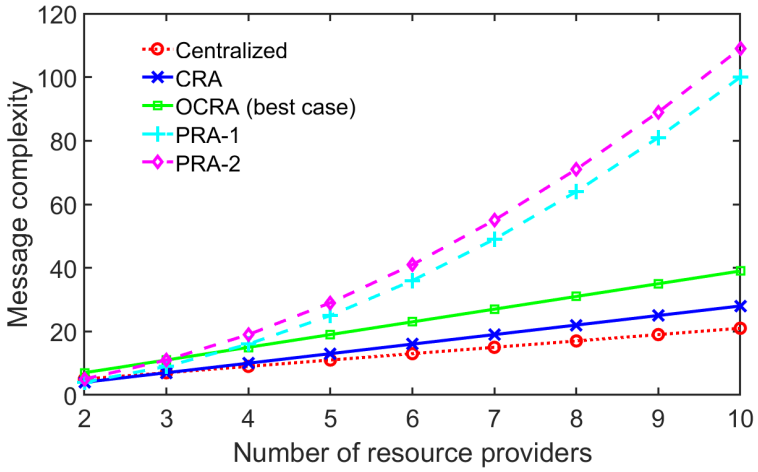


Figure 1: Messages complexity as function of the number of resource providers.

*Francesca Fossati is with CentraleSuplec, Gif-sur-Yvette, France. francesca.fossati@l2s.centralesupelec.fr
†Stéphane Rovedakis and Stefano Secci are with Cnam, Cedric, 75003 Paris, France. Email: {firstname.lastname}@cnam.fr

2 Numerical analysis: Amazon EC2 instances

We provide in the following the Amazon EC2 instances table used to generate users demands.

API Name	Memory (GB)	vCPUs	Gbps	Instance Type
m4.10xlarge	160.00	40.00	10.00	General purpose
m4.16xlarge	256.00	64.00	25.00	General purpose
c5.9xlarge	72.00	36.00	10.00	Compute optimized
c5.18xlarge	144.00	72.00	25.00	Compute optimized
c4.8xlarge	60.00	36.00	10.00	Compute optimized
r4.8xlarge	244.00	32.00	10.00	Memory optimized
r4.16xlarge	488.00	64.00	25.00	Memory optimized
x1.16xlarge	976.00	64.00	10.00	Memory optimized
x1.32xlarge	1952.00	128.00	25.00	Memory optimized
x1e.16xlarge	1952.00	64.00	10.00	Memory optimized
x1e.32xlarge	3904.00	128.00	25.00	Memory optimized
p3.8xlarge	244.00	32.00	10.00	Accelerated comput.
p3.16xlarge	488.00	64.00	25.00	Accelerated comput.
p2.8xlarge	488.00	32.00	10.00	Accelerated comput.
p2.16xlarge	732.00	64.00	25.00	Accelerated comput.
g3.8xlarge	244.00	32.00	10.00	Accelerated comput.
g3.16xlarge	488.00	64.00	25.00	Accelerated comput.
f1.16xlarge	976.00	64.00	25.00	Accelerated comput.
h1.8xlarge	128.00	32.00	10.00	Storage optimized
h1.16xlarge	256.00	64.00	25.00	Storage optimized
d2.8xlarge	244.00	36.00	10.00	Storage optimized
i3.8xlarge	244.00	32.00	10.00	Storage optimized
i3.16xlarge	488.00	64.00	25.00	Storage optimized

Table 1: Amazon EC2 instances