DOI: https://doi.acm.org/10.1145/3281411.3281435

Overbooking Network Slices through Yield-driven End-to-End Orchestration

Josep Xavier Salvat[1 2], Lanfranco Zanzi[1 2],
Andres Garcia-Saavedra[1], Vincenzo Sciancalepore[1], Xavier Costa-Perez[1]
[1] NEC Laboratories Europe, Heidelberg, Germany
[2] Technische Universität Kaiserslautern, Germany

ABSTRACT

Network slicing allows mobile operators to offer, via proper abstractions, mobile infrastructure (radio, networking, computing) to vertical sectors traditionally alien to the telco industry (e.g., automotive, health, construction). Owning to similar business nature, in this paper we adopt yield management models successful in other sectors (e.g. airlines, hotels, etc.) and so we explore the concept of slice *overbooking* to maximize the revenue of mobile operators.

The main contribution of this paper is threefold. First, we design a hierarchical control plane to manage the orchestration of slices *end-to-end*, including radio access, transport network, and distributed computing infrastructure. Second, we cast the orchestration problem as a stochastic yield management problem and propose two algorithms to solve it: an optimal Benders decomposition method and a suboptimal heuristic that expedites solutions. Third, we implement an experimental proof-of-concept and assess our approach both experimentally and via simulations with topologies from three real operators and a wide set of realistic scenarios.

Our performance evaluation shows that slice overbooking can provide up to 3x revenue gains in realistic scenarios with minimal footprint on service-level agreements (SLAs).

CCS CONCEPTS

• **Networks → Mobile networks**; *Network algorithms*;

KEYWORDS

5G; Network slicing; Orchestration; Yield management

ACM Reference Format:
Josep Xavier Salvat[1 2], Lanfranco Zanzi[1 2], Andres Garcia-Saavedra[1], Vincenzo Sciancalepore[1], Xavier Costa-Perez[1]. 2018. Overbooking Network Slices through Yield-driven End-to-End Orchestration. In *The 14th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '18), December 4–7, 2018, Heraklion, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3281411.3281435

1 INTRODUCTION

The hype around software-defined networking (SDN) and network function virtualization (NFV) is the projection of a trend towards network *softwarization* and *programmability* that is blending together telecommunication and computing industries. This combination has a deep impact on mobile communications infrastructure that is yielding a transformation from relatively complex monolithic architectures into a flurry of *commoditized* networking, computing and radio resources [7, 20].

Clearly, the need of mobile operators to augment their revenue is a strong pull towards said convergence, spawning uncharted sources of monetization as a result. Namely, the availability of *cloudified* networking, computing, and radio resource pools can now be offered, via proper abstractions, to *vertical* sectors (e.g., automotive, health, construction)—traditionally alien to the telco sector—as a means to enable new services such as remote-controlled machinery, augmented/virtual reality (AR/VR), etc. [11, 48]. An example of this symbiosis is the momentum that multi-access edge computing (MEC) is gaining to provide services near *the edge*, a unique commodity that only mobile operators can offer.

In this context, *Network Slicing* appears as a key solution to accommodate these emerging business opportunities in next generations of mobile systems [19]. The Next Generation Mobile Networks (NGMN) Alliance defines a network slice as "*a set of network functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the service instance(s)*" (c.f. [35]). Inspired by recent advances on SDN and NFV, this concept shall provide the required tools to allocate (virtual) resources to 3rd-parties in an isolated, flexible and guaranteed manner. It thus becomes evident that the orchestration of resources *end-to-end*[1] is, albeit challenging, a requirement in order to provision network slices with (*i*) spectrum at radio sites, (*ii*) transport services in the backhaul and (*iii*) computing/storage at distributed computing clouds.

Nevertheless, its benefits are compelling. Network Slicing leads mobile operators towards business models that, perhaps surprisingly, have a similar nature to successful yield management strategies popular in areas such as airline or hotel industries, and promise substantial gains in the revenue attained to mobile investments. In particular, in this paper we explore the concept of *slice overbooking*, accommodating the common practice in airline services of intentionally allocating more cargo than available capacity to the allocation of mobile network slices for 3rd-party services.

The challenge to adopt an orchestration system based upon the concept of slice overbooking is threefold: (*i*) when doing overbooking, resource deficit (and thus violations of system-level agreements) may occur; and so, in order to maintain the incentives for

[1]With *end-to-end*, we refer to all domains of the mobile network ecosystem, including network/storage/computing/radio resources. Domains beyond the ownership of a mobile operator, e.g., Internet Service Providers (ISPs), are not considered.

3rd-parties (users) to join the system, a balance between overbooking and potential service disruption must be taken care of; (*ii*) we need to untangle the coupling between resource reservation *and* slice admission control decisions, which is further compounded by the heterogeneous nature of the resources required to build a slice across the whole system; (*iii*) we need to make an appropriate use of monitoring information to be able to adapt to behavioral dynamics of 3rd-party services embedded in network slices.

The main contributions of our paper are:

- We design an end-to-end (E2E) orchestration platform for mobile systems based on a hierarchical control plane that exploits feedback information from network slices to make orchestration decisions;
- We formalize our orchestration problem as a yield management problem that jointly performs admission control and resource reservation across all domains of the mobile system exploiting the concept of slice overbooking. We derive two algorithms: an optimal approach based on Benders decomposition and a sub-optimal heuristic that expedites decisions;
- We build an experimental proof-of-concept with conventional mobile equipment and assess the performance of our system via experiments and simulations with large urban topologies from three real operators in Europe.

The remainder of this paper goes as follows. §2 presents our system design including control and data planes, implementation details and a mathematical model of our system. Our orchestration problem is formalized as a yield management problem in §3. Then, §4 introduces two algorithms: an exact solver based on Benders decomposition a light heuristic for larger-scale systems, and a set of simulation results to assess their performance in urban networks from 3 real operators. §5 introduces our experimental prototype and a set of experiments in a controlled testbed comprised of conventional mobile equipment for validation. Finally, §6 discusses related literature and §7 presents our concluding remarks.

2 SYSTEM DESIGN AND MODEL

We now introduce the design of our system and a mathematical model that allows us to make orchestration decision. Our system has decoupled control and data planes. The data plane is comprised of base stations, switches and computing infrastructure. In the control plane, we have a hierarchical architecture where local domain controllers are governed by an end-to-end (E2E) orchestrator.

2.1 Data Plane

As depicted in Fig. 1, we consider a system with a radio access network (RAN) comprised of $\mathcal{B} := \{1, \ldots, B\}$ base stations (BS), a distributed computing fabric with $\mathcal{C} := \{1, \ldots, C\}$ computing units (CUs), and a transport network connecting BSs and CUs that we model as an undirected graph where the edges, collected in set $\mathcal{E}$, are network links.

2.1.1 Service model. We allow tenants to deploy their services, dubbed vertical services (VSs), within a slice of the system. Such VSs are provided by the tenant in an offline on-boarding phase, e.g., as virtual machines (VMs). The first task to create a slice is to construct a *network service* (NS) with sufficient computing resources

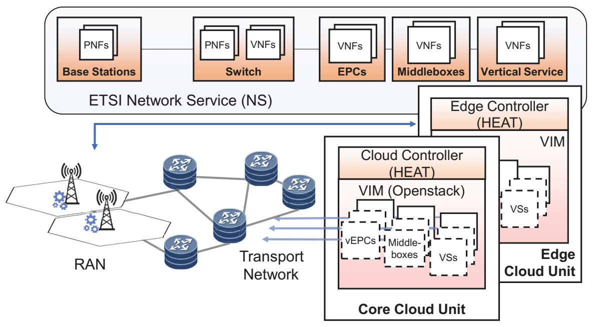


Figure 1: Data plane

allocated to the VS (and related mobile functions), connectivity in the transport network, and spectrum resources at radio sites to enable VS access to the tenant's users. To this aim, we model such network service as an ETSI NFV NS [29], with a chain of physical network functions (PNFs, e.g., slices of BSs and switches), the vertical service (VS) and all virtual network functions (VNFs) that connect end-users and VS (e.g., GTP gateways, MME, etc.). This is shown in Fig. 1.

2.1.2 Resources. We assume BSs with RAN sharing or slicing support (e.g. [13]), an SDN-based transport network and OpenStack as compute infrastructure manager (although other cloud managers can be accommodated). BSs, network links and CUs are characterized by a *capacity* value C_b, C_e and $C_c \in \mathbb{R}_+$ indicating, respectively, the maximum amount of radio resources (spectrum chunks), transport network resources (bits per second) and computing resources (shares of aggregated CPU pools)[2] that can be allocated to a service in BS $b \in \mathcal{B}$, network link $e \in \mathcal{E}$ and CU $c \in \mathcal{C}$. To keep our problem tractable, we assume that the microscopic problem of selecting a server for a VNF within a CU is handled locally by a cloud orchestrator (e.g., Heat),[3] and focus in this paper on the macroscopic problem of jointly optimizing (*i*) slice access control, (*ii*) CU selection, and (*iii*) reservation of resources across the system for the NS. Now, we let $p_{b,c} = \langle e_1, e_2, \cdots \rangle$ be a sequence of links $e_i \in \mathcal{E}$ connecting BS b and a CU c (i.e., a *path*) and $\mathcal{P}_{b,c}$ be a set with all possible available paths $p_{b,c}$. This can be readily computed offline using, e.g., k-shortest path methods based on Dijkstra's algorithm. Each path $p \in \mathcal{P}_{b,c}$ is further characterized with a delay D_p.

2.1.3 Middleboxes. We rely on an overbooking mechanism that adapts the reservation of resources to the actual demand of each slice (or a prediction of it) as explained later on. However, we may violate service-level agreements (SLAs) when making overly optimistic predictions (slice overbooking). In these cases (which we strive to minimize), it is important to avoid perturbations of the transmitter's behavior. If we simply delayed or dropped packets, TCP's transmission control of end-users would react in an undesirable manner. Hence, we need a scheme to under-provision resources that is also transparent to the tenant's users.

[2]To avoid notation clutter, we focus on compute resources only; however our model can be readily extended to consider others such as storage.
[3]We refer the reader for more details on the microscopic issue to [6].

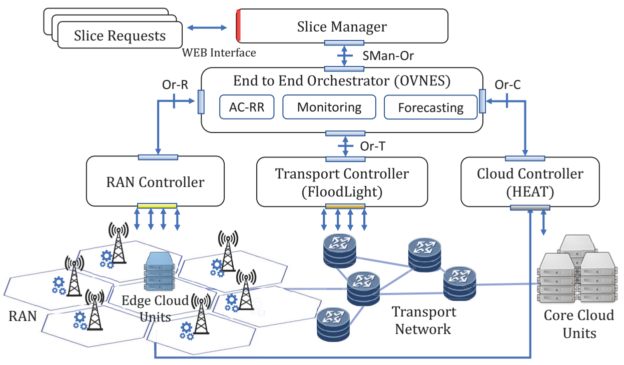


Figure 2: Control plane

TCP proxies are nowadays common in many service gateways and load balancers in operational networks to improve throughput performance, enhance security, perform network analysis and traffic control [27, 30]. In our system, we exploit basic TCP proxy functionality in a *middlebox* as depicted in Fig. 1. Our proxy creates a TCP overlay network splitting each connection into two as per Split TCP [28]: the former between the service of the slice and the middlebox, and the latter between the middlebox and the end-user(s) of the slice *where we do rate control*. If the slice's (aggregate) load exceeds the SLA, packets are randomly dropped to adjust the rate to the SLA. If the load is within the SLA parameters *and* below the maximum network capacity reserved for the slice (as detailed later), the middlebox simply forwards packets transparently. Finally, if the load is within the SLA parameters *but* it exceeds the network capacity reserved for the slice, the middlebox buffers packets to adjust the rate to the reserved capacity. Buffered packets are immediately acknowledged back to the service and then transmitted to the final user upon capacity availability. This avoids that the rate controller of the transmitter's TCP implementation reacts to our traffic control actions when the load is within the tenant's SLA.

2.2 Control Plane

Our control plane is depicted in Fig. 2. At the top of the hierarchy, a *slice manager* interacts with the tenants and oversees the setup of a NS for the slice. In the middle, the *end-to-end orchestrator* embeds most of our system's intelligence and is in charge of performing access control and resources reservation activities for the slices all across the mobile system, and interacts with domain *controllers* (RAN, transport, cloud) to deploy the NS, accordingly.

2.2.1 *Slice Manager.*

We consider a time slotted system whereby time is divided into *decision epochs* $\langle 1, 2, \dots \rangle$. Tenants issue slice requests to the slice manager at any time within one decision epoch.[4] We then let $\mathcal{T}^{(t)}$ be the set of tenants requesting a slice in epoch t.

Each slice request is characterized by $\Phi_\tau := \{s_\tau, \Delta_\tau, \boldsymbol{\Lambda}_\tau, L_\tau\}$. s_τ is a function that maps the network load received by tenant τ's VS into computing requirements (details later). Δ_τ describes the latency tolerance between τ's service and any BS, and $\boldsymbol{\Lambda}_\tau = \left\{\Lambda_{\tau,p} \mid \forall p \in \mathcal{P}_{b,c}, b \in \mathcal{B}, c \in \mathcal{C}, \Lambda_{\tau,p} \in \mathbb{R}_+\right\}$ captures the service bitrate requested for τ's service at each radio site. Finally, L_τ is the duration of the slice. Should the slice be accepted, Φ_τ becomes an SLA between the tenant and the operator during L_τ intervals.

From the implementation perspective, we build our slice manager as a web app where tenants can introduce their Φ_τ requests. Internally, we use TOSCA templates to model NSs as shown in Fig. 1, and send it down to the E2E orchestrator using a REST interface.

2.2.2 *E2E Orchestrator.*

This is the main building block of our system. On the one hand, it processes monitoring data provided by each controller and provides data aggregation functions and forecasting algorithms. On the other hand, it makes judicious decisions regarding resource reservation and admission control, and interacts with the different controllers in order to enforce such decisions. From a software perspective, and to prove our concept, we develop our own orchestrator in Java.[5] This is the only entity that maintains system *state* information. All the remaining entities (i.e., slice manager, controllers) are stateless in order to guarantee consistency. As shown in Fig. 2, the main functional sub-blocks (connected by means of a REST interface) are the following:

Admission Control and Resource Reservation (AC-RR). At the beginning of each decision epoch t the AC-RR engine has to (*i*) decide which slices are accepted among those requests arrived during the previous decision interval, (*ii*) select a CU to instantiate the VNFs/VS of each NS, and (*iii*) make radio/transport/compute resource reservations across the system (i.e., make an infrastructure slice) *in order to maximize the net revenue* obtained from the tenants. To this aim, we let $x_{\tau,p}^{(t)}$ denote whether tenant τ is granted access to path p ($x_{\tau,p}^{(t)} = 1$) or not ($x_{\tau,p}^{(t)} = 0$); if slice Φ_τ is rejected, then $\sum_p x_{\tau,p}^{(t)} = 0$. Let us also define $z_{\tau,p}^{(t)}$ as the resource reservation for tenant τ, in terms of bitrate, when using path p, as illustrated in Fig. 3 (top). Importantly, $z_{\tau,p}^{(t)}$ is not necessarily the amount of transport resources reserved in path p (there are transport overheads we need to account for), but the bitrate associated to the service when using this path. Based on $z_{\tau,p}^{(t)}$, however, we derive the reservations of radio, transport and compute resources for slice Φ_τ. For notation convenience, we vectorize $x_{\tau,p}^{(t)}$ and $z_{\tau,p}^{(t)}$ into $\boldsymbol{x}^{(t)} \in \{0,1\}^{\mathcal{S}^{(t)}}$ and $\boldsymbol{z}^{(t)} \in \mathbb{R}_+^{\mathcal{S}^{(t)}}$, where $\mathcal{S}^{(t)} := \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}} \sum_{p\in\mathcal{P}_{b,c}} |\mathcal{T}^{(t)}|$.

In order to make decisions, we formalize our problem as a yield management problem (§3) and devise two algorithms to solve it (§4). As a result, the TOSCA NS descriptors are modified accordingly and passed down to the different domain controllers through a REST interface that follows closely the ETSI GS NFV-IFA 005 specification.

Monitoring and Feedback. We further divide the time window between two decision epochs into $\kappa^{(t)} := \langle 1, 2, \dots \rangle$ *monitoring samples*. As depicted in Fig. 3 (bottom), the monitoring function collects VS network load samples in sequences $\langle \lambda_{\tau,p}^{(\theta)} \mid \theta \in \kappa^{(t)} \rangle$ for every epoch t. With a slight abuse of notation, we let $\lambda_{\tau,p}^{(t)} = \max\left\{\lambda_{\tau,p}^{(\theta)} \mid \theta \in \kappa^{(t)}\right\}$ denote the maximum demand of resources during epoch t. This value can be computed for past epochs $\{1, \dots, t-1\}$ but it is unknown in the current one. Note that we

[4]We assume it as an adjustable parameter, e.g., based on (off-)peak hours [31, 32] that may trade off the forecast accuracy and speed of reaction.

[5]We acknowledge the fact that there exists a plethora of software projects developing NFV orchestration tools (Tacker, OSM, Cloudify, etc.). We advertise that none of the tools accommodate our needs in full and thus we develop our own for the purpose of this paper. As future work, we aim to integrate our concept within a mainstream orchestration platform.

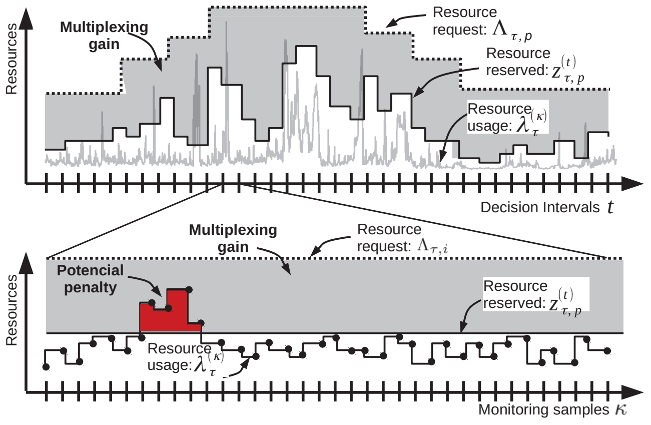


Figure 3: Resource dynamics and resource (under-)provisioning.

use max to account for peak aggregate loads that will allow us to minimize our under-allocation footprint. Therefore, we let $\hat{\lambda}^{(t)}_{\tau,p}$ denote the estimated (predicted) value for epoch t, and $0 < \hat{\sigma}^{(t)}_{\tau,p} \leq 1$ denote the level of uncertainty of such prediction. This is performed by the Forecasting sub-block, explained below.

In addition to service demand, another source of uncertainty is the wireless channel capacity. To model this, we let $\eta^{(t)}_{\tau,b}$ be a factor that maps radio spectrum (physical resource blocks (PRBs)) into actual load injected into the transport network (bits per second) for tenant τ and BS b at epoch t. Note that $\eta^{(t)}_{\tau,b}$ depends mostly on the average signal quality between users and BS, which can be monitored with conventional utilities and then estimated using standard radio models.

From our implementation perspective, we use `sFlow` to collect service load samples, `OpenStack Ceilometer/Gnocchi` to collect computing/storage monitoring data, and a proprietary protocol to gather signal quality samples from the RAN. Finally, we exploit `InfluxDB` to store time-series data and a `MySQL` database to save additional control plane information, e.g., current state of each slice.

Forecasting. This block processes the measurements (observations) performed during previous decision epochs t and provides the forecasting information to drive the system towards optimal states. In particular, we focus on a specific class of machine-learning algorithms that learn and predict the future traffic behaviors $\hat{\lambda}^{(\delta)}_{\tau,p}$ for the next N decision intervals, i.e., $\delta \in \{t+1,\ldots,t+N\}$. Exponential smoothing methods are common to properly handle future resource provisioning in cloud computing environments. However, the main drawback of (double) exponential smoothing is the inability to account for seasonabilities. Hence, our forecasting algorithm is based on a three-smoothing function.[6] This accurately applies to our problem as mobile data has periodicity features [36] that can be exploited to provide predicted traffic levels with a certain accuracy $\hat{\sigma}^{(\delta)}_{\tau,p}$. Therefore, we rely on the multiplicative version of Holt-winters (HW) algorithm [43], where the forecasting function f_{HW} is defined as $f_{HW} : \mathbb{R}^{|t-1|} \rightarrow \mathbb{R}^{|t+\delta|} \mid \lambda_{\tau,\mathbf{p}} \rightarrow \hat{\lambda}_{\tau,\mathbf{p}}$.

[6]Naturally, we can seamlessly plug in alternative forecasting methods, e.g., recent approaches based on neural networks [50].

2.2.3 Controllers. As depicted in Fig. 2, our orchestrator interacts with domain controllers to enforce orchestration decisions and to retrieve monitoring information. At the northbound of the Cloud controller, we translate the received `TOSCA` descriptor into a `Heat` template and send it down to a driver that interfaces with `OpenStack Heat` and `Keystone` for proper instantiation and CPU reservation (using CPU pinning [24]). Similarly, at the northbound of the Transport controller we translate the `TOSCA` descriptor into a series of OpenFlow instructions that are processed with `Floodlight` SDN controller to set up paths between BSs and CUs with appropriate capacity. Finally, we use the same descriptor file to configure radio shares of commercial LTE base stations, wherein each slice is connected to a different mobile core.

3 ADMISSION CONTROL & RESOURCE RESERVATION (AC-RR) PROBLEM

Maximization of a business' revenue falls into the category of *yield management*, a mainstream business theory that studies fare management, access control and resource allocation [42]. In the airline industry, the problem is to decide, based on the number of seat reservations, whether to accept or reject new requests considering that passengers may cancel, or even be "no-shows", prior to the flight departure. Thus, overbooking is performed with associated penalties determined by a penalty-cost function. Owning to similar business nature, we cast our slice orchestration problem into a stochastic yield management optimization problem.

3.1 Design of the objective function

Analogously to the airline example, we exploit the fact that users rarely consumes *all* the resources they request [22]. This gives us the opportunity to allocate more tenants than those presumably allowed by the leftover capacity, and gain additional revenue from slice multiplexing (overbooking). Clearly, an overly aggressive strategy may lead to resource deficit, discouraging potential users to join the system. We address this by (*i*) considering (forecasted) peak loads at each interval and (*ii*) designing a proper penalty-cost function. Consequently, we define

$$\psi^{(t)} := \sum_{\tau \in \mathcal{T}^{(t)}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \overbrace{K_\tau \Pr\left[z^{(t)}_{\tau,p} < \lambda^{(t)}_{\tau,p}\right] x^{(t)}_{\tau,p}}^{\text{Expected penalty}} - \overbrace{R_\tau x^{(t)}_{\tau,p}}^{\text{Reward}}$$

as the expected *instantaneous* cost in epoch t, and define

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}, \boldsymbol{z} \in \mathbb{R}^{\mathcal{S}}_{+}} \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \psi^{(t)} \qquad (1)$$

as our optimization problem, where R_τ is the reward obtained from accepting slice Φ_τ (e.g., subscription fee) and K_τ is a penalty paid to tenant τ when we violate its SLA,[7] which happens with probability $\Pr\left[z^{(t)}_{\tau,p} < \lambda^{(t)}_{\tau,p}\right]$. The target is to asymptotically minimize the aggregate cost or, equivalently, maximize the net reward.

[7]These coefficients K_τ and R_τ shall be designed to balance user incentives and revenue. We refer the reader to related economy literature [33].

A possible approach to solve this problem is to model $\lambda_{\tau,p}$ as a random variable with known distribution, and estimate its parameters looking at the realizations. This falls into the realm of stochastic programming where the aim is to balance reward maximization (right-hand side of $\Psi^{(t)}$) with the cost of a recourse action (left-hand side). However, in practice, $\lambda_{\tau,p}$ may be characterized by an intractable distribution and/or discretization may lead to overly complex computation. Hence, we adopt a more practical approach.

First, we assume that the duration of a slice L_τ is relatively small compared to the system's time horizon. Therefore, solving Eq. (1) is equivalent to minimizing $\psi^{(t)}$ at each decision epoch. This also allows us to drop the superscript (t) to simplify the notation and mitigate clutter in our analysis.

Second, we substitute $\Pr\left[z_{\tau,p}^{(t)} < \lambda_{\tau,p}^{(t)}\right]$ with *a risk cost function* $\rho(z_{\tau,p}, \hat{\sigma}_{\tau,p}, L_\tau) := P_{\tau,p} \cdot \xi_{\tau,p}$ that depends on the resource reservation $z_{\tau,p}$, forecast uncertainty $\hat{\sigma}_{\tau,p}$ and slice duration L_τ, where the term

$$P_{\tau,p} := \frac{\Lambda_{\tau,p} - z_{\tau,p}}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}}, \quad 0 \le P_{\tau,p} \le 1,^{8}$$

captures *the risk of resource deficit* due to overly aggressive under-provisioning, and

$$\xi_{\tau,p} := \hat{\sigma}_{\tau,p} L_\tau, \quad 0 < \xi_{\tau,p} \le L_\tau,$$

is a *scaling factor* that accounts for the uncertainty in our prediction ($\hat{\sigma}_{\tau,p} > 0$) and the duration of the slice request ($L_\tau > 0$). In this way, we can rewrite our problem as:

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}, \boldsymbol{z} \in \mathbb{R}_+^{\mathcal{S}}} \Psi := \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \overbrace{K_\tau \rho(z_{\tau,p}, \hat{\sigma}_{\tau,p}, L_\tau) x_{\tau,p}}^{\text{Estimated penalty}} - \overbrace{R_\tau x_{\tau,p}}^{\text{Reward}}$$

We next introduce the constraints of our problem.

3.2 Constraints

We first formulate the system capacity constraints as

$$\sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}}} a_\tau + z_{\tau,p} b_\tau \le C_c, \qquad \forall c \in C \tag{2}$$

$$\sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} z_{\tau,p} \eta_e \mathbb{1}_{e \in p} \le C_e, \qquad \forall e \in \mathcal{E} \tag{3}$$

$$\sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall c \in C}} z_{\tau,p} \eta_{\tau,b} \le C_b, \qquad \forall b \in \mathcal{B} \tag{4}$$

describing capacity constraints of CU resources, transport links, and BSs, respectively. Parameters $a_\tau, b_\tau \in s_\tau$ in Eq. (2), characterize the linear relationship between network load arriving at the service of tenant τ and its computing requirements.[9] a_τ models a baseline consumption associated to, e.g., the VS operative system, the mean number of users of the tenant, etc., and b_τ models the amount of computation required to serve the allocated bitrate. In Eq. (3), we let η_e model the overhead of the specific transport protocol used in link $e \in \mathcal{E}$ (e.g. VLAN/MPLS tags, GTP tunnels, etc.); and $\mathbb{1}_{e \in p}$ is equal to 1 only if link e belongs to path p. Finally, in Eq. (4), $\eta_{\tau,b}$ maps bitrate resources into radio resources, which can be estimated with readily available radio models.

We also add the following constraints:

$$\sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall c \in C}} x_{\tau,p} \le 1, \qquad \forall \tau \in \mathcal{T}, \forall b \in \mathcal{B} \tag{5}$$

to prevent multipath connections;[10]

$$\sum_{p_1 \in \mathcal{P}_{m,c}} x_{\tau,p_1} \le \sum_{p_2 \in \mathcal{P}_{n,c}} x_{\tau,p_2}, \quad \forall m \ne n \in \mathcal{B}, \forall c \in C, \forall \tau \in \mathcal{T} \tag{6}$$

to guarantee that accepted slices are given a slice of *all* BSs and that each BS slice belonging to the same system slice Φ_τ is connected to the same CU; and the delay constraint

$$\sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall c \in C}} x_{\tau,p} D_p \le \Delta_\tau, \qquad \forall \tau \in \mathcal{T}, \forall b \in \mathcal{B}. \tag{7}$$

Finally, we formulate the constraints that *couple* the resource reservation decisions ($\boldsymbol{z}$) and the routing/function placement and access control decisions ($\boldsymbol{x}$) as follows:

$$\boldsymbol{z} \le \boldsymbol{x}\Lambda \tag{8}$$

$$\boldsymbol{x}\hat{\boldsymbol{\lambda}} \le \boldsymbol{z} \tag{9}$$

that yield $\hat{\boldsymbol{\lambda}} \le \boldsymbol{z} \le \Lambda$, if Φ_τ is accepted, or $\boldsymbol{z} = \boldsymbol{0}$, otherwise.

3.3 AC-RR Problem

Consolidating the above, our problem becomes:

Problem 1 (AC-RR Problem).

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}, \boldsymbol{z} \in \mathbb{R}_+^{\mathcal{S}}} \quad \Psi(\boldsymbol{x}, \boldsymbol{z})$$

$$\text{s.t.} \quad (2), (3), (4), (5), (6), (7), (8), (9).$$

We note that $\Psi(\boldsymbol{x}, \boldsymbol{z})$ is a quadratic function. Fortunately, the structure of our problem yields the following conventional linearization technique. First, we create an auxiliary variable $y_{\tau,p} := z_{\tau,p} \cdot x_{\tau,p}$ and then rearrange the terms in Ψ to be linear with $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows. $\Psi(\boldsymbol{x}, \boldsymbol{z}) = \Psi(\boldsymbol{x}, \boldsymbol{y}) =$

$$\sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \left(\frac{\Lambda_{\tau,p} \xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} - R_\tau \right) x_{\tau,p} - \frac{\xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} y_{\tau,p}.$$

Second, we add the following constraints to maintain the linearized problem equivalent to the original Problem 1:

$$\boldsymbol{y} \le \Lambda \boldsymbol{x} \tag{10}$$

$$\boldsymbol{y} \le \boldsymbol{z} \tag{11}$$

$$\boldsymbol{z} + \Lambda \boldsymbol{x} \le \boldsymbol{y} + \Lambda \tag{12}$$

As a result, our AC-RR problem can be formulated as the following mixed integer linear problem (MILP):

[8]We later impose $\hat{\lambda}_{\tau,p} \le z_{\tau,p} \le \Lambda_{\tau,p}$, which yields $0 \le P_{\tau,p} \le 1$.
[9]This model is motivated by the strong linear correlation between network load and storage/compute usage in network services evinced in several works, e.g. [16, 23], and our own measurements. We assume the model parameters are *learnt* during an offline on-boarding phase.

[10]This constraint is motivated by the reluctance of operators to deploy multipath systems due to additional expenditures and delay (due to packet reordering) [41] but it can be relaxed if a multipath protocol is implemented [15].

Problem 2 (AC-RR MILP).

$$\min_{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}},\boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}},\boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} \quad \Psi(\boldsymbol{x},\boldsymbol{y})$$
$$\text{s.t.} \quad (2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12).$$

We next establish the complexity of our problem.

Theorem 1. *Problem 2 (and so Problem 1) is NP-Hard.*

Proof. The proof goes by reduction. Consider a restricted instance of Problem 2 (or Problem 1) with n tenants with no associated penalty ($K_\tau = 0$, $\forall\tau$), 1 CU c_1 with unlimited capacity $C_{c_1} \to \infty$, 1 BS b_1 with capacity $C_{b_1} = B$, and a simple transport network with a direct link e_1 connecting c_1 and b_1 with unlimited capacity $C_{e_1} \to \infty$ and no delay. Given this setting, it is trivial to cast this problem (in polynomial time) into the well-known knapsack problem [12], which is NP-hard. Adding multiple BSs and CUs increases the complexity of the problem, making it even harder to solve. This proves that Problem 2 is NP-Hard. □

3.4 Practical Considerations

There are a few additional practical details we need to consider. In particular, if tenant τ is accepted in t, we need to ensure that τ is also accepted in epochs $\{t+1, t+2, \ldots, t+L_\tau\}$. This can be done by adding the following constraint to Problem 2:

$$\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}} x_{\tau,p}\mathbb{1}_{\Omega_\tau\in\mathbb{Z}_{>0}} = 1, \forall\tau\in\left\{\mathcal{T}^{(1)},\ldots,\mathcal{T}^{(t-1)}\right\} \qquad (13)$$

where Ω_τ is a state variable of slice Φ_τ indicating the time the slice has left till expiration (for all previously accepted tenants).

However, (13) may render unfeasibility. Imagine a scenario where two slices have been accepted in t_1 for a duration equal to L. Now, if the load forecast of any tenant exceeds the capacity of some resource in t_2, $t_2 < t_1 + L$, we would encounter a deficit of resources that represents an unfeasible setting due to constraint (13). To address this, we relax the capacity constraints (2)-(4) as follows,

$$\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B}}} a_\tau + z_{\tau,p}b_\tau \le C_c + \delta_c, \qquad \forall c\in\mathcal{C} \qquad (14)$$

$$\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}} z_{\tau,p}\eta_e\mathbb{1}_{e\in p} \le C_e + \delta_b, \qquad \forall e\in\mathcal{E} \qquad (15)$$

$$\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \in\mathcal{C}}} z_{\tau,p,i}\eta_{\tau,b} \le C_b + \delta_r, \qquad \forall b\in\mathcal{B} \qquad (16)$$

and Problem 2 as follows

$$\min_{\substack{\boldsymbol{x}\in\{0,1\}^{\mathcal{S}},\boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}},\boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}\\ \delta_r\in\mathbb{R}_+,\delta_b\in\mathbb{R}_+,\delta_c\in\mathbb{R}_+}} \quad \Psi(\boldsymbol{x},\boldsymbol{y}) + M(\delta_r+\delta_b+\delta_c)$$
$$\text{s.t.} \quad (14),(15),(16),(5),(6),(7),(8),(9),(10),(11),(12),$$

where $\delta_r, \delta_b, \delta_c \in \mathbb{R}_+$ are auxiliary variables accounting for the deficit of radio, transport and computing resources, respectively, and M is a large value accounting for the cost of leasing these resources (e.g., via federation) or the penalties that we would have to pay (also sometimes known as "big M method"). This method fixes the unfeasibility issue as the resource deficit potentially caused by Eq. (13) is absorbed by the new auxiliary variables (at a high cost M). While we consider this in our implementation (as shown in §5), we omit these details in the following analysis to keep our presentation simple.

4 ALGORITHMS

We next present two algorithms to solve Problem 2: an optimal method based on *Benders decomposition*, designed for small to medium-scale networks, and a suboptimal *heuristic* that expedites solutions in medium to large-scale networks.

4.1 Benders Method

Our first methodology to solve Problem 2 lies on the observation that constraints (8), (9), (10) and (12) couple the real-valued resource reservation decision variables ($\boldsymbol{z}$, $\boldsymbol{y}$), and the binary placement and path selection decision variables ($\boldsymbol{x}$). We relax these constraints and decouple the slack problem into two subproblems by means of Benders decomposition [9]: one that involves the so-called "complicated" variables and one that involves only continuous variables.

We first describe our *slave* subproblem as follows:

Problem 3 (Slave problem $P_S(\bar{\boldsymbol{x}})$).

$$\min_{\boldsymbol{y}\in\mathbb{R}_+^{\mathcal{S}},\boldsymbol{z}\in\mathbb{R}_+^{\mathcal{S}}} \quad \sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}} -\frac{\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p}-\hat{\lambda}_{\tau,p}}y_{\tau,p}$$
$$\text{s.t.} \quad (2),(3),(4),(11)$$
$$\boldsymbol{z} \le \bar{\boldsymbol{x}}\boldsymbol{\Lambda} \qquad (17)$$
$$\bar{\boldsymbol{x}}\hat{\boldsymbol{\lambda}} \le \boldsymbol{z} \qquad (18)$$
$$\boldsymbol{y} \le \boldsymbol{\Lambda}\bar{\boldsymbol{x}} \qquad (19)$$
$$\boldsymbol{z} + \boldsymbol{\Lambda}\bar{\boldsymbol{x}} \le \boldsymbol{y} + \boldsymbol{\Lambda} \qquad (20)$$

which can be solved with standard linear programming solvers, and define its dual problem as $P_{DS}(\bar{\boldsymbol{x}})$.

Problem 4 (Dual slave problem $P_{DS}(\bar{\boldsymbol{x}})$).

$$\max_{\boldsymbol{\mu}\in\mathbb{R}_+^N} \quad g(\bar{\boldsymbol{x}},\boldsymbol{\mu})$$
$$\text{s.t.} \quad -b_\tau\mu_{1,c} - \sum_{e\in p}\eta_e\mu_{2,e} - \eta_{\tau,p}\mu_{3,b} - \mu_{4,\tau,p} + \mu_{5,\tau,p} +$$
$$+\mu_{7,\tau,p} - \mu_{8,\tau,p} \le 0, \ \forall b\in\mathcal{B},\forall c\in\mathcal{C},\forall p\in\mathcal{P}_{b,c},\forall\tau\in\mathcal{T}$$
$$-\mu_{6,\tau,p} - \mu_{7,\tau,p} + \mu_{8,\tau,p} \le -\frac{\xi_{\tau,p}K_\tau}{\Lambda_{\tau,p}-\hat{\lambda}_{\tau,p}},$$
$$\forall b\in\mathcal{B},\forall c\in\mathcal{C},\forall p\in\mathcal{P}_{b,c},\forall\tau\in\mathcal{T}$$

where $g(\bar{\boldsymbol{x}},\boldsymbol{\mu}) =$

$$\sum_{c\in\mathcal{C}}\mu_{1,c}\left(\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B}}} a_\tau - C_c\right) - \sum_{e\in\mathcal{E}}\mu_{2,e}C_e - \sum_{b\in\mathcal{B}}\mu_{3,b}C_b +$$
$$+\sum_{\tau\in\mathcal{T}}\sum_{\substack{p\in\mathcal{P}_{b,c}\\ \forall b\in\mathcal{B},c\in\mathcal{C}}}\Big(-\mu_{4,\tau,p}\bar{x}_{\tau,p}\Lambda_{\tau,p} + \mu_{5,\tau,p}\bar{x}_{\tau,p}\hat{\lambda}_{\tau,p} -$$
$$-\mu_{6,\tau,p}\Lambda_{\tau,p}\bar{x}_{\tau,p} + \mu_{8,\tau,p}(\Lambda_{\tau,p}\bar{x}_{\tau,p} - \Lambda_{\tau,p})\Big)$$

Algorithm 1 Benders method

```
1:  $k \leftarrow 1$
2:  Initialize $C_1 = C_2 = \varnothing$, $UB^{(1)} = -LB^{(1)} >> 1$
3:  while $UB^{(k)} - LB^{(k)} > \epsilon$ do
4:      $LB^{(k)}, \boldsymbol{x}^{(k)}, \theta^{(k)} \leftarrow PM(C_1, C_2)$
5:      $\boldsymbol{\mu}^{(k)} \leftarrow_{DS} (\boldsymbol{x}^{(k)})$
6:      if $P_{DS}(\boldsymbol{x}^{(k)})$ is unbounded then
7:          $\boldsymbol{\mu}^l \leftarrow$ extreme ray
8:          $C_2 \leftarrow C_2 \cup \{\boldsymbol{\mu}^l\}$
9:      else
10:         $\boldsymbol{\mu}^m \leftarrow$ extreme point
11:         $C_1 \leftarrow C_1 \cup \{\boldsymbol{\mu}^m\}$
12:     $\Gamma = \sum_{\tau \in \mathcal{T}} \sum_{p \in \mathcal{P}_{b,c}, \forall b \in \mathcal{B}, c \in C} \left( \frac{\Lambda_{\tau,p} \xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} - R_\tau \right) x_{\tau,p}^{(k)} - g\left(\boldsymbol{x}^{(k)}, \boldsymbol{\mu}^{(k)}\right)$
13:     if $UB^{(k-1)} > \Gamma$ then $UB^{(k)} = \Gamma$
14:     $k \leftarrow k + 1$
15: $\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$
16: $\boldsymbol{y}^*, \boldsymbol{z}^* \leftarrow P_{DS}(\boldsymbol{x}^{(k)})$
```

and $\boldsymbol{\mu}$ is the vector of $N = C + |\mathcal{E}| + B + 5\mathcal{S}$ dual variables.

We then formulate our *master* subproblem as follows:

Problem 5 (Master problem $P_M(C_1, C_2)$).

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}, \theta \in \mathbb{R}_+} \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \left(\frac{\xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} \Lambda_{\tau,p} - R_{\tau,p} \right) x_{\tau,p} + \theta$$

$$\text{s.t.} \quad (5), (6), (7)$$

$$g\left(\boldsymbol{x}, \boldsymbol{\mu}^m\right) \leq \theta, \qquad \forall \boldsymbol{\mu}^m \in C_1 \quad (21)$$

$$g\left(\boldsymbol{x}, \boldsymbol{\mu}^l\right) \leq 0, \qquad \forall \boldsymbol{\mu}^l \in C_2 \quad (22)$$

where θ is a surrogate variable substituting the "cost" of the resource reservation decisions, and equations (21) and (22) correspond to the optimality and feasibility cuts, respectively, added iteratively by Algorithm 1. We then use the iterative Algorithm 1 to solve Problem 2. The optimality of this approach is formalized in the following theorem.

Theorem 2 (Algorithm 1 Optimality). *Algorithm 1 converges to the optimal solution of Problem 2 in a finite number of iterations.*

Proof. The proof follows from the Partition Theorem in [9]. Let us consider the abstract formulation of Problem (5):

$$\min_{\boldsymbol{x}, \theta} \; c_1^T \boldsymbol{x} + \theta \quad \text{s.t.} \quad (\boldsymbol{x}, \theta) \in \mathcal{G}, \qquad (23)$$

where $\mathcal{G}$ is the set of constraints, created by the intersection of the constraints in $\mathcal{X}$ and the convex hull of the extreme halflines resulting from the dual slave problem (which is a polyhedral cone C). Algorithm 1 is initialized with empty sets C_1 and C_2 and thus $\mathcal{G}^{(1)}$ corresponds to a minimal set of constraints. At each iteration $k > 1$, the algorithm appends a point of the dual slave problem into set C_1 or C_2, which results in the addition of one extreme halfline of the cone C in $\mathcal{G}^{(k)}$. As a result, set $\mathcal{G}$ is iteratively reconstructed and, given that there is a finite number of them, convergence to the optimal solution is guaranteed because, in the worst case, the algorithm will reconstruct the full set $\mathcal{G}$. □

4.2 Heuristic Algorithm

While Benders method provides an optimal solution, it might take long time to converge. For larger scale systems, we propose a heuristic to solve Problem 5 by casting it into a classical multi-constrained 0-1 Knapsack problem model [34]:

Problem 6 (Multi-constrained Knapsack Problem).

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{S}}} \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \gamma_{\tau,p} \, x_{\tau,p}$$

$$\text{s.t.} \quad \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} w_{\tau,p}^{(k)} x_{\tau,p} \leq W^{(k)}, \quad \forall k \qquad (24)$$

$$\sum_{j \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}, c \in C}} \mathbb{1}_{j=\tau} \, x_{j,p} \leq 1, \quad \forall \tau \in \mathcal{T}; \qquad (25)$$

where $\gamma_{\tau,p}$ and $w_{\tau,p}^{(k)}$ in constraint (24) are the cost and the weight of item $x_{\tau,p}$, respectively, whereas $W^{(K)}$ is the total capacity of the knapsack. They are defined as follows.

$$\gamma_{\tau,p} = \left(\frac{\xi_{\tau,p} K_\tau}{\Lambda_{\tau,p} - \hat{\lambda}_{\tau,p}} \Lambda_{\tau,p} - R_{\tau,p} \right) \qquad (26)$$

$$w_{\tau,p}^{(k)} = -\mu_{4,\tau,p} \Lambda_{\tau,p} + \mu_{5,\tau,p} \hat{\lambda}_{\tau,p} - \mu_{6,\tau,p} \Lambda_{\tau,p} + \mu_{8,\tau,p} \Lambda_{\tau,p} \qquad (27)$$

$$W^{(k)} = -\sum_{c \in C} \mu_{1,c} \left(\sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}}} a_\tau - C_c \right) + \sum_{e \in \mathcal{E}} \mu_{2,e} C_e + $$
$$+ \sum_{b \in \mathcal{B}} \mu_{3,b} C_b + \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}}} \mu_{8,\tau,p} \Lambda_{\tau,p}. \qquad (28)$$

Note that constraints are dynamically added by Algorithm 1 at each iteration $k \geq 1$. The constraint set (25) accounts for constraint (5) in Problem 5.

When devising a lightweight solution to solve the above-mentioned problem, we rely on classical heuristics proposed for knapsack problems. We name our proposal Knapsack Admission Control (KAC) algorithm and we show the details in Algorithm 2. First, we combine together different weights $w_{\tau,p}^{(k)}$ into one single value per item $x_{\tau,p}$ and we calculate the overall system capacity W as follows

$$\bar{w}_{\tau,p} = \sum_k \epsilon_k w_{\tau,p}^{(k)}, \quad \text{and} \quad \bar{W} = \sum_k \epsilon_k W^{(k)}, \qquad (29)$$

where ϵ_k is recursively defined as follows

$$\epsilon_k = \left| \epsilon_{k-1} W^{(k)} - \sum_{\tau \in \mathcal{T}} \sum_{\substack{p \in \mathcal{P}_{b,c} \\ \forall b \in \mathcal{B}}} \epsilon_{k-1} w_{\tau,p}^{(k)} \right|, \quad \forall k > 0, \qquad (30)$$

assuming that $\epsilon_0 = 1$. This translates the problem into a classical 0-1 Knapsack problem with one single capacity constraint. Thus,

Algorithm 2 Knapsack-Solver($\bar{W}, \bar{\boldsymbol{w}}$)

```
1:  Initialize $H = 0, C = \{e\}$ where $\{e\} = \{\tau, p\}, \forall \tau, p$
2:  **Calculate** $w_{\tau,p}$ and $W$ based on (29)
3:  $H = \bar{W}$
4:  **for** $e \in C$ **do**
5:      $\phi_{\tau,p} = \frac{\gamma_{\tau,p}}{\bar{w}_{\tau,p}}$
6:  **Sort** $C$ based on $\phi_{\tau,p}$ in a decreasing order
7:  **while** $(H > 0 \wedge |C| > 0)$ **do**
8:      **Pool** the first $e \leftarrow C$
9:      **if** $H - w_{\tau,p} \geq 0$ **then**
10:         $x_{\tau,p} = 1$
11:         $H = H - w_{\tau,p}$
```

Algorithm 3 Knapsack Admission Control (KAC)

```
1:  $k \leftarrow 1$
2:  Initialize $\bar{W} = \varnothing, \bar{\boldsymbol{w}} = \varnothing, \epsilon_0 = 1$
3:  $\boldsymbol{x}^{(k)} \leftarrow$ Knapsack-solver($\bar{W}, \bar{\boldsymbol{w}}$)
4:  **while** $P_{DS}(\boldsymbol{x}^{(k)})$ is unbounded **do**
5:      $\boldsymbol{\mu} \leftarrow$ extreme ray
6:      **Compute** $\boldsymbol{w}^{(k)}$ and $W^{(k)}$ based on (27) and (28)
7:      $\bar{\boldsymbol{w}} = \bar{\boldsymbol{w}} + \epsilon_k \boldsymbol{w}^{(k)}, \quad \bar{W} = \bar{W} + \epsilon_k W^{(k)}$
8:      **Compute** $\epsilon_k$ based on (30)
9:      $\boldsymbol{x}^{(k)} \leftarrow$ Knapsack-solver($\bar{W}, \bar{\boldsymbol{w}}$)
10:     $k \leftarrow k + 1$
11: $\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$
12: $\boldsymbol{y}^*, \boldsymbol{z}^* \leftarrow P_{DS}(\boldsymbol{x}^{(k)})$
```

we compute the ratio $\phi_{\tau,p} = \frac{\gamma_{\tau,p}}{\bar{w}_{\tau,p}}$ per item $x_{\tau,p}$. Based on such ratio, we sort all the items in a decreasing order and we try to fit them into our system capacity $\bar{W}$, following the classical first-fit decreasing (FFD) algorithm [25].

Algorithm 2 is a heuristic that allows us to expedite solutions of Problem 5. Then, by combining Algorithm 2 and removing the optimality cuts from our Benders approach, we can design a fast method to solve our orchestration Problem 2 in larger scale scenarios. We describe such method, descriptively labeled Knapsack Admission Control (KAC), in Algorithm 3.

4.3 Simulation Results

We now evaluate, with emulated data planes from real operators, the revenue gains achievable by our approach under different slice types, traffic patterns and penalties/rewards.

4.3.1 Infrastructure. We consider real urban networks from 3 different operators in Romania (N1), Switzerland (N2) and Italy (N3), shown in Fig. 4(a)-(c). First, we observe that they *do not have canonical structure.* Some BSs are as far as 20Km from the edge CU (in N3), while others are within 0.1Km range. There is therefore high path diversity across networks. N1 has high path redundancy (mean of 6.6 paths), while in N3 several BSs have only 1 path (mean 1.6). As a result, the delay[11] distribution differs across networks. Second, they use heterogeneous link technologies. N3 uses mainly fiber, N2 wireless and N1 fiber, copper and wireless. This induces high diverse link capacities (from 2 to 200 Gb/s). This diversity, illustrated in Fig. 4(d)-(e), evinces that a one-size-fits-all orchestration policy may be arbitrarily inefficient.

[11]Assuming store-and-forward and $12000/C_e$, 4 or 5μs/Km (cable or wireless), and 5μs for transmission, propagation, and processing delay.

Slice type	R	Δ (ms)	Λ (Mb/s)	σ (Mb/s)	$s = \{a, b\}$ (CPUs)
(x)eMBB	1	30	50	variable	$\{0, 0\}$
mMTC	$(1 + b)$	30	10	0	$\{0, 2\}$
uRLLC	$(2 + b)$	5	25	variable	$\{0, 0.2\}$

Table 1: End-to-end network slice template

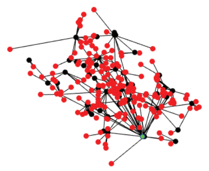
Romania (N1) and Switzerland (N2) have $N = 198$ and $N = 197$ BSs, respectively. We consider $C_b = 20$ MHz for all BSs b that, assuming ideal channel conditions and 2x2 MIMO, yield $\eta_b = 20/150$.[12] Conversely, Italy (N3) has 1497 radio units clustered in 200 groups of 5-10 radio units. We consider each cluster as one BS with capacity equal to the aggregate capacity of the cluster (between $C_b = 80$ and $C_b = 100$ MHz). Finally, we connect the edge CU (green dot in Fig. 4(a)-(c)) with a core CU (not shown in the figure) with a link with unlimited bandwidth and a latency equal to 20 ms. We let the edge CU have a capacity equal to $20N$ CPU cores, i.e., enough capacity to accommodate one mMTC tenant (the more compute-hungry, as we show later) at maximum load, and the core CU have five times as much. Moreover, to ease presentation, we neglect transport overheads and so $\eta_e = 1$.

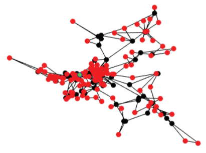
4.3.2 Scenarios. Based on 3GPP guidelines on 5G network design [44], 3 different slice types may be specified in Network Slice Selection Assistance Information (NSSAI): enhanced/extreme Mobile BroadBand (e/xMBB), massive Machine-Type-Communications (mMTC) and ultra reliable low-latency communications (uRLLC). We rely on such 3 heterogeneous slice types to account for diverse delay/throughput requirements, summarized in Table 1. The reward R gained when accepting a tenant differs across slice types to reflect such heterogeneity. Slice requests Φ_τ are generated with a fixed $\boldsymbol{\Lambda}_\tau = \{\Lambda_{\tau,p} = \Lambda \mid \forall p \in \mathcal{P}_{b,c}, \forall b \in \mathcal{B}, \forall c \in C\}$. Then, the actual traffic demand $\boldsymbol{\lambda}_\tau^{(\theta)}$ follows a Gaussian distribution with variable mean $\bar{\lambda}$ and standard deviation σ. The only exception is the mMTC template that has a deterministic load (i.e., $\sigma_{\text{mMTC}} = 0$). Finally, the service model parametrization s is also shown in the table.

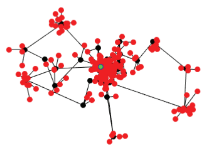
We compare both (`Benders` and `KAC`) against a baseline approach labeled `no-overbooking`. For the latter, we solve the same AC-RR problem but we replace constraint (9) with $\boldsymbol{x\Lambda} \leq \boldsymbol{z}$. As a result, accepted slices upon the `no-overbooking` policy are allocated the amount of resources agreed in their SLA. Note that we use our optimal `Benders` method to find the `no-overbooking` policy and so it is *an upper-bound benchmark.* All slice requests are issued at the beginning of each simulation, which runs until the mean revenue has a standard error lower than 2%. This is almost immediate for `no-overbooking` but it requires longer for our overbooking methods due to the time needed to *learn* slice load patterns.

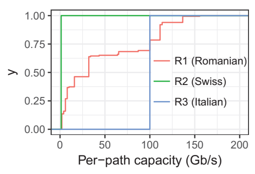
We present results for a variable setting of mean load $\bar{\lambda}$, load variability σ, and penalty $K_\tau = K, \forall \tau$. In our results, depicted in Fig. 5 and 6, different colors represent different penalties such that $K = \frac{m}{\Lambda} R$, where $m = \{1, 4, 16\}$. In this way, if $m = 1$, failing to serve 10% of the SLA incurs in a penalty equal to 10% of the reward payed by the tenant (40% if $m = 4$ and so on). Finally, we set $\sigma = \{0, \bar{\lambda}/4, \bar{\lambda}/2\}$ with different line types (for `Benders`)

[12]We consider ideal conditions to ease the analysis. In practice, however, radio models can be used to make a more accurate estimation.

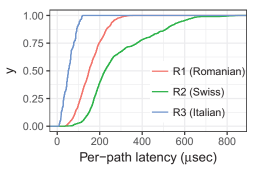

(a) Romanian topology (N1).


(b) Swiss topology (N2).


(c) Italian topology (N3).



(d) Path Capacity Distribution



(e) Path Delay Distribution

Figure 4: (a)-(c): Networks from 3 European operators: red dots indicate the BSs' locations, black dots the routers/switches, and the green dot an edge CU (placed at the most central position). (d)-(e) Path capacity and delay distribution for the 3 networks.

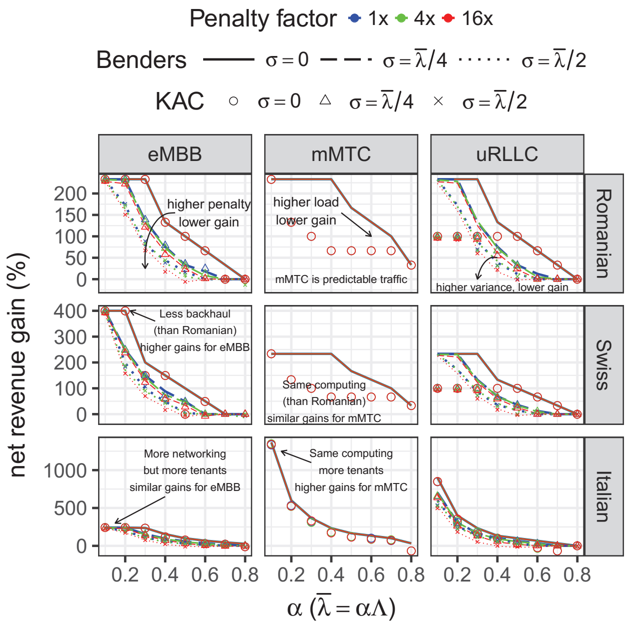


Figure 5: Relative revenue (percentage) of our approaches over `no-overbooking` in homogeneous scenarios. Variable mean load $\bar{\lambda}$.

or shapes (for KAC). We consider a total number of 10 tenants for "Romanian" and "Swiss" and 75 tenants for "Italian" (with more radio and transport capacity). In this way, our simulations span not only realistic topologies but also a wide set of parameters.

4.3.3 Homogeneous cases. We first let all the slices use the same template and have equal (but independent) service demand statistics ($\bar{\lambda}$ and σ). Fig. 5 depicts the relative net revenue gain (percentage) with our approaches and with `no-overbooking` for all slice types and all topologies described above. In the x-axis, we use parameter $0 \leq \alpha \leq 1$ to control the mean load of each slice such that $\bar{\lambda} = \alpha\Lambda$ (e.g., if $\alpha = 1$ the mean load of Φ_τ is equal to Λ_τ).

We note that both KAC and Benders provide equal performance when all slices are eMBB, regardless of the topology. This is remarkable because Benders may take a few hours to converge with some settings whereas KAC boils down this number to a few seconds. In case of mMTC and uRLLC slices, KAC under-performs when compared to Benders, though it still provides between 200% and 75% additional revenue w.r.t. `no-overbooking` in low to medium load regimes. However, as above-mentioned, we use an optimal method to implement `no-overbooking` and it thus suffers from convergence times similar to our optimal method.

Let us focus on the eMBB slices in "Romanian" (top left plot of Fig. 5). In this setting, `no-overbooking` obtains a revenue equal to 3 monetary units irrespective of the conditions of the system (not shown due to space limitations). Regarding our approaches, we obtain up to 220% additional revenue (i.e., up to 10 monetary units) when the mean load is low (relative to the SLA). This is intuitive because the lower the ratio between mean load $\bar{\lambda}$ and Λ, the larger the chances for multiplexing load. The second observation is that, when $\sigma = 0$ (no traffic variability), our approach obtains the same revenue gains independently from the penalty factor imposed. This results in overbooking with no risk as the forecasting process is performed with high certainty. The third due observation is that higher slice load variability leads to less revenue gains. The rationale behind is that higher variability incurs in a higher risk of committing an SLA violation and so our mechanism overbooks more conservatively. Finally, when $\sigma > 0$, higher penalty factors also negatively affect the potential revenue gains due to a conservative behavior.

The net revenue attained to mMTC or uRLLC is higher (up to 30 and 25 units in "Romanian", respectively) due to their higher reward. However, we can observe that the relative gains remain very similar for all slice types in "Romanian". This is not the case for "Swiss", where the maximum gain of eMBB is twice its gain in "Romanian" (and twice the gain for mMTC and uRLLC). The reason is that the transport of "Swiss" is constrained by low-capacity wireless links whereas the computing capacity (used by uRLLC and specially mMTC) remains the same. As a result, `no-overbooking` obtains less net revenue when there are eMBB slices only w.r.t. "Romanian". However, our approaches are capable of accepting more eMBB tenants when their actual load is limited.

Last, "Italian" has considerably more radio and transport resources than both Romania and "Swiss", whereas the computing capacity remains the same. Indeed, `no-overbooking` obtains up to 25 monetary units when all slices are eMBB (8x more than the same scenario in "Swiss" and "Romanian"), and very similar net revenue when slices are mMTC and uRLLC (because they mostly depend on computing, which keeps constant across topologies). Given that we have 75 tenants (instead of 10), the relative gains when applying overbooking are similar for eMBB as in the other topologies. This is due to the fact that increasing radio and transport capacity benefits both `no-overbooking` and our approaches. However, these gains are substantially higher when the mean load of the slices is mild to low with mMTC and uRLLC as computing is severely constrained thereby substantially helping in these load regimes.

Notably, the gains shown in Fig. 5 come at a negligible cost on the tenants. Specifically, a tenant's SLA violation occurred with a

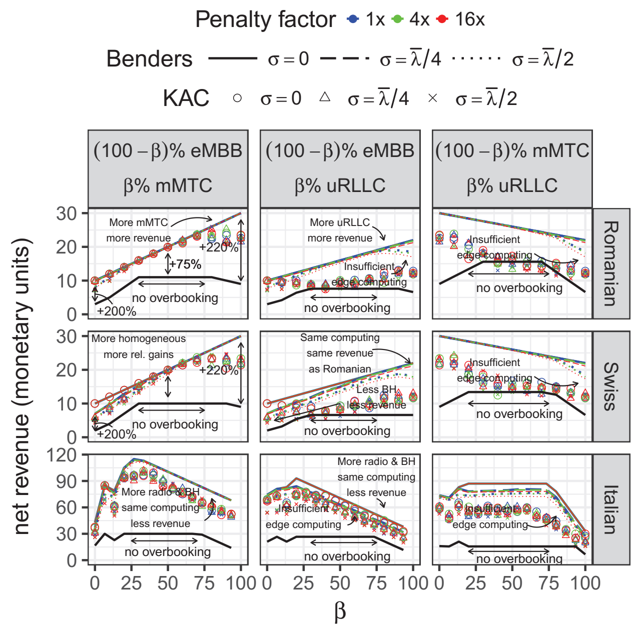


Figure 6: Revenue of our approaches (colors) and no-overbooking (black) in heterogeneous scenarios. Mean load is $\bar{\lambda} = 0.2\Lambda$.

probability lower than 0.0001% in the most aggressive configuration ($\sigma = \bar{\lambda}/2$ and $m = 1$), and even in such rare cases, the dropped traffic is as much as 10%. As a sanity check, a more aggressive overbooking ($\sigma = 3\bar{\lambda}/4$ and $m = 0.01$) increases the chances of violating an SLA to only 0.043% samples with as much as 20% of traffic drop.

4.3.4 Heterogeneous cases. We now consider mixed setups. To simplify the visualization of our results, we focus on scenarios that merge eMBB and mMTC, uRLLC and eMBB, and mMTC and uRLLC slices, respectively, and fix the mean load $\bar{\lambda}_\tau = 0.2 \cdot \Lambda_\tau$. Fig. 6 depicts the net revenue of our approaches and no-overbooking (with a black line) for the same range of σ and penalty parameters m used before. The scenarios have a fix number of slices (10 for "Romanian" and "Swiss", 75 for "Italian") and we vary the percentage of one type of slice w.r.t. the other using parameter β.

First, let us study the top left plot where we have $10\frac{\beta}{100}$ mMTC slices and $10\frac{100-\beta}{100}$ eMBB slices in "Romanian". The revenue attained to no-overbooking grows as we increase the ratio of mMTC tenants until $\beta = 25\%$ onwards when the revenue remains flat. At that point, no-overbooking is not capable of accommodating computing resources to the increasing number of mMTC slices but there are sufficient eMBB slices to compensate. This occurs until $\beta = 75$ where there are not enough eMBB tenants and therefore the revenue falls as computing resources are fully consumed. In marked contrast, our approach obtains a linearly increasing revenue as we increase the number of mMTC slices that are all eventually accepted. Interestingly, the larger relative gains over no-overbooking occurs when the scenario is more homogeneous ($\beta = 0\%$ and $\beta = 100\%$). Similar observations can be obtained from the other two mixes of slice types. We obtain similar revenues also for "Swiss". The main difference is that, given the constrained transport, higher values of σ and higher penalty factors incur in lower revenues compared to the "Romanian" topology.

Compared to "Romanian" and "Swiss", similar revenue trends are observed for no-overbooking but substantially different for our approaches in "Italian" taking the first case (eMBB and mMTC slices). The revenue of both Benders and KAC rapidly grows as we accept more mMTC slices while declining after we reach $\beta = 25\%$. Counter-intuitively, while "Italian" has substantially more radio and transport resources (and more slice requests) than the other two topologies, the computing resources are essentially the same, and there are not sufficient eMBB slices to compensate the rejected mMTC slices from $\beta = 25\%$ onwards. Similar observations can be made for "Italian" in the other two mixes of slices.

Importantly, our overbooking schemes cause SLA violations as often as in the homogeneous case (less than 0.001% samples with the most aggressive configuration) and so our gains come at a negligible cost for the tenants. In this way, we conclude that our system manages to trade off hard SLA guarantees of traditional systems for substantial revenue gains with minimal SLA violations and practically zero footprint from the overbooking scheme.

5 EXPERIMENTAL PROOF-OF-CONCEPT

We evaluate our orchestrator[13] with a real data plane. To this aim, we deploy the experimental testbed depicted in Fig. 7. The hardware components are summarized in Table 2.

In the RAN, we use 2 commercial BSs with RAN sharing support and we use different PLMN-Ids [45] to identify slices due to the lack of 5G network slicing-support equipment. The proprietary interface of the BSs allows us to grant shares of bandwidth, physical radio blocks (PRBs) specifically, to different mobile networks

[13]The algorithm implementation has been carried out using the framework of IBM ILOG CPLEX and its Python API.

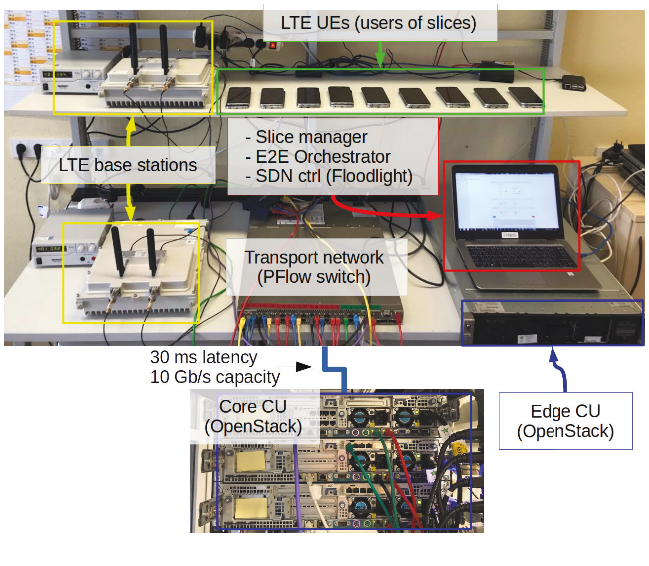
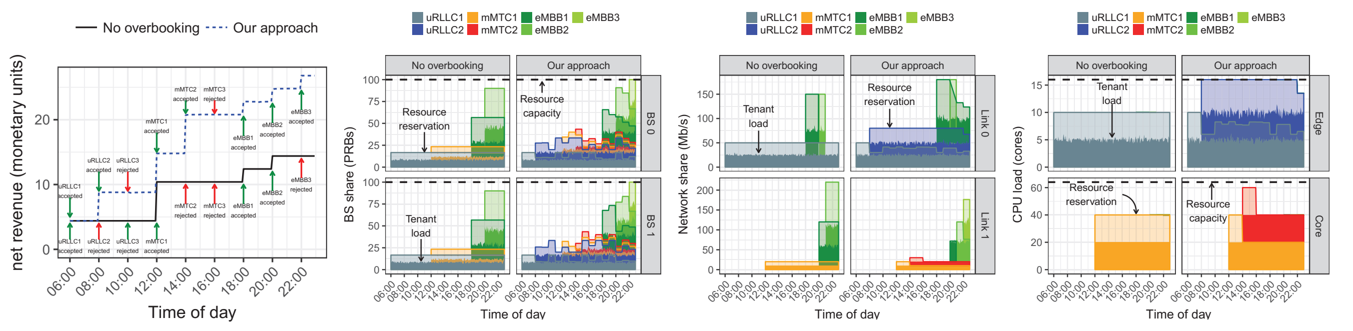


Figure 7: Testbed

Device type	Description	Ref.
vEPCs	OpenEPC Rel. 7 (1x per slice)	[4]
UEs	Samsung Galaxy 7 (1x per slice and BS)	[5]
Transport	OpenFlow 1.5 switch with 48 1-gigabit ports	[3]
RAN	2x 20 MHz NEC small cell with RAN sharing @ band 3	[2]
CU	OpenStack Queens with 16 (Edge) and 64 (Core) CPUs	[1]

Table 2: Detailed HW components in our testbed



(a) Net Revenue. (b) Radio utilization. (c) Transport utilization. (d) Computation utilization.

Figure 8: Net revenue over time (a); and resource reservation and actual utilization across BSs (b), two transport links (c) and both CUs (d), respectively, for 9 heterogeneous slice requests arriving at different times.

(associated with a different PLMN-id).[14] The BSs are set in 20-MHz channels (capacity equal to 100 PRBs). In the transport, we use a programmable `OpenFlow` switch to virtualize the backhaul topology shown in Fig. 1, comprised of 1-Gb/s Ethernet links. For computing, we connect two conventional servers with two 1Gb/s Ethernet links, respectively. The first server has 16 CPU cores and emulate an edge CU; the second has 64 CPU cores and we use `netem` to emulate 30 ms latency in its backhaul link, emulating a core CU. To construct each slice's network service (see Fig. 1), we create a VM instance of `OpenEPC` to connect the slice to the mobile system, a VM with our rate-control middlebox and an additional VM with `mgen` to generate traffic with custom traffic patterns, emulating the VS of the slice. Finally, we use one Android smartphone per slice and BS, connected to the BS with coaxial cables for isolation, to emulate a crowd of UEs receiving traffic from each VS.

We set up a dynamic scenario where slice requests arrive every 2 epochs for a total of 18 epochs (i.e., up to 9 slices). We take one monitoring sample every 5 minutes (which is conventional [31]), and collect 12 samples per epoch (i.e., 1 hour). The first three slice requests "uRLLC1", "uRLLC2", "uRLLC3" are uRLLC (with the parameters described in Table 1), the next three "mMTC1", "mMTC2", "mMTC3" are mMTC and the remaining slices "eMBB1", "eMBB2", "eMBB3" are eMBB. To ease the analysis, we fix the mean load of each slice to be half its Λ (SLA) with a standard deviation equal to 10% of its mean, and a penalty equal to $K = \frac{R}{\Lambda}$ ($m = 1$ in Fig. 5 and 6). We repeat the experiment with our approach (using Benders) and with "no overbooking". The results are summarized in Fig. 8(a)-(d). Fig. 8(a) shows the net revenue per BS of both approaches over time; and Fig. 8(c)-(d) show, with stacked areas, the utilization and the actual reservation made on each domain of the system. For the transport, we selected the two links that connect each CU to the rest of the system to guarantee that any possible path is represented.

The first 3 slice requests (uRLLC) arrive at 6h, 8h and 10h, respectively, requesting an aggregate of 10 CPUs each in the edge CU. While "no overbooking" accepts only "uRLLC1", our mechanism adapts the CPU reservation to the actual load of the slices and thus accepts also "uRLLC2" as shown by Fig. 8(d). This results in twice the revenue we obtain at 10h. The next 3 slice requests are mMTC requesting up to 40 CPUs. Similarly, our approach adapts the CPU reservation to the actual load and allows us to accept an additional slice over "no overbooking", which results in 100% revenue gain at 16h. From this time on, one eMBB slice request arrives every 2h requesting 50 Mb/s service SLA. This forces "no overbooking" to accept only 2 slices at the moment, since some radio resources are already used by uRLLC and mMTC tenants. Conversely, our approach allows us to squeeze one extra eMBB slice, leading at an extra 86% revenue after 22h.

6 RELATED WORK

As a result of the 5G hype, network slicing has recently gained much attention. However, most of the literature focuses on domain-specific issues that leave a significant gap in the design of practical mechanisms for the end-to-end orchestration of network slices. In addition, most research focuses either on analytical work with considerable system assumptions or, conversely, on the design of an orchestration system that neglects formal analysis of optimization models. In our work, we design an end-to-end orchestration system that is feasible in practice and relies on well-grounded optimization methods to make yield-driven decisions, as shown in our simulation and experimental assessments.

A RAN admission controller was presented in [39], an experimental prototype of a slice-capable LTE stack was introduced in [13], a preliminary network slicing orchestration solution was shown in [49], and a radio resource allocation algorithm achieving fairness and isolation among different slices was designed and analyzed in [10]. All these works show that substantial multiplexing gains can be attained by designing a proper radio resources slicing solution.

The key feature to support network slicing is customization of mobile system resources. With this in mind, different studies analyze the slicing of transport and cloud resources. The Virtual Network Embedding (VNE) [18, 51] and Virtual Network Function (VNF) placement problems [8, 17, 40] have become very popular in the last few years. In [37], the authors integrate two well-known NP-hard problems to model the VNF placement problem: a facility location problem and a generalized assignment problem. Later, this framework was extended with real-time constraints [46]. In [47], an approximate Markov-decision-process-based algorithm is designed, and a first approximation algorithm to solve the VNF placement problem is presented in [38]. The works of [8, 17] focus on the orchestration of service function chains in cloud platforms via linear

[14]We use commercial BSs for convenience; however, our approach is a natural fit to open source initiatives such as [13].

programming (LP) relaxation and a heuristic, respectively. In [26], the joint problem of deploying chains of virtual functions and path computation in a distributed cloud is studied. A similar problem is addressed by [21] and [6], where the joint VNF placement and routing problem is considered. These works allow the deployment of multiple instances of the same service chain in case of several traffic flows generated by many distributed nodes. Finally, a service model where datacenter slices are dynamically created over commodity hardware was proposed in [14]. Then, on top of each slice, an on-demand virtualized infrastructure manager (VIM) is instantiated to control the allocated resources.

To summarize, despite the attention that network slicing has received upon the wave of 5G, the design of an orchestration solution that spans across multiple domains of a mobile network and the design of business models that take advantage of it, remain as open challenges. Our work is, to the best of our knowledge, the first attempt to fill this gap.

7 CONCLUSIONS

In this paper, we have presented a novel yield-driven orchestration platform that explores the concept of *slice overbooking*. Notably, our solution is specifically designed for the orchestration of slices end-to-end, across multiple heterogeneous domains of the mobile ecosystem. To this aim, our design is based on a hierarchical control plane that governs multiple domain controllers across a mobile system and uses ETSI-compliant interfaces and data models. Our system embeds a control engine in charge of making (*i*) admission control and (*ii*) resource reservation decisions by exploiting monitoring and forecasting information. Our overbooking mechanism is grounded on an optimization formulation providing provably-performing algorithms that achieve up to 3*x* revenue gains in several realistic scenarios built upon data from three real mobile operators. Finally, we have presented an experimental proof-of-concept that validates the feasibility of implementing our approach with conventional mobile equipment on top of available open-source software.

ACKNOWLEDGEMENTS

We thank our shepherd, Jia Wang, and the anonymous reviewers for their valuable comments and feedback.

This work was supported in part by the H2020 5G-Transformer Project under Grant 761536 and in part by H2020-MSCA-ITN-2015 5G-Aura Project under Grant 675806.

REFERENCES

[1] HP DL380p Gen8 server. https://www.hpe.com/h20195/v2/default.aspx?cc=za&lc=en&oid=5177957.
[2] NEC MB4420. https://uk.nec.com/en_GB/global/solutions/nsp/sc2/.
[3] NEC PFlow. http://www.nec.com/en/global/prod/pflow/pf5240.html.
[4] OpenEPC. http://www.openepc.com/.
[5] Samsung Galaxy S7. http://www.samsung.com/global/galaxy/galaxy-s7/.
[6] Agarwal, S., et al. Joint VNF Placement and CPU Allocation in 5G. In *IEEE INFOCOM* (2018).
[7] Andrews, J. G., et al. What Will 5G Be? *IEEE Journal on Selected Areas in Communications 32*, 6 (Jun. 2014), 1065–1082.
[8] Bari, M. F., et al. Orchestrating virtualized network functions. *IEEE Transactions on Network and Service Management* (2016).
[9] Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Num. Mathematik 4*, 1 (Dec 1962), 238–252.
[10] Caballero, P., et al. Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads. *IEEE/ACM Transactions on Networking* (Oct. 2017).
[11] Chen, B., et al. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access* (2017).
[12] Chu, P., and Beasley, J. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics 4* (Jun. 1998), 63–86.
[13] Foukas, X., et al. Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture. In *ACM MobiCom '17* (2017), pp. 127–140.
[14] Freitas, L., et al. Slicing and Allocation of Transformable Resources for the Deployment of Multiple Virtualized Infrastructure Managers (VIMs). In *IEEE Conference on Network Softwarization* (Jun. 2018).
[15] Garcia-Saavedra, A., et al. Low delay random linear coding and scheduling over multiple interfaces. *IEEE Transactions on Mobile Computing* (Nov 2017).
[16] Garcia-Saavedra, A., et al. Joint optimization of edge computing architectures and radio access networks. *IEEE JSAC* (2018).
[17] Gember, A., et al. Stratos: A network-aware orchestration layer for middleboxes in the cloud. Tech. rep., 2013.
[18] Gong, L., et al. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *IEEE INFOCOM* (Apr. 2014), pp. 1–9.
[19] GSMA. Smart 5G Networks: enabled by network slicing and tailored to customers' needs, Sep. 2017.
[20] Guan, Z., and Melodia, T. The value of cooperation: Minimizing user costs in multi-broker mobile cloud computing networks. *IEEE Transactions on Cloud Computing 5*, 4 (Oct 2017), 780–791.
[21] Gupta, A., et al. A Scalable Approach for Service Chain Mapping With Multiple SC Instances in a Wide-Area Network. *IEEE journal on Selected Areas in Communications 36*, 3 (Mar. 2018), 529–541.
[22] Huang, J., et al. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *SIGCOMM Comput. Commun. Rev. 43*, 4 (Aug. 2013), 363–374.
[23] J. Kuo, *et al.* Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture. In *IEEE INFOCOM* (May 2017).
[24] Kavanagh, A. OpenStack as the API framework for NFV: the benefits, and the extensions needed. *Ericsson Review 2* (2015).
[25] Korf, R. E. A new algorithm for optimal bin packing. In *Eighteenth National Conference on Artificial Intelligence* (2002), American Association for Artificial Intelligence, pp. 731–736.
[26] Kuo, T. W., Liou, B. H., Lin, K. C., and Tsai, M. J. Deploying chains of virtual network functions: On the relation between link and server usage. In *IEEE INFOCOM* (Apr. 2016), pp. 1–9.
[27] Le, F., et al. Understanding the performance and bottlenecks of cloud-routed overlay networks: A case study. CAN '16, ACM, pp. 7–12.
[28] Le, F., Nahum, E., Pappas, V., Touma, M., and Verma, D. Experiences Deploying a Transparent Split TCP Middlebox and the Implications for NFV. HotMiddlebox '15, ACM, pp. 31–36.
[29] Li, X., et al. 5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks. *IEEE Communications Magazine 55*, 8 (2017), 128–137.
[30] Maki, I., et al. Performance analysis and improvement of tcp proxy mechanism in tcp overlay networks. In *IEEE ICC '05* (2005).
[31] Marquez, C., et al. Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage. In *ACM CoNEXT '17* (2017).
[32] Marquez, C., et al. How Should I Slice My Network? A Multi-Service Empirical Evaluation of Resource Sharing Efficiency. In *ACM MobiCom '18* (2018).
[33] Mendelson, H., and Whang, S. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations research 38*, 5 (1990), 870–883.
[34] Nauss, R. The 0 1 knapsack problem with multiple choice constraints . *European Journal of Operational Research 2*, 2 (1978), 125–131.
[35] NGMN Alliance. Description of Network Slicing Concept. *White Paper* (2016).
[36] Qian, F., et al. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *Proceedings of WWW '12* (2012), pp. 51–60.
[37] R. Cohen, *et al.* Near optimal placement of virtual network functions. In *IEEE INFOCOM* (Apr. 2015), pp. 1346–1354.
[38] Sang, Yu , *et al.* Provably efficient algorithms for joint placement and allocation of virtual network functions. In *IEEE INFOCOM* (2017).
[39] Sciancalepore, V., et al. Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization. In *IEEE INFOCOM* (2017).
[40] Sciancalepore, V., Yousaf, F. Z., and Costa-Perez, X. z-TORCH: An Automated NFV Orchestration and Monitoring Solution. *IEEE Transactions on Network and Service Management* (2018).
[41] Singh, S. K., et al. A survey on internet multipath routing and provisioning. *IEEE Communications Surveys Tutorials 17*, 4 (Fourthquarter 2015), 2157–2175.
[42] Subramanian, J., et al. Airline yield management with overbooking, cancellations, and no-shows. *Transportation science* (1999), 147–167.
[43] Taylor, J. W. Exponentially weighted methods for forecasting intraday time series with multiple seasonal cycles. *International Journal of Forecasting* (2010).
[44] Third Generation Partnership Project (3GPP). System Architecture for the 5G System; Stage 2 (Release 15). 3GPP TS 23.501 v15.2.0, June 2018.

[45] Xenakis, D., et al. Mobility Management for Femtocells in LTE-Advanced: Key Aspects and Survey of Handover Decision Algorithms. *IEEE Communications Surveys Tutorials 16*, 1 (July 2014), 64–91.
[46] Yang, L., et al. Network functions virtualization with soft real-time guarantees. In *IEEE INFOCOM* (2016), pp. 1–9.
[47] Z. Han , *et al.* Dynamic virtual machine management via approximate markov decision process. In *IEEE INFOCOM* (Apr. 2016), pp. 1–9.
[48] Zanella, A., et al. Internet of things for smart cities. *IEEE Internet of Things Journal 1*, 1 (Feb 2014), 22–32.
[49] Zanzi, L., et al. Overbooking network slices end-to-end: Implementation and demonstration. In *SIGCOMM '18 (poster and demo session)* (2018).
[50] Zhang, C., and Patras, P. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *ACM Mobihoc '18* (2018), Mobihoc '18.
[51] Zhang, S., et al. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In *IEEE INFOCOM* (2012).