Mohammed, Thaha; Jedari Ghourichaei, Behrouz; Di Francesco, Mario


Published in:
IEEE Internet of Things Journal

DOI:
10.1109/JIOT.2022.3192291

Published: 15/12/2022

Document Version
Peer reviewed version

Please cite the original version:
Abstract—Future wireless networks should meet heterogeneous service requirements of diverse applications, including interactive multimedia, augmented reality, and autonomous driving. The fog radio access network (Fog-RAN) is a novel architecture that enables efficient and flexible allocation of network resources to end users. However, guaranteeing application-specific service requirements while maximizing resource utilization is an open challenge in Fog-RANs. This article proposes a multi-resource Fog-RAN slicing scheme that maximizes network resource utilization and satisfies important economic properties: Pareto-optimality, envy-freeness, and sharing incentive. The proposed solution considers both heterogeneous resources (i.e., bandwidth, storage, and computing) and the different service levels defined in 5G networks. Accordingly, a two-level resource scheduling mechanism is devised to jointly allocate Fog-RAN resources to slices in two stages: a broker allocates resources to slices at fog nodes over a given time window; a slice hypervisor then allocates slice-specific resources at each fog node to users with a much shorter time scale. An extensive evaluation based on real-world datasets demonstrates that the proposed solution significantly increases the monetary gain of service providers, namely, by 32% to 60% compared to the state of the art, including dynamic hierarchical resource allocation and dynamic slicing with proportional allocation.

Index Terms—5G networks, RAN slicing, Fog networking, Resource allocation and scheduling, Heterogeneous resources, Utility maximization, Network economics

I. INTRODUCTION

The number of Internet-connected devices is expected to reach 29.3 billion (three times the world population) by 2030 [1]. More than half of the related traffic will consist of Machine-To-Machine (M2M) communication in the Internet of Things (IoT), represented by emerging applications including video surveillance, health-care and smart transportation [2]. These applications also have heterogeneous requirements, characterized in terms of the different types of services supported by 5G networks [3]: enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC), and massive machine-type communications (mMTC). Therefore, mobile network operators (MNOs) should not only significantly expand their wireless bandwidth capacity, but also provide low-latency and ultra-reliable Internet access to meet the heterogeneous quality of service (QoS) requirements of emerging applications [4]. Although both the core and radio access network (RAN) domains have been evolving, the RAN domain faces the major challenges to fulfill the QoS requirements of such applications [5].

Fundamental improvements on RAN architectures have been introduced to maximize the utilization of limited RAN resources [5]. The cloud radio access network (C-RAN) [6] is a well-known solution to provide flexible yet efficient wireless resource management and allocation to mobile devices. The C-RAN employs a centralized unit (CU) which is connected to radio heads (RUs) through the fronthaul. The CU performs the physical signal processing in a centralized manner by decoupling the user plane from the control plane. RUs include the radio components, the associated amplification/filtering modules, and the antennas. However, performing the functions of the physical layer at the CU requires exchanging data and control information with the RUs. Such an extra message exchange is not efficient and increases latency [5]. The fog radio access network (Fog-RAN) has emerged as a novel architecture to address the shortcomings of the C-RAN [7]. Specifically, the Fog-RAN consists of a fog access point (F-AP) with signal processing, storage and computing capabilities attached to each RU, jointly referred to as a fog node. The main idea is to bring some RAN functionalities and resources to RUs so as to provide end-devices (e.g., smartphones, surveillance cameras, and autonomous vehicles) with reliable and low-latency Internet connectivity [8]. Generally, a fog node includes multiple types of resources (e.g., bandwidth, computing, and storage), which are jointly allocated to devices based on their required QoS, thereby improving the quality of experience.

Network slicing has been recognized as a cost-efficient and flexible technique to allocate the RAN resources in form of self-contained logical network instances to service providers (SPs), such as vertical industries with different QoS requirements [9–11]. Fog-RAN slicing specifically faces several challenges, mainly related to efficient scheduling and allocation of resources at both the CU and RU to users of multiple SPs [12, 13]. On the one hand, the CU monitors the load of slices in a centralized manner, and upscales or downscales slices to increase resource utilization. On the other hand, each RU allocates the local slice resources to UEs, thereby guaranteeing the service level agreement (SLA) between the MNO and SPs. However, the joint scheduling and allocation of multi-resource Fog-RANs involving multiple SPs has received limited attention in the literature [14, 15]. The key consideration in this work is that the architecture of a Fog-RAN is very suitable for hierarchical slicing [16] by using a coarser granularity at the CU which enables monitoring and

1For conciseness, RU is used instead of fog node throughout even though considerations involving caching and computing actually refer to the F-AP.

Table I summarizes the acronyms used in the article.
scaling the network globally; a fine granularity at the RUs to guarantee target service characteristics. However, robust scheduling mechanisms should be designed to achieve efficient resource allocation to SPs with diverse QoS requirements, mainly in terms of bandwidth and latency. For instance, a slice allocated to an eMBB service (e.g., video streaming) may need ultra high-bandwidth with a 10 ms end-to-end latency, whereas a slice delivering an URLLC service (e.g., object detection) requires a low data-rate but a 1 ms end-to-end latency [17]. The joint allocation of multiple Fog-RAN resources to slices with the aim of maximizing the overall resource utilization is already challenging. It becomes even harder when it should also satisfy economic properties of practical importance (i.e., fairness, Pareto-optimality, envy-freeness, and sharing incentive).

This article formulates the Fog-RAN slicing problem in a network with multiple SPs and proposes an efficient solution to serve SPs delivering the different types of generic services in 5G (i.e., eMBB, URLLC or mMTC). Tasks associated to applications belonging to different services are accurately modeled for both downlink (DL) and uplink (UL) scenarios, with special attention to their needs for caching, computing, and streaming data. A multi-resource allocation to Fog-RAN slices is formulated accordingly as a utility maximization problem. Then, the resource utilization and utility (i.e., monetary gain) of SPs are determined based on executing the tasks submitted by UEs through Fog-RAN slices. A joint multi-resource allocation and slice admission scheme for dynamic Fog-RAN slicing is finally proposed, with a two-level mechanism (2L-MRA) to assign Fog-RAN resources first to multiple slices, and then to UEs admitted to the slices to maximize the profit of SPs. 2L-MRA allocates resources to slices at a larger time period and slice resources to users at a finer granularity. In particular, 2L-MRA extends the heterogeneous dominant resource fairness (DRF) [18] level-wise to suit Fog-RANs.

The major contributions of this work are the following.

- It presents a novel Fog-RAN slicing scheme that considers the joint allocation of multiple types of resources to SPs with different QoS requirements in both DL and UL scenarios (Section II).
- A comprehensive model of RAN slicing and UL/DL tasks is introduced by considering the major features of eMBB, URLLC, and mMTC service types (Sections III-IV). The task model characterizes realistic streaming scenarios employing file caching, in which users can also decide to stop receiving data before the end of the stream.
- The joint allocation of multiple types of Fog-RAN resources to multiple slices by an MNO is formulated as a utility maximization problem, where the utility of SPs are characterized in terms of their revenues and costs (Section V).
- The 2L-MRA algorithm is devised to maximize the revenue of SPs in Fog-RAN slicing (Section VI). 2L-MRA is efficient (it runs in polynomial time) and satisfies key economic properties: envy-freeness (i.e., fair division), Pareto-optimality, and sharing incentive.
- An extensive evaluation based on real-world datasets demonstrates that 2L-MRA significantly increases the total utility of SPs, namely, by 32% to 60% compared to other approaches in the state of the art (Section VII).

## II. RELATED WORK

Several works have recently addressed resource allocation and scheduling in 5G networks [19]. Other studies have also considered task offloading scenarios [20, 21]. The rest of this section focuses on the state of the art in the most relevant topics of resource allocation for fog computing scenarios [22] and wireless network slicing [12, 23].

Resource allocation in fog networks has been studied in recent works. Ni et al. [24] propose a dynamic resource allocation scheme by applying priced-timed Petri Nets wherein users select fog resources from pre-allocated resources. Wang and Chen [25] discuss joint optimization of offloading decisions and the allocation of computing resources to minimize latency with a hybrid genetic algorithm. Liu et al. [26] address joint task scheduling and resource allocation with latency constraints through an alternating convex optimization method. Several works consider the joint allocation of communication and computational resources for task offloading under energy or latency constraints [21, 22]. However, they consider simple settings, such as a single resource pool or only one resource type without slicing. In contrast, this article studies multi-resource allocation for Fog-RAN slicing with a focus on UL/DL tasks and the QoS classes in 5G.

Some research has studied resource allocation in wireless network slicing by explicitly targeting user or social utility. Wang et al. [27] aim at maximizing user profit and the social utility of the network to improve resource efficiency. In particular, they consider a single resource for uplink scenarios in the form of virtual network functions. Caballero et al. [28] propose a network slicing game based on Nash equilibrium wherein tenants react to the allocation of other tenants by maximizing their own utility. Instead of considering a base station as a resource, Narmanlioglu and Zeydan [29] apply the same framework to allocate RUs to virtual MNOs. Tran and Le [30] employ a Stackelberg game to model the allocation and pricing of resources for network slicing, so as to capture interactions among access/backhaul SPs and their UEs. However, the

### TABLE I: List of used acronyms.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>eMBB</td>
<td>enhanced Mobile Broadband</td>
</tr>
<tr>
<td>URLLC</td>
<td>Ultra Reliable Low Latency Communications</td>
</tr>
<tr>
<td>mMTC</td>
<td>massive Machine Type Communications</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>CU / RU</td>
<td>Centralized Unit / Radio Unit</td>
</tr>
<tr>
<td>F-AP</td>
<td>Fog Access Point</td>
</tr>
<tr>
<td>MNO</td>
<td>Mobile Network Operator</td>
</tr>
<tr>
<td>SP</td>
<td>Service Provider</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UL / DL</td>
<td>Upload / Download</td>
</tr>
<tr>
<td>DRF</td>
<td>Dominant Resource Fairness</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>TTI</td>
<td>Transmission Time Interval</td>
</tr>
</tbody>
</table>
related solution only considers radio resources in the downlink without discussing scheduling for different service types. Tun et al. [31] address resource allocation in the uplink through a generalized Kelly mechanism and constrained non-linear optimization. Specifically, bandwidth and power are allocated to slices and tenants in a hierarchical fashion. Different from all the works above, this article targets SPs instead of tenants or access providers, which makes the problem substantially different.

Some other research has considered resource allocation in wireless network slicing with a focus on utility of SPs and security. Fu and Kozat [32] introduce an auction-based framework for resource allocation in RAN slices for uplink transmissions. Specifically, SPs bid for radio resources and the network operator regulates the game to ensure the optimal allocation and price. Kamel et al. [33] consider dynamic allocation of radio resource blocks in LTE network slices by using heuristic algorithms for a downlink scenario. Kazmi et al. [34] propose a hierarchical scheme that considers two different allocations for the uplink: between the virtual MNO and UEs as well as between virtual MNOs and infrastructure providers. Different from all these schemes, the solution presented here addresses both uplink and downlink scenarios as well as different types of services. Aijaz [35] leverages reinforcement learning and heuristic algorithms to allocate power and resource blocks in 5G networks, for multiple service classes with both uplink and downlink scenarios. Zanzi et al. [36] introduce network slicing brokering: they leverage smart contracts for infrastructure providers to allocate network resources to intermediate brokers, which in turn re-distribute resources among tenants in a secure manner. Boateng et al. [37] securely maximize the utilities of resource buyers and a seller while balancing QoS satisfaction and resource utilization. Their solution is based on deep Q-networks and blockchain technology. Unfortunately, all the literature discussed above only considers a single type of resource, instead of the more challenging scenario represented by the multiple resources (i.e., bandwidth, storage, and computing) targeted by this article.

A few recent works have addressed multi-resource allocation in 5G network slicing [38]. Fossati et al. [23] target fair allocation of multiple constrained RAN resources to slice instances in critical scenarios. Specifically, they introduce a framework based on ordered weighted average. However, their work does not consider the economic aspects involved in network slicing. Some recent studies have applied auction techniques for resource pricing and the allocation of multiple RAN resources to slices. Jiang et al. [39] introduce an auction-based mechanism for joint optimization of resource utilization and network revenue. In particular, a price auction mechanism determines the selling price, while a network slicing auction scheme maximizes the network revenue for customers. Instead, this work targets the revenue of SPs and focuses on their total utility. Finally, Leconte et al. [13] present a technique that integrates placement of virtual network functions with resource allocation. Specifically, resources are allocated through a tunable utility function based on network bandwidth and cloud processing power. In contrast, this work also considers storage in addition to computing and communication resources.

### III. System Model

This section defines the key elements of the system, the reference network architecture, and the slicing model. Fig. 1 illustrates the system model of Fog-RAN slicing. The most important parameters of the system are summarized in Table II.

**Key entities.** The system comprises: a mobile network operator (MNO) providing the Fog-RAN infrastructure and the related resources (i.e., bandwidth, computing, and storage); a broker, namely, a virtual MNO which is responsible for virtualizing the network resources, as well as performing network slicing; service providers (SPs) that acquire network slices with specific SLAs (i.e., QoS guarantees) from the broker to provide downlink (DL) or uplink (UL) services to their subscribers; and user equipment (UEs) (e.g., smartphones or IoT devices) that subscribe to SPs to receive certain services.

**Network model.** The network operates in $T$ discrete transmission time intervals (TTIs) over $t$ ($1 \leq t \leq T$), each with a length of $\Delta t$ (in the order of milliseconds). The network includes a virtual CU (vCU) pool with radio signaling and processing capabilities, as well as content storage and processing servers. The vCU manages the RAN infrastructure and its resources remotely. The RAN includes multiple RUs which are limited-complexity units, typically realizing only radio-frequency functionalities, and connected to the vCU via fronthaul links. The set of RUs is denoted as $R = \{1, \ldots, R\}$, where $r \in \mathbb{R}$ refers to a single RU. The RUs are connected to the vCU through (fronthaul) fiber-optic links. The average DL and UL link capacities between the vCU and RU $r$ are denoted as $B^{ru}_d$ and $B^{ru}_u$ (bits/s), respectively. The aggregated computing capacity of the vCU is denoted as $P^v$ (CPU-cycles/s). RUs are assumed to support simultaneous DL/UL transmissions, e.g., by using full-duplex technology [40]. The bandwidth allocated to the DL and UL transmissions in RU $r$ is denoted by $B^d_r$ and $B^u_r$, respectively, expressed as the number of physical resource blocks (PRBs). In addition, each RU $r$ is equipped with one fog access point (F-AP), whose...
storage and computing capacities are denoted as $C^r$ (bits) and $P^r$ (CPU-cycles/s) [41], respectively. For simplicity, device-to-device communication between UEs is not considered.

**Broker.** The broker abstracts the network resources (i.e., bandwidth, storage, and computing) of both the vCU and RUs. Furthermore, it leverages network virtualization functions to manage the life-cycle of network slices including the admission, resource allocation, inter-slice isolation, and deallocation. In addition, the broker offers application programming interfaces (APIs) for providers to customize slice allocation to their UEs. Slice hypervisors (located at RUs) periodically (e.g., every 1,000 TTI{s}) report the status of their resource usage and the active UEs to the vCU, based on which the broker can effectively allocate RU resources to slices. The reliability and scalability of the broker can be ensured with standard approaches that apply to network orchestrators in software-defined networks, including partitioning [42] as well as elastic selection or placement [43].

**SPs.** The system includes a set $S = \{S_D\} \cup \{S_U\}$ (is $S_D \cap S_U = \emptyset$) of $S$ SPs. The DL SPs ($S_D$) deliver bandwidth-intensive download services (e.g., eMBB applications), whereas the UL SPs ($S_U$) provide compute-intensive and delay-sensitive upload services (e.g., mMTC applications) to UEs. Thus, both DL and UL SPs jointly require caching and computing resources (besides bandwidth) in the Fog-RAN to deliver their services. For instance, a DL SP offering video delivery or online gaming needs to transcoded video frames to appropriate bitrates before transmitting them to UEs (e.g., to meet the QoS requirements). In contrast, a task offloading UL SP may need caching resources (besides computing) to store the task output (e.g., in object recognition/detection [44]) for future use.

**UEs.** The system also includes a set $I$ of UEs (namely, $I = \sum_{s=1}^{S} I_s$), where $I_s$ denotes the subset of UEs subscribed to SP $s \in S$ and $i_s$ indicates a single UE subscribed to SP $s$. For simplicity, each UE is assumed to subscribe to only one SP ($I_s \cap I_{s'} = \emptyset, \forall s \neq s'$) and to associate with only a single RU, i.e., the one with the highest channel gain at each TTI, as in [45]. The set of UEs of SP $s$ associated with RU $r$ is denoted as $I^r_s$. The binary variable $\psi^r_{is}$ indicates the UE/RU association, where $\psi^r_{is} = 1$ signifies that UE $i_s$ is associated with RU $r$ and $\psi^r_{is} = 0$ if not. The achievable spectrum efficiency of UE $i_s$ through one PRB of the DL channel of RU $r$ is given by:

$$z^r_{is} = \log_2 \left(1 + \frac{p^r_{is}g^r_{is}}{\sigma^2 + I^r} \right)^2$$

(1)

where $p^r_{is}$ and $b^r_{is}$ denote the transmission power and channel gain (including path loss and shadowing) of UE $i_s$ in RU $r$, respectively [46]. In addition, $\sigma^2$ denotes the background noise variance and $I^r$ the inter-cell interference of other RUs with RU $r$. Accordingly, the DL rate of UE $i_s$ in RU $r$ is:

$$r^r_{is} = \alpha^r_{is}B^r z^r_{is}$$

(2)

where $\alpha^r_{is} \in [0, 1]$ is the fraction of the DL bandwidth in RU $r$ allocated to UE $i_s$. Similar to Eq. (2), the UL rate of UE $i_s$ in RU $r$ is $r^r_{is} = \alpha^r_{is}B^r z^r_{is}$, where $\alpha^r_{is} \in [0, 1]$ is the fraction of the UL bandwidth in RU $r$ allocated to UE $i_s$. Furthermore, the processing capacity of UE $i_s$ is defined as $p^r_{is} = \varphi^r_{is} P^r$, where $\varphi^r_{is} \in [0, 1]$ denotes the fraction of the computing capacity in RU $r$ allocated to UE $i_s$. The location and rate of UEs are assumed to be constant during each TTI, even though they may change across different TTIs (e.g., due to UE mobility). The task arrival for each UE $i_s$ is modeled according to a Poisson process, similar to [20]. Each task is considered atomic and cannot be divided into subtasks.

**Slices.** There are $S$ elastic Fog-RAN slices ($|S| = |S|$) in the system, where a single slice $\hat{s} \in \hat{S}$ is allocated to each SP $s \in S$. Each slice $\hat{s}$ jointly acquires bandwidth, caching, and computing resources in each RU, depending on its service requirements. Following the major service types defined in 5G networks [23], three types of slices are defined in the network: eMBB ($\hat{s}_{e}$), URLLC ($\hat{s}_{u}$), and mMTC ($\hat{s}_{m}$), whose main characteristics are described in Table III. Each type of slices has certain latency requirements (e.g., applications may require the end-to-end latency of 1-10 ms or even shorter [17]). Accordingly, $\epsilon_e$, $\epsilon_u$, and $\epsilon_m$ indicate the number of slots to be scheduled at each TTI for the instances of eMBB, URLLC, and mMTC slices, respectively. Furthermore, $\zeta_{err}$, $\zeta_{urr}$, and $\zeta_{mrr}$ denote the number of minimum UEs that should be admitted by RU $r$ in slice classes $\hat{s}_e$, $\hat{s}_u$, and $\hat{s}_m$, respectively.

The size of slice $\hat{s}$ is described by the tuple $|\hat{s}| = \langle B_{\hat{s}}, b_{\hat{s}}, C_{\hat{s}}, \zeta_{\hat{s}}, P_{\hat{s}} \rangle$, where $\hat{s}_e$, $\hat{s}_u$, and $\hat{s}_m$ are used interchangeably in the rest of the article.

**TABLE II: Summary of key notation.**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R, r$</td>
<td>Set of RUs in the network and a single RU</td>
</tr>
<tr>
<td>$S = (S_D) \cup (S_U)$</td>
<td>Set of SPs in the network (DL SPs as well as UL SPs)</td>
</tr>
<tr>
<td>$I_s, \mathbb{T}_s$</td>
<td>Set of UEs in the network and UEs subscribed to SP $s$</td>
</tr>
<tr>
<td>$\alpha^r_{is}$</td>
<td>A single UE subscribed to SP $s$</td>
</tr>
<tr>
<td>$\psi^r_{is}$</td>
<td>UEs subscribed to SP $s$ in RU $r$</td>
</tr>
<tr>
<td>$B^r, B^{ restless}$</td>
<td>Mean DL and UL bandwidth capacities of vCU</td>
</tr>
<tr>
<td>$\varphi^r_{is}$</td>
<td>Aggregated computing capacity of vCU</td>
</tr>
<tr>
<td>$P^r$</td>
<td>Bandwidth allocated to DL and UL by RU $r$</td>
</tr>
<tr>
<td>$C^r, P^r$</td>
<td>Caching and processing capacities of RU $r$</td>
</tr>
<tr>
<td>$\psi^r_{is}$</td>
<td>Binary variable for the association of UE $i_s$ to RU $r$</td>
</tr>
<tr>
<td>$z^r_{is}$</td>
<td>Spectral efficiency of DL channel of RU $r$ for UE $i_s$</td>
</tr>
<tr>
<td>$t^r_{is}$</td>
<td>DL transmission rate of UE $i_s$ in RU $r$</td>
</tr>
<tr>
<td>$\alpha^r_{is}$, $\alpha^{ restless}_{is}$</td>
<td>DL and UL bandwidth allocated to UE $i_s$ at RU $r$</td>
</tr>
<tr>
<td>$\psi^r_{is}$, $\psi^{ restless}_{is}$</td>
<td>Processing and storage resources allocated to UE $i_s$ at RU $r$</td>
</tr>
<tr>
<td>$P^r_{is}$</td>
<td>Processing capacity of UE $i_s$ at RU $r$</td>
</tr>
<tr>
<td>$b^r_{is}$, $c^r_{is}$, $P^r_{is}$</td>
<td>Bandwidth, caching, and processing resources allocated to slice $\hat{s}$ at RU $r$</td>
</tr>
<tr>
<td>$\chi^r_{is}$, $\chi^{ restless}_{is}$</td>
<td>Resource allocation and demand of slice $\hat{s}$ at RU $r$</td>
</tr>
<tr>
<td>$C^r$</td>
<td>Resource capacity vector of RU $r$</td>
</tr>
<tr>
<td>$p^r_{is}$, $c^r_{is}$, $P^r_{is}$</td>
<td>Minimum bandwidth, caching, and processing resources allocated to UE $i_s$ in slice $\hat{s}$</td>
</tr>
<tr>
<td>$\alpha_r$</td>
<td>Binary variable indicating slice/RU/UE admission</td>
</tr>
<tr>
<td>$b^r_{is}$</td>
<td>Binary variable indicating caching file $f$ at RU $r$</td>
</tr>
<tr>
<td>$V^f$</td>
<td>Viewer retention of file $f$ by UE $i_s$</td>
</tr>
<tr>
<td>$P^f$</td>
<td>Abandon probability of file $f$ at RU $r$</td>
</tr>
<tr>
<td>$\epsilon_r$, $\epsilon_r^*$</td>
<td>Cost of executing tasks $\tau^r_{i,s}$ in TTI $t$</td>
</tr>
<tr>
<td>$\tau^r_{i,s}$, $\tau^r_{i,s}^*$</td>
<td>Revenue of executing tasks $\tau^r_{i,s}$ and $\tau^r_{i,s}^*$ in TTI $t$</td>
</tr>
<tr>
<td>$\tau^r_{i,s}$, $\tau^r_{i,s}^*$</td>
<td>Resource and computing tasks $\gamma^r_{i,s}$ and $\gamma^r_{i,s}^*$ in TTI $t$</td>
</tr>
<tr>
<td>$\gamma^r_{i,s}$, $\gamma^r_{i,s}^*$</td>
<td>Dominant resource of slice $\hat{s}$ and UL $i_s$ at slice $\hat{s}$</td>
</tr>
<tr>
<td>$\delta^r_{is}$</td>
<td>Dominant share of slice $\hat{s}$ and UE $i_s$ at slice $\hat{s}$</td>
</tr>
</tbody>
</table>
TABLE III: Slice classes and their characteristics.

<table>
<thead>
<tr>
<th>Slice class (s)</th>
<th>Number of slots per TTI</th>
<th>Min. UEs supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMBB (β_e)</td>
<td>ε_e = 2^m_e, ∀ μ_e ∈ 2^{m_e}</td>
<td>(\zeta_{e,i}, \forall r \in R)</td>
</tr>
<tr>
<td>URLLC (β_i)</td>
<td>ε_u = 2^m_u, ∀ μ_u ∈ 2^{m_u}</td>
<td>(\zeta_{ur}, \forall r \in R)</td>
</tr>
<tr>
<td>mMTC (β_m)</td>
<td>ε_m = 2^m_m, ∀ μ_m ∈ 2^{m_m}</td>
<td>(\zeta_{mr}, \forall r \in R)</td>
</tr>
</tbody>
</table>

\[ b^*_s, c^*_s, \text{ and } p^*_s \] (respectively) denote the bandwidth, caching, and computing resources allocated to slice \(s\) in RU \(r\). Here, \(b^*_s\) refers to either the DL or UL bandwidth, according to the type of SP \(s\). The resources allocated to each slice \(s\) from RU \(r\) are represented through the vector \(\mathbf{A}^*_s = (a^*_s, a^*_s, a^*_s, a^*_s)^T\). An allocation \(\mathbf{A}^*_s\) is feasible if RU \(r\) has sufficient resources to satisfy the request (i.e., \(\mathbf{A}^*_s \leq \mathbf{C}^r\)), where \(\mathbf{C}^r = (c^r_s, c^r_s, c^r_s, c^r_s)^T\) expresses the maximum bandwidth, caching, and computing capacities of RU \(r\). The total resources allocated to slices at a given RU \(r\) should not exceed the capacity of RU \(r\), i.e., \(\sum_{s \in S} \mathbf{A}^*_s \leq \mathbf{C}^r\). Accordingly, the total resources allocated to slice \(s\) from all RUs should not exceed the cumulative capacity of all \(r \in R\), i.e., \(\sum_{s \in S} \sum_{r \in R} \mathbf{A}^*_s \leq \sum_{r \in R} \mathbf{C}^r\).

The minimum resources allocated to UE \(i_s\) (once admitted to slice \(s\)) are denoted as the tuple \(i_s = (b_{i_s}^m, c_{i_s}^m, p_{i_s}^m)\), where \(b_{i_s}^m, c_{i_s}^m, p_{i_s}^m\) are the bandwidth, caching, and computing resources. The tuple signifies the SLA between the SPs and their subscripted UEs. Therefore, UE \(i_s\) is not admitted to slice \(s\) unless its minimum resource requirements \(i_s\) are guaranteed. The binary variable \(d_{i_s}^{(t)}\) indicates the UE/slice/RU admission, where \(d_{i_s}^{(t)} = 1\) implies that UE \(i_s\) is admitted to slice \(s\) in RU \(r\) at TTI \(t\), and its tasks are served by slice \(s\) through RU \(r\). If the slice resources at an RU exceed the minimum requirements of active UEs, they are proportionally allocated to the UEs. The maximum number of UEs that can be admitted to slice \(s\) at RU \(r\) is finally given by:

\[ |I_s^r|_{\text{min}} = \min \left\{ \left( \frac{b^*_s}{b_{i_s}^m}, \frac{c^*_s}{c_{i_s}^m}, \frac{p^*_s}{p_{i_s}^m} \right) \right\} \]  

and the maximum number of UEs that can be admitted to slice \(s\) in the whole network is \(\sum_{r=1}^R |I_s^r|_{\text{min}}\).

IV. DOWNLINK AND UPLINK TASK MODELS

This section models the properties of the DL and UL tasks submitted by UEs to SPs and explains how they are served.

**DL tasks.** A DL task includes simultaneous processing and transmission of streaming data (e.g., audio and video files) from the vCU to a UE [2]. The DL SPs publish a common set \(F\) of file segments, where the size of file \(f \in F\) is \(l_f\) (bits). All the files are initially stored at the vCU; the most popular ones (i.e., those with the highest request probability) are proactively cached at each RU to locally serve the maximum number of UEs’ requests through F-APs. The popularity of the files in \(F\) is assumed to be known in advance (e.g., through data analysis and learning techniques [47]). Specifically, the popularity of the \(f\)-th ranked popular file in \(F\) is described through the Zipf distribution [48] as \(p_f = f^{-\gamma} / \sum_{j=1}^F j^{-\gamma}\), where \(p_f \in [0, 1]\) and \(\sum_{f=1}^F p_f = 1\). Here, \(\gamma\) expresses content re-usability; \(\gamma = 0\) implies that the files in \(F\) are requested with the same frequency, and \(\gamma\) near or above one implies that few files are requested more frequently. The binary variable \(k_f^r\) indicates file caching, i.e., \(k_f^r = 1\) if file \(f\) is cached at RU \(r\). The cache space in RU \(r\) is divided into two segments, where \(C^{rd}\) and \(C^{ru}\) (\(C' = C^{rd} + C^{ru}\)) denote the cache size allocated to the DL and UL services, respectively. Accordingly, caching is subject to the constraint \(\sum_{f=1}^F k_f^r l_f \leq C^{rd}\). The tuple \(\tau^{d,(t)}_{i_s} = (f, q_f, l_f)\) describes a DL task submitted by UE \(i_s\) \((s \in S_D)\) at TTI \(t\) to download file \(f\), where \(q_f\) is the number of CPU-cycles required for processing file \(f\) and \(l_f^r\) is the size of the output file to be downloaded by UE \(i_s\). Here, \(q_f\) is assumed to be proportional to \(l_f\) (i.e., \(q_f = \gamma f\)), where \(\gamma\) is the processing density of file \(f\) (CPU-cycles/bit) [49].

The completion time of task \(\tau^{d,(t)}_{i_s}\) depends on whether file \(f\) is hit in RU \(r\) or not. If file \(f\) is hit in RU \(r\), then task \(\tau^{d,(t)}_{i_s}\) is served by RU \(r\), and its completion time is:

\[ t_f^r(\tau^{d,(t)}_{i_s}) = \max \left\{ \frac{l_f}{p_{is}^r}, \frac{l_f}{r_{is}^r} + \Delta t \right\} + \Delta t \]  

where \(l_f/p_{is}^r\) and \(l_f/r_{is}^r\) are the total processing and transmission time of task \(\tau^{d,(t)}_{i_s}\), respectively. If file \(f\) is not hit in RU \(r\), then task \(\tau^{d,(t)}_{i_s}\) is served by the vCU (through RU \(r\)), and its completion time is:

\[ t_f^v(\tau^{d,(t)}_{i_s}) = \max \left\{ \frac{l_f}{p_{is}^{rv}}, \max \left\{ \frac{l_f}{p_{is}^{rv}}, \frac{l_f}{r_{is}^{rv}} \right\} + \Delta t \right\} + \Delta t \]  

where \(p_{is}^{rv} \in [0, 1]\) and \(p_{is}^{rv} \in [0, 1]\) denote the fraction of the processing and bandwidth resources in the vCU allocated to UE \(i_s\) through RU \(r\), respectively.

Optimal scheduling of DL tasks should minimize unnecessary transmission of streaming data, particularly, portions of video that are not actually utilized by UEs. To achieve this, the concept of abandon probability is introduced as the likelihood that a UE interrupts the streaming of a certain file [50]. A streaming file \(f\) has \(T_f\) segments with fixed viewing intervals equal to a TTI. Then, the abandon probability of UE \(i_s\) for file \(f\) at each file segment \(j\) is \(P_j^f(j) = V_j^{frd} - V_j^{frd}\), where \(V_j^{frd}\) denotes the joint probability of UE \(i_s\) viewing the file \(f\) from start to the \(j\)-th TTI (the \(j\)-th file segment), i.e., \(V_j^{frd} = \prod_{k=1}^j p_k(X=1), \forall j \in [1, T_f], i_s \in \mathbb{I_s}, X \in \{0, 1\}\). The expected amount of unused data downloaded when a UE starts playing a video at segment (TTI) \(j\) and interrupts it before the ending segment \(T_f\) is derived as follows [50]:

\[ \mathbb{E}[D_u(j, T_f)] = \sum_{k=1}^\omega P_j^f(j + k) k [r_{js}^{rd} - r_e] \]  

where \(r_e\) is the encoding rate of streaming file, \(r_{js}^{rd}\) is the download rate for the given file for UE \(i\) and slice \(s\), and \(\omega = \lfloor (T_f - t_r)/v_{frd}^r \rfloor \). Here, \(\omega\) is the last file segment which is viewed and downloaded simultaneously. The overall download schedule reduces when \(T_f > t_r\) (here, \(t_r\) is the abandonment threshold), leading to a lower \(\mathbb{E}[D_u(j, T_f)]\) which allows to estimate a new file size (as opposed to \(l_f\)) in Eqs. (4) and (5).
UL tasks. UL tasks are characterized based on the requirements of real-world compute-intensive or delay-sensitive UL applications. In this context, $\tau_{is}^{u,(t)} = \{d_{is}, q_{is}, c_{is}\}$ describes the UL task submitted by UE $i_s$ ($s \in S_d$), where $d_{is}$ is the size of input data (in bits), $q_{is}$ is the number of CPU-cycles required to complete the task, and $c_{is}$ is the size of the storage space required to cache the output for possible future use. Similar to DL tasks, $q_{is} = d_{is} \eta_{is}$, where $\eta_{is}$ denotes the processing density (CPU-cycles/bit) of task $\tau_{is}^{u,(t)}$. The execution time of task $\tau_{is}^{u,(t)}$ running locally at UE $i_s$ is:

$$t^u(\tau_{is}^{u,(t)}) = \Delta t \ q_{is}$$

where $q_{is}$ (CPU cycles/s) is the computation capacity of UE $i_s$. If instead task $\tau_{is}^{u,(t)}$ runs at RU $r$, its offloading time is:

$$t^r(\tau_{is}^{u,(t)}) = \Delta t \ d_{is} r + \frac{q_{is}}{\beta_{is} P_r}$$

where the terms on the right side of Eq. (8) indicate the transmission and execution time of task $\tau_{is}^{u,(t)}$, respectively. Once the execution of task $\tau_{is}^{u,(t)}$ is over, its outcome is cached at RU $r$. Generally, the QoE of UE $i_s$ improves if $t^r(\tau_{is}^{u,(t)})$ is considerably shorter than $t^u(\tau_{is}^{u,(t)})$. Assuming that the size of the output files of the UL tasks are very small compared to the input files [51], the delay of transmitting the output files to UEs is considered negligible.

V. UTILITY MODEL AND PROBLEM FORMULATION

This section first defines the utility of SPs for the case of UEs running their DL/UL tasks through Fog-RAN slices. It then formulates multi-resource allocation to Fog-RAN slices as a utility maximization problem.

The utility of SPs is expressed in terms of the difference between the revenue and cost of the DL/UL tasks submitted by their UEs for execution through the Fog-RAN slices. Without loss of generality, the utility of a task is considered in a single TTI $t$. The revenue ($T$) of task $\tau_{is}^{d,(t)}$ is defined in terms of the delay saving when it is executed through the slice. The rationale behind this choice is that SP $s$ charges UE $i_s$ due to its extra downloading within time period $\Delta t$. Specifically, the price in $\$ for downloading 1 MB extra data during $\Delta t$ is referred to as PCT. The utility of SP $s$ from executing tasks $\tau_{is}^{d,(t)}$ and $\tau_{is}^{u,(t)}$ within TTI $t$ (respectively) are derived as:

$$U^{d,(t)}_{isr} = T_r(\tau_{is}^{d,(t)}) - T_r(\tau_{is}^{d,(t)}) \Phi_s - \delta_s$$

$$U^{u,(t)}_{isr} = T_r(\tau_{is}^{u,(t)}) - T_r(\tau_{is}^{u,(t)}) \Phi_s - \delta_s$$

The joint allocation of multiple Fog-RAN resources to slices is formulated next as a utility maximization problem.

Problem 1 (Multi-Resource Allocation to Fog-RAN Slices). The Multi-Resource Allocation to Fog-RAN Slices (MRAFS) problem is defined as:

$$\max_{\{\psi, \alpha, \varphi, \delta\}} \sum_{s \in S_D} \sum_{r \in R} \sum_{i \in I_r} U^{d,(t)}_{isr} + \sum_{s \in S_d} \sum_{r \in R} \sum_{i \in I_r} U^{u,(t)}_{isr}$$

subject to:

$$\sum_{s \in S} \psi_{is} \leq 1, \forall i \in I_s, s \in S$$

$$\sum_{s \in S} \varphi_{is} \leq 1, \forall i \in I_s, s \in S$$

$$\sum_{s \in S} \delta_{is} \leq 1, \forall x \in \{u, d\}, r \in R$$

$$\sum_{s \in S} \psi_{is} \varphi_{is} \leq 1, \forall r \in R$$

$$\sum_{s \in S} \psi_{is} \delta_{is} \leq C^d, \forall r \in R$$

$$\sum_{s \in S} \sum_{i \in I_r} \psi_{is} \delta_{is} \varphi_{is} \leq C^u, \forall r \in R$$

The related optimality gap will be discussed in Section VII-B.
VI. TWO-LEVEL MULTI-RESOURCE SLICING

This section presents a two-level multi-resource allocation (2L-MRA) mechanism for dynamic Fog-RAN slicing, followed by an analysis of its efficiency and economic properties.

A. Overview of 2L-MRA

A two-level scheduling approach is proposed to allocate the network resources to slices and then to the UEs in each slice. Such an approach leverages time windows composed of multiple TTIs (Fig. 2). Scheduling takes place at two levels: at the vCU and at the RU. In the CU-level scheduling, the broker dynamically allocates the resources of RUs to active slices at the beginning of every time window \( w \) with duration \( W \), such that \( 0 \leq W < T \) and \( \Delta t < W \). Next, in the RU-level scheduling, the slice hypervisors (at each RU) allocates the resources of each slice to its active UEs at the beginning of every TTI \( t \) within each time window \( w \). The TTIs have a short time scale, with a duration \( \Delta t \) in the order of milliseconds. In contrast, time windows have a significantly longer time scale, with a duration \( W \) that is at least one order of magnitude higher than \( \Delta t \). CU-level and RU-level scheduling provide inter-slice and intra-slice isolation, respectively.

2L-MRA leverages the dominant resource fairness (DRF) framework [53] for resource allocation, a generalization of the max-min fairness rule to heterogeneous resources (e.g., in cloud servers) [54]. Accordingly, the dominant resource \( \tau^*_s \) for each UE is defined as the most heavily required resource for executing a task or availing a service (i.e., \( \arg\max_{x \in \{s^*_b,c^*_b,p^*_b\}} (d_x^s) \)), where \( d_x^s \) denotes the required resource \( x \) for a UE. Furthermore, the dominant share for each UE is defined as the fraction of the dominant resources allocated to a UE. DRF aims to find a maximum allocation that equalizes the dominant share of each UE (i.e., it applies max-min fairness across the dominant share of the UEs).

2L-MRA maximizes the utility for SPs while guaranteeing UEs’ minimum resource requirements while satisfying the following properties:

- Polynomial time complexity (PTC): the algorithm should allocate resources in polynomial time.
- Envy-freeness (EF): a UE (and a slice) should not prefer the allocation vector of another UE (and a slice) to its own allocation when obtaining any given service from an SP.
- Pareto optimality (PO): it should not be possible to increase the resource of a UE (and a slice) without decreasing the resource for other UEs (and slices).
- Slice sharing incentive (SSI): each slice should be able to serve more UEs compared to a uniform allocation where each slice is assigned the same amount of RU resources.

The details of 2L-MRA are presented next.

B. 2L-MRA: Operations and Key Properties

As briefly mentioned, 2L-MRA allocates network resources to the slices in two phases, one operating at the CU level and another at the RU level. These phases are illustrated in Algorithm 1 and explained next.

**CU-level scheduling.** Initially \((w=0)\), the broker initializes the demand vector \( D^r_s = (d^s_{b|\theta r}, d^s_{c|\theta r}, d^s_{p|\theta r}) \) for each RU \( r \) by using the minimum value of supported UEs \( \{s^r_{b}, s^r_{c}, s^r_{p}\} \) based on the slice classes. The demands \( d^s_{b|\theta r}, d^s_{c|\theta r}, \) and \( d^s_{p|\theta r} \) are initialized as \( \sum_{i=1}^{\hat{b}} \psi^s_{i|\theta} = \beta^{b|\theta} \), \( \sum_{i=1}^{\hat{c}} \psi^s_{i|\theta} = \beta^{c|\theta} \), and \( \sum_{i=1}^{\hat{p}} \psi^s_{i|\theta} = \beta^{p|\theta} \), respectively. At the beginning of each time window \( w \) (for \( w>0 \)), the broker estimates the resource demand \( \bar{d}^s_{sm}(w) \), which is sent to the broker at every TTI. Then, the broker initializes the allocation matrices for each \( s \in \mathbb{S} \) such that \( \bar{d}^s_{sm}(w) \rightarrow \infty \) is a parameter such that \( \lim_{w \rightarrow \infty} \alpha(w) = 0 \) for any window \( w \), which ensures that \( \bar{d}^s_{sm}(w) \) is the time average demand rate for slice \( s \) at RU \( r \) at \( w \rightarrow \infty \). The actual demand in the previous window \( \bar{d}^s_{sm}(w-1) \) is obtained based on usage statistics and the demand of the resources for each \( s \) at each RU \( r \), which is sent to the broker at every TTI. Then, the broker initializes the allocation matrices for each \( r \in \mathcal{R} \), \( \mathbf{A}^s_{r} \in \mathbb{R}^{[s] \times 3} \) with zero, whose row \( \mathbf{a}^s_{r} = (a^s_{b|\theta r}, a^s_{c|\theta r}, a^s_{p|\theta r}) = (\alpha^s_{r}, \beta^s_{r}, \gamma^s_{r}) \) indicates the fractional amount of both the DL and UL bandwidth, caching, and processing resources allocated to slice \( s \) at RU \( r \). Next, the broker allocates the resources by considering the DRF, through which it determines the fractional resources \( \mathbf{a}^s_{r} \) to be assigned to each slice \( s \) at RU \( r \). Specifically, the broker jointly allocates resources across all the slices at RU \( r \) with the aim of achieving max-min fair allocation for the dominant resources \( \tau^*_s \). Lemma 1 discusses the relationship between a dominant share \( \tau^*_s \) at a slice \( \hat{s} \) and the remaining demand for a slice \( s \) that receives the demands of a slice, and a non-wasteful resource allocation to a slice.

**Lemma 1.** An allocation \( \mathbf{A}^s_{s} \) is not-wasteful for a slice \( \hat{s} \) and RU \( r \) if and only if there exists a \( \alpha^s_{\hat{s}} \) such that \( \mathbf{A}^s_{s} = \alpha^s_{\hat{s}} \cdot \mathbf{d}_s^m \), where \( \alpha^s_{\hat{s}} = \min_{x \in [\hat{s}^*_b, \hat{s}^*_c, \hat{s}^*_p]} (d_x^s) / d_{\hat{s}x}^s \) is the dominant share of the UEs. Furthermore, \( \mathbf{d}_s^m \) is the normalized demand, i.e., \( \mathbf{d}_s^m = \begin{pmatrix} \frac{d_{b|\theta}}{d^s_{b|\theta}} \cdot \frac{d_{c|\theta}}{d^s_{c|\theta}} \cdot \frac{d_{p|\theta}}{d^s_{p|\theta}} \end{pmatrix} \) and \( d_{\hat{s}x}^s \) is the demand of the dominant resource \( \tau^*_s \) of slice \( \hat{s} \).
Based on Lemma 1, the broker solves the following problem at each window:

\[
\begin{align*}
\max_{\{o^r_s\}} & \quad \min_{s \in S} o^r_s \\
\text{s. t.} & \quad \sum_{s \in S} o^r_s d_{s,x} \leq 1, \quad \forall x \in \{b^r_s, c^r_s, p^r_s\}, r \in R \\
& \quad w^r_{s,i}d_{s,i,x} = w^r_{s,j}d_{s,j,x}, \quad \forall s \in S, \quad \{i,j\} \in \mathbb{Z}^+ 
\end{align*}
\]  

where Eq. (13a) states that resources are allocated in proportion to the slice’s demands and should not exceed the (normalized) maximum capacity, which leads to a non-wasteful allocation. Furthermore, Eq. (13b) signifies the fairness in the equalized dominant share. The resource allocation for slice \(s\) at RU \(r\) is \(A^r_s = (\alpha^r_s, \varphi^r_s, \varphi^r_s) = o^r_s \cdot d_{s,r}^i\). The broker allocates each slice the maximum dominant share \(o^r_s\) under the fairness and capacity constraints. Progressive filling\(^3\) is employed to solve Eq. (13).

**RU-level scheduling.** Once the broker determines the allocation matrix \(A^r_s\) for each slice \(s\) at RU \(r\) (at the beginning of each window), the slice hypervisor receives the resource demands \(d^r_{s,i} = i = c \cdot b^r_s \cdot c^r_s \cdot p^r_s\) (namely, the minimum resources required for slice admission) from each UE \(i\) at the beginning of every time slot within TTI \(t\). The slice hypervisors at each RU \(r \in R\) initialize the UE resource allocation matrices \(A^i_{rs} \in R^{R \times |S|}\) with zeros, where row \(a^i_{rs} = (\alpha^r_s, \alpha^r_s, \varphi^r_s, \varphi^r_s)\) determines the fractional resources assigned to UE \(i\) at RU \(r\) for the slice \(s\). Similar to CU-level scheduling, each slice hypervisor at \(r \in R\) jointly allocates the resources to all the UEs admitted to the slice while achieving the max-min fair allocation for the dominant resource. The scheduling interval for the three classes of slices (i.e., URLLC, mMTC, and eMBB) for each SP depends on the number of slots per TTI \(\mu_i\), with \(\mu \in \{e, u, m\}\), according to Table III. This reflects the completion times allowed by the different services. Next, Lemma 2 discusses the relationship between a dominant share each UE receives, resource demands of a UE, and a non-wasteful resource allocation to a UE.

**Lemma 2.** An allocation \(a^i_{rs}\) is non-wasteful for a UE \(i\) and slice \(s\) at \(r\) if and only if there exists a \(v^r_{s,i}\) such that \(a^r_s = v^r_{s,i} \cdot d^r_{s,i}\), where \(v^r_{s,i} = \min_{x \in \{b^r_s, c^r_s, p^r_s\}} \{a^i_{rs}/d_{s,x}\}\) is the dominant share UE \(i\) receives at slice \(s\) at RU \(r\) under \(a^r_s\). Furthermore, \(d^r_{s,i}\) is the normalized demand, i.e., \(d^r_{s,i} = (d_s^r, d_s^c, d_s^p, d_s^r)/\{a^i_{rs}/d_{s,x}\}\), and \(d_{s,r}^i\) is the demand of the dominant resource \(r^i_{s}\) of UE \(i\) in slice \(s\).

Intuitively, Lemma 2 indicates that a non-wasteful allocation assigns resources in proportion to the UEs demand. Accordingly, the slice hypervisors at each RU solve the following problem in each TTI \(t\):

\[
\begin{align*}
\max_{\{v^r_{s,i}\}} & \quad \min_{i \in I_s} v^r_{s,i} \\
\text{s. t.} & \quad \sum_{i \in I_s} v^r_{s,i} d^r_{s,i} \leq \alpha^r_s, \quad \forall v^r_{s,i} \in \{b^r_s, c^r_s, d^r_s\}, r \in R, s \in S \\
& \quad \sum_{i \in I_s} v^r_{s,i} d^r_{s,i} \leq \varphi^r_s, \quad \forall r \in R, s \in S \\
& \quad \sum_{i \in I_s} v^r_{s,i} d^r_{s,i} \leq \varphi^r_s, \quad \forall r \in R, s \in S \\
& \quad \sum_{i \in I_s} v^r_{s,i} d^r_{s,i} \leq v^r_{s,i} d_{s,i,x}, \quad \forall x \in \{b^r_s, c^r_s, p^r_s\}, l, j \in \mathbb{Z}^+ \\
& \quad \chi^r_{s,i} \leq v^r_{s,i} \leq 1, \quad \forall x \in \{b^r_s, c^r_s, p^r_s\}, r \in R, s \in S 
\end{align*}
\]

where the constraints in Eqs. (14a)–(14c) state that resources are allocated in proportion to the UE’s demands and should not exceed the maximum capacity allocated to each slice by the broker, which leads to non-wasteful allocation. Recall that slices either provide UL or DL services, thus the bandwidth \(b^r_s\) corresponds to upload or download, depending on the service class. The constraint in Eq. (14d) signifies the fairness in the equalized dominant share. Furthermore, the constraint in Eq. (14e) guarantees the minimum resources for slice admission with a given QoS (i.e., according to an SLA). The problem in Eq. (14) does not have a solution when the constraint in Eq. (14e) is not satisfied, namely, when there are not enough resources to satisfy the minimum requirements of UEs/SPs. Hence, in addition to progressive filling [53], an admission queue is maintained: the maximum number of UEs \(|I_s|\) from Eq. (3) is considered for admission to slice \(s\) at RU \(r\) on a first-in-first-out basis and obtains the resources according to Eq. (14). If the utility of the SPs for the considered UL/DL tasks in the slice admission list are less than zero (no profit), then the tasks are served through the backhaul or by other means at the given TTI. These tasks are also removed from the potential slice admission list and replaced with waiting tasks from the queue. Utilities are calculated for all users in the \(s\) considering all possible matchings with the current resource assignment to tasks in potential admission list. The tasks in \(s\) with higher utilities are swapped with tasks in potential list with lower utilities (including resource allocation). The users in the potential list are admitted to the slices. The remaining services in the queue at the given TTI are served through the backhaul or by other means. This ensures that the slices \(S\) only admits the UEs which can satisfy the minimum required resources. The resource allocation for UE \(i\) at RU \(r\) is \(a^i_{rs} = (\alpha^r_s, \varphi^r_s, \varphi^r_s) = v^r_{s,i} \cdot d^r_{s,i}\), \(\forall a^i_{rs} \in \{\alpha^r_s, \alpha^r_s, \varphi^r_s, \varphi^r_s\}\). Once \(a^i_{rs}\) is determined (if \(|I_s| < |I_s|\)), for each slice \(s\), then each UE \(i\) is provided additional fractional \((\alpha^r_s - \alpha^r_s)/|I_s|\) bandwidth, \((\varphi^r_s - \varphi^r_s)/|I_s|\) caching, and \((\varphi^r_s - \varphi^r_s)/|I_s|\) processing resources. This is because the initial resource allocation by solving Eqs. (13) and (14) is non-wasteful, hence, the UEs at most receive only the demanded resources for a service. The excess resources are fairly allocated to the UEs of that slice to improve the QoS.

**C. Analysis of 2L-MRA**

This section proves that 2L-MRA satisfies the efficiency, fairness, and economic properties outlined in Section VI-A.

**Theorem 1.** 2L-MRA has polynomial time complexity (PTC).
Algorithm 1: 2L-MRA Mechanism

Input: A set of UEs $i, j \in I$, SPS $s \in S$, slices $s \in S$
Output: Fair allocation of resources to UEs through slices $\forall b \in B, s \in S$

1. Initialize $A^{ei} \in \mathbb{R}^{(|I| \times |S|)}$ and $A^{ri} \in \mathbb{R}^{(|I| \times |S|)}$ with zeros
2. foreach time window $w$ do
   // Phase 1: CU-level Scheduling
   3. Estimate $d^{ei}_w$ of all $s$ at each RU $r$ \( \triangleright \) Eq. (12)
   4. Identify $\tau^{ei}_w$ of slice $s$
   5. Compute $o^{ei}_w$ for all $s$ at each RU $r$ \( \triangleright \) Eq. (13)–(13b)
   6. $A^{ei} \leftarrow o^{ei}_w \cdot d^{ei}_w$
   7. Send $A^{ei}$ to each RU $r$
   // Phase 2: RU-level Scheduling
   8. foreach TTI $t \in W$ and time slots $c \in \{e, e_u, e_m\}$ do
      9. $\psi^{ei}_i \leftarrow \emptyset$
      10. $\psi^{ei}_i \leftarrow$ Select first $|I|_m$ users from $s$ for probable slice admission
      11. Compute $\psi^{ei}_i, \delta^{ei}_j$ for all UE $i$, in $\psi^{ei}_i$ for all slices $x$ at each RU $r$, $\forall x \in \{s, s_u, s_m\}$ \( \triangleright \) Eq. (14)–(14e)
      12. $A^{ei} \leftarrow \psi^{ei}_i \cdot d^{ei}_x, \forall x \in \{s, s_u, s_m\}$
      13. foreach user $i \in \psi^{ei}_i$ do
         14. if $U^{ei}(t) \leq 0$ or $U^{ei}(t) \leq 0$
            15. Serve the UE $i$ without slicing
            16. $\psi^{ei}_i \leftarrow \psi^{ei}_i \setminus \{i\}$
            17. Calculate $U^{ei}(t)$ or $I^{ei}(t)$ for all users in $s$, $\forall A^{ri}$
            18. Swap users in $\psi^{ei}_i$ with users in $\delta^{ei}_j$ if $U^{ei}(t)$ or $U^{ei}(t)$ is higher for any user.
            19. Serve all users in $\psi^{ei}_i$ through slice $s$
            20. Serve the remaining users in $\delta^{ei}_j$ without slicing when $|I|_s + |I|_m > |I|_s$, $\forall r \in R$
            21. if $|I|_s < |I|_s$
               22. $y_{i,s} + y_{i,s} + (y_{i,s} - y_{i,s})/|I|_s$, $\forall y \in \{x, y, z\}$
               23. $\alpha^{ei}_{i,s} \leftarrow a^{ei}_{i,s}$

Proof. According to Algorithm 1, the broker allocates the resources of $|R|$ RU to $|S|$ slices at each time window, which takes $O(|R| \cdot |S|)$ time. Within each TTI, the slice hypervisors at each RU independently allocate their resources in $|I|_s$ UEs in $O(|I|_s)$ time (following progressive filling after UE removal from slice queues). The swapping process has a complexity of $|I|_s \cdot |I|_s$. Consequentially, the total time complexity of 2L-MRA is $O(|R| \cdot |S| + |I|_s + |I|_s |I|_s) \approx O(|I|_s^2)$. 

Theorem 2. 2L-MRA is envy-free (EF).

Proof. It is sufficient to prove that the dominant shares are equal for any two UEs, namely, that $\tau^{ei}_w \leq \tau^{ei}_j$, $\forall i, j \in I$. Solving Eq. (14) gives $\{v^{ei}_x\}$, then $\{v^{ei}_x\} = \min_x \{v^{ei}_x\}$. \( \{d^{ei}_x\} = \min_x \{v^{ei}_x\} \cdot d^{ei}_x \}

The final equality holds given that $\{v^{ei}_x\} \leq d^{ei}_x \leq d^{ei}_x \leq d^{ei}_x / d^{ei}_x \leq d^{ei}_x / d^{ei}_x \leq d^{ei}_x / d^{ei}_x \leq d^{ei}_x / d^{ei}_x \leq 1$. This implies that the dominant resource allocation is always the same for any two UEs at any time. Further, the allocation of excess resources to the UEs is done equally. Hence, the allocation is still EF for any two UEs $i, j$ subscribing to an SP $s$ at RU $r$. It can be similarly shown that any two SPs in a given RU have the same dominant shares for their respective slices. Thus, SPs are EF as well.

Theorem 3. 2L-MRA is Pareto-optimal (PO).

Proof. Let $\{o^{ei}_w\}$ be the dominant shares found as the solution to Eq. (13), and $A^{ei}_s = o^{ei}_w \cdot d^{ei}_w$ the allocation at RU $r$ for slice $s$ utilized by SP $s$. In addition, assume that $A^{ei}_s$ is not PO, i.e., there exists another allocation $\hat{A}^{ei}_s$ such that a slice $s$ can increase its dominant share $\hat{o}^{ei}_w$ without decreasing the $o^{ei}_w$ of other slices. This implies that the dominant share $\hat{o}^{ei}_w$ of $s$ in $\hat{A}^{ei}_s$ is greater than all $o^{ei}_w$ of all slices in $A^{ei}_s$. However, $\hat{A}^{ei}_s = o^{ei}_w \cdot d^{ei}_w$ holds due to Lemma 1. Moreover, excess resources are equally distributed among the UEs, resulting in saturation. Decreasing the allocation of any resource in slice $s$ also decreases its $\hat{o}^{ei}_w$ since the resource allocation is proportional to the demand vector. This contradicts the assumption that $\hat{A}^{ei}_s$ is not PO; therefore, $A^{ei}_s$ is PO. A similar proof can be constructed for the Pareto-optimality of resource allocation to UEs.

Theorem 4. 2L-MRA satisfies the slice sharing incentive (SSI) property.

Proof. Consider $S$ slices that provide services requiring different dominant resources. Let resource $x$ be the first one to saturate by using the progressive filling at any RU $r$. Assume that slice $s$ is allocated the majority $(\hat{f}^s_x)$ of resource $x$. This implies $\hat{f}^s_x \geq (1/n)$. Moreover, it is $\hat{f}^s_x \geq f^s_x \geq (1/n)$ by the definition of the current resource. Each slice obtains at least $1/n$ of its resource dominant, as progressive filling in the dominant allocation at the same rate and further allocation of excess resources is done equally among the slices. Consequently, 2L-MRA satisfies the SSI property.

VII. PERFORMANCE EVALUATION

This section evaluates the performance of 2L-MRA by extensive simulations based on real datasets.

A. Simulation Setup and Methodology

Experiments are carried out with a custom network simulator built on top of the SliceSIM software [56], a python-based simulation framework for network slicing in 5G. The scenario illustrated in Fig. 1 is considered, with $|S|=6$ SPs, equally divided between eMBB (DL), URLLC (UL), and mMTC (UL). Simulations last for $T=1 \times 10^7$TTIs, where each $\Delta t=1$ms. The additional parameters employed in the simulations are reported in Table IV.

Two datasets are used to characterize the service utilization of the UEs of the DL SPs: the Live Streaming Sessions Dataset [57], with streaming session data from YouTube Live and Twitch.tv; and the Music Streaming Sessions Dataset [58], with 130 million listening sessions (as well as the associated user interactions) on the Spotify music service. The services provided by the DL SPs are randomly selected from the datasets according to a uniform distribution. Furthermore, the abandon probability is calculated for each UE subscribed to the service from the associated data. The UL tasks are assumed to be independent from each other; they arrive at users according to a Poisson distribution at the rate of 10, 20, and 30 tasks/s for the eMBB, URLLC, and mMTC services, respectively. The minimum resource requirements for the eMBB, URLLC, and mMTC slices (for both UL and DL) are randomly selected.
from a subset of 20 real-world Amazon instances [59] suitable for the three slice categories (e.g., c4.8xlarge is compute-optimized, r4.16xlarge is memory-optimized, 13.8xlarge is storage-optimized). This choice is also similar to the resource requirements of slicing services in 5G networks [60]. The resource capacity of each F-AP is randomly selected from a set of M4 and M5 Amazon EC2 instances (e.g., m4.4xlarge has 16 vCPUs, 64 GB of memory, and 500 Mbps of bandwidth).

The following schemes in the state of the art are used for comparison purposes.

- **Static slicing with equal allocation (SSE) [28]:** resources are allocated to the slices once and remain the same over time. All the slices are provided with the same resources and UEs are allocated to the slices with the minimum resource guarantees.

- **Static slicing with proportional allocation (SSP) [64]:** resources are allocated proportionally to the slices based on their demands (i.e., the number of UEs) and also remain the same over time. UEs are served through the slices with minimum resource guarantees.

- **Dynamic slicing with dynamic hierarchical resource allocation (DSDHR) [45]:** resources are dynamically allocated to slices along with user admission at each window \( w \) and UEs are dynamically allocated resources every TTI (i.e., there is one slot per TTI) for the considered slice classes.

- **Dynamic slicing with proportional allocation (DSP):** a variant of 2L-MRA in which resources are dynamically scheduled and allocated to the slices based on their demands (i.e., the number of UEs) at each window \( w \). The resources for UEs are scheduled once every TTI (i.e., there is one slot per TTI) for the considered slice classes.

All the schemes above guarantee the minimum resource requirements to UEs through the slices, similar to 2L-MRA.

Simulations followed the independent replication method with twenty iterations for each experiment. The results report the average values obtained, along with error bars representing the corresponding standard deviations where noticeable.

**B. Results and Discussion**

The performance of 2L-MRA is evaluated next in terms of the obtained utility and other performance metrics. In particular, the utility is first assessed over time; then the impact of multiple parameters on it is characterized: the number of SPs and RUs (i.e., topology), the task arrival rate (i.e., the traffic), and the availability of different types of resources (i.e., computing and bandwidth). Finally, the optimality of 2L-MRA and the fairness of the considered schemes are analyzed, together with the related resource utilization.

**Utility over time and impact of SPs/RUs.** Fig. 3a shows the impact of slice scheduling on the total utility over time, expressed as windows during a representative simulation run, for the different schemes. Both SSE and SSP have a slower rate of utility change, except for the very beginning of the simulation time. In particular, the rate of utility change for SSE, SSP, DSDHR, DSP, and 2L-MRA is 4.17, 6.77, 10.9, 11.41, and 18.23 per window (respectively). Such a higher rate change for 2L-MRA, DSP, and DSDHR is due to the dynamic slice scheduling at each window. Moreover, 2L-MRA improves over DSP by a factor of 6.82 per window and over DSDHR by 7.33 per window as a result of scheduling multiple UEs within a single TTI for multiple services. DSP performs better than DSDHR as the latter is unable to efficiently assign resources to slices. Another important factor that improves the utility of 2L-MRA is the improved allocation of caching resources compared to the rest of the schemes.

Fig. 3b illustrates the total utility as a function of the number of SPs for the considered schemes. As the number of SPs increases, the utility also increases in all cases. The total utility with 2L-MRA is particularly high, due to the dynamic scaling of the slices and the more effective allocation of resources to UEs across multiple slots per TTI. Clearly, the two static scheduling schemes, SSE and SSP, obtain the lowest utilities. Instead, 2L-MRA provides the highest utility in all cases, with a utility gain of 63\%, 53\%, 42\%, and 33\% with respect to SSE, SSP, DSDHR, and DSP (respectively) when there are 16 SPs.

Fig. 3c depicts the total utility as a function of the number of RUs for the considered schemes. The total utility of the SPs always increases with the number of RUs. This is due to an increase in edge resources, allowing a higher number of UEs to be admitted then served through the slices. As in the previous cases, 2L-MRA obtains the highest utility and SSE the lowest utility. For instance, for 30 RUs the utility of 2L-MRA is $8.582$, which is 43\% higher than SSE, 28\% higher than DSDHR, and 20\% higher than DSP. DSP and DSDHR are much closer to 2L-MRA due to the use of dynamic scheduling. However, scheduling over multiple slots per TTI enables 2L-MRA to admit more UEs to the slices. DSP achieves better performance than DSDHR due to more efficient scheduling of multiple resources in each time slot.

**TABLE IV: Simulation parameters.**

<table>
<thead>
<tr>
<th>Parameter(s)</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density size</td>
<td>500 x 500 m</td>
</tr>
<tr>
<td>DL rate capacity of vCU: $B_{DL}$</td>
<td>1 Gbps [61]</td>
</tr>
<tr>
<td>UL rate capacity of vCU: $B_{UL}$</td>
<td>500 Mbps [61]</td>
</tr>
<tr>
<td>Computing capacity of vCU: $P_{vCU}$</td>
<td>9 x $10^{10}$ cycle/s [61]</td>
</tr>
<tr>
<td>Number of RUs: $</td>
<td>R</td>
</tr>
<tr>
<td>Transmission range of RU $r_{T}$</td>
<td>200 meters</td>
</tr>
<tr>
<td>DL/UL data-rates of RU $r_{T}$: $B_{DL}^{r_{T}}$, $B_{UL}^{r_{T}}$</td>
<td>501, 250 Mbps [60]</td>
</tr>
<tr>
<td>Storage capacity of RU $r_{T}$: $C_{RU}^{r_{T}}$</td>
<td>2 TB</td>
</tr>
<tr>
<td>Processing capacity of RU $r_{T}$: $P_{RU}^{r_{T}}$</td>
<td>4 x $10^{10}$ cycle/s [62]</td>
</tr>
<tr>
<td>Number of UEs: $</td>
<td>U</td>
</tr>
<tr>
<td>Transmission power of UEs: $p_{UE}^{r_{T}}$</td>
<td>10 dBm [63]</td>
</tr>
<tr>
<td>Rate parameter: $\alpha(w)$</td>
<td>$1/[63]$</td>
</tr>
<tr>
<td>Processing density: ($c$)</td>
<td>162.5 cycle/bit [20]</td>
</tr>
<tr>
<td>Window duration: $W$</td>
<td>40 s</td>
</tr>
<tr>
<td>TTI duration: $\Delta_{T}$</td>
<td>100 ms</td>
</tr>
<tr>
<td>Number of SPs: $</td>
<td>S</td>
</tr>
<tr>
<td>Skewness of the file popularity of SP $s \in S$: $\gamma$</td>
<td>0.8</td>
</tr>
<tr>
<td>Abandon probability threshold for UE $i \in Z_{s}$: $\phi$</td>
<td>0.7</td>
</tr>
<tr>
<td>Minimum number of UEs: $\xi_{u}, \xi_{c}, \xi_{m}$</td>
<td>10, 20, 300</td>
</tr>
<tr>
<td>Price in unit PCT: $\Phi_{u}, \Phi_{c}$</td>
<td>1, 0.8</td>
</tr>
<tr>
<td>Number of SPs in service classes: $</td>
<td>S_{u}</td>
</tr>
<tr>
<td>Number of UEs in service classes: $</td>
<td>U_{u}</td>
</tr>
<tr>
<td>Min. resources in URLLC slices: $r_{UL}^{m_{u}}, r_{UL}^{m_{c}}, r_{UL}^{m_{m}}$</td>
<td>15, 10, 4</td>
</tr>
<tr>
<td>Min. resources in eMBB slices: $r_{DL}^{m_{u}}, r_{DL}^{m_{c}}, r_{DL}^{m_{m}}$</td>
<td>30, 15, 35</td>
</tr>
<tr>
<td>Min. resources in mMTC slices: $r_{DL}^{m_{u}}, r_{DL}^{m_{c}}, r_{DL}^{m_{m}}$</td>
<td>5, 15, 25</td>
</tr>
<tr>
<td>Number of slots per TTI in slices: $e_{u}, e_{c}, e_{m}$</td>
<td>2, 4, 8</td>
</tr>
</tbody>
</table>
Impact of traffic and resource availability. Fig. 4a illustrates the total utility as a function of the task arrival rate ($\lambda$) for the considered schemes. Clearly, such utility increases as the number of tasks increases as well. Moreover, 2L-MRA obtains the highest utility, with an increase of 50.67% over the second-best scheme, DSP; both DSDHR and SSP perform very poorly. The rate of utility increase, instead, is inversely proportional to the task arrival rate. This is due to reduced availability of resources for newly arrived tasks, which are served through the backhaul. The average increase for each SP in 2L-MRA until $\lambda = 100$ is $11.44$ per task while the average increment per task for each SP is $1.4$ between $\lambda = 100$ to $\lambda = 1,000$. In contrast, DSP achieves an average utility of $7.53$ per task and $0.927$ per task for a given SP.

Fig. 4b depicts the utility as a function of computing capacity. The total utility increases when the computation capacity increases due to a large number of compute-intensive tasks, which is consistent with real-world scenarios. 2L-MRA performs the best over the entire range of computing capacity considered, i.e., from $10^9$ to $10^{10}$ cycles/s. Specifically, the total utility of 2L-MRA increases from $1,245$ to $4,270$ at an average rate of $325$. Among the other schemes, SSP performs the worst, followed by DSDHR, and DSP with an average gap of $643$, $985$, and $1,395$ respectively.

Fig. 4c shows the utility of the SPs as a function of bandwidth. The trend is similar to that in Fig. 4b; here 2L-MRA achieves the highest utility increase with an average rate of $364.895$ when the bandwidth increases from 200 Mbps to 300 Mbps. In this case, however, the difference between DSP and DSDHR is smaller, with a gap that is consistent across the considered values of bandwidth. This is mainly because of the higher demand for compute-intensive UL tasks compared to bandwidth-intensive DL tasks from UEs. Moreover, a larger amount of DL traffic is routed through the backhaul for DSP and DSDHR, as more bandwidth is allocated to UL tasks. Finally, SSP performs worse with varying bandwidth as opposed to computing capacity because the related allocation does not change within a time window.

Impact of service classes, optimality and fairness. Fig. 5 illustrates the utility of the SPs as a function of the number of RUs for the considered schemes, broken by service class (i.e., eMBB, mMTC, and URLLC). Clearly, the utility of SPs increases with the number of RUs, as in Fig. 3c. However, the figure shows that DSP and DSDHR obtain values that are not very dissimilar. Moreover, the utility clearly varies with the service class with 2L-MRA, while it is almost the same for the other schemes. As a consequence, each service class can handle more UEs and efficiently utilize resources with 2L-MRA. This result shows how 2L-MRA is effective in considering and allocating different service classes according to their requirements, due to scheduling UEs at a finer granularity.

Fig. 6 depicts the optimality gap of 2L-MRA for a small network topology with $|U| = 6$ and $|S| = 3$. In particular, the figure shows the total utility of both 2L-MRA and the optimal solution of Problem 1 obtained through the IBM
Utility of Individual SPs ($)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number of RUs</th>
<th>Total Utility ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2L-MRA</td>
<td>5</td>
<td>0.4</td>
</tr>
<tr>
<td>DSP</td>
<td>10</td>
<td>0.6</td>
</tr>
<tr>
<td>DSDHR</td>
<td>15</td>
<td>0.8</td>
</tr>
<tr>
<td>SSP</td>
<td>20</td>
<td>1.0</td>
</tr>
<tr>
<td>SSE</td>
<td>25</td>
<td>1.2</td>
</tr>
<tr>
<td>Optimal</td>
<td>30</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Jain’s Fairness Index

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Jain’s Fairness Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>2L-MRA</td>
<td>0.4</td>
</tr>
<tr>
<td>DSP</td>
<td>0.6</td>
</tr>
<tr>
<td>DSDHR</td>
<td>0.8</td>
</tr>
<tr>
<td>SSP</td>
<td>1.0</td>
</tr>
<tr>
<td>SSE</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Resource utilization. The rest of the discussion focuses on the efficiency in the use of resources, quantified in terms of: the utilization ratio, as the fraction of used resources with respect to those allocated; and resource wastage, as the amount of resources that remain unallocated due to dependencies in multi-resource allocation scenarios.

Fig. 6a illustrates the utilization ratio of the dominant resource (DR) as a function of time across all slices at RUs, for a representative simulation run. 2L-MRA achieves the best resource utilization with an average of 96% and a maximum of 99% across all slices and RUs under consideration. Instead, DSP and DSDHR achieve a lower average DR resource utilization of 91% and 88%, respectively. DSDHR peaks in resource utilization over several time windows compared to DSP, but cannot maintain the resource utilization at a consistent level. In contrast, 2L-MRA consistently entails a higher DR utilization ratio. The lowest average DR utilization is 73% obtained by SSE. This clearly illustrates that 2L-MRA has a better DR resource utilization across the slices and RUs.

Fig. 6b shows the average resource utilization ratio with respect to the individual resources across all the slices at RUs. 2L-MRA achieves the highest resource utilization with 96%, 98%, and 99% of bandwidth, storage, and vCPU usage – as opposed to the 90%, 86%, and 50% utilization of the same in the second-best scheme, DSP. It is clear that, except for 2L-MRA, a higher utilization of one or two resource types results in a poor utilization of the remaining resources. This correlates with the achieved utility, as previously discussed. Consequently, 2L-MRA is a much better option while considering multiple resources for network slicing.

Finally, Fig. 6c illustrates resource wastage as a function of the multiple resources used across all the slices at RUs. The resource wastage in 2L-MRA is similar for bandwidth, storage, and vCPU with extremely low values: 0.15%, 0.11%, and 0.018%, respectively. Whereas, DSP and DSDHR are
much closer to each other in terms of resource wastage, with the exception of storage. Wasted resources greatly vary for DSDHR, SSP, and SSE, with a standard deviation of 4.5%, 16%, and 18%, respectively. Different from these schemes, 2L-MRA incurs a negligible resource wastage for all resource types across slices.

VIII. CONCLUSION

This article introduced 2L-MRA, a utility-based mechanism for joint allocation of heterogeneous resources in Fog-RAN slicing. 2L-MRA considers both UL and DL SPs as well as different classes of services to maximize utility over time. It has a polynomial time complexity and achieves the economic properties of Pareto optimality, envy-freeness, and sharing incentive. Extensive simulations in realistic scenarios have demonstrated that 2L-MRA significantly increases utility in varying network conditions due to dynamic resource allocation to slices and fine-grained resource scheduling for UEs, resulting in higher utilization and better QoS than the state of the art. In particular, 2L-MRA obtains 32% to 60% higher utility in higher utilization and better QoS than the state of the art. A future work could consider how the pricing of different resource types affects the economics of multi-resource allocation with multiple SPs. Another promising direction is given by the design of techniques based on machine learning to allocate heterogeneous resources according to different application requirements.

REFERENCES


