Self-Learning Multi-Mode Slicing Mechanism for Dynamic Network Architectures

Haitham H. Esmat, *Member, IEEE*, and Beatriz Lorenzo, *Senior Member, IEEE*

***Abstract*— Dynamic network architectures that utilize communication, computing, and storage resources at the wireless edge are key to delivering emerging services in next-generation networks (e.g., AR/VR, 3D video, intelligent cars, etc). Network slicing can be significantly enhanced by including dynamically available resources throughout the fog/edge/cloud continuum and using mmWave/THz bands. However, network slicing of dynamic multi-tier computing networks remains under-explored. In this paper, we present a self-learning end-to-end network slicing mechanism (SELF-E2E-NS) that facilitates collaboration between the Infrastructure Provider (InP) and tenants to slice their subscribers' resources (i.e., radio, computing, and storage) as fog resources. To adapt to the uncertain availability of resources at the edge and minimize the risk of non-satisfying service level agreements (SLAs), our slicing mechanism has two operational modes. Operational mode 1 is for joint network slicing (JNS) in which the InP infrastructure is augmented with fog resources and jointly sliced to meet high throughput and delay tolerant requirements. Operational mode 2 is for independent network slicing (INS) in which the InP infrastructure and fog resources are sliced separately to achieve high throughput, low-latency, and high-reliability requirements. Our schemes leverage mmWave/THz, fog/edge/cloud computing, and caching to achieve new service requirements. We design a DQ-E2E-JNS algorithm that uses Deep Dueling network and a MAAC-E2E-INS algorithm based on multi-agent actor-critic, which incorporate service-aware pricing feedback and fog trading matching, respectively. These algorithms find the optimal slice request admission and collaboration policy that maximizes the long-term revenue of the InP and tenants for each mode. The simulation results show that our novel slicing mechanism can serve up to 4 times more requests and effectively exploits different spectrum bands and fog resources to improve revenue and performance.**

***Index Terms*— Dynamic network architectures, network slicing, multi-agent actor-critic, fog/edge/cloud, risk model.**

I. Introduction

THE sixth-generation (6G) of wireless networks is expected to rely on dynamic multi-tier network architectures that utilize resources at the wireless edge to accommodate traffic demands from emerging services such as virtual/augmented reality (VR/AR), smart driving and eHealth. The incorporation of millimeter wave (mmWave) communication has significantly improved the offloading capability at the edge enabling low latency and high throughput [1], [2], [3], [4]. On the other hand, network slicing (NS) [5], [6] creates logical networks customized to meet the needs of different applications and enables multi-tenancy, reducing operational and capital expenditures. However, slicing the available communication and computing resources throughout the fog/edge/cloud continuum brings additional challenges that remain underexplored [7]. First, cooperation between the InP and tenants is needed to cope with the varying resource availability and service requirements. Second, validating the service level agreement (SLA) and mitigating any service degradation is more challenging since slices include different types of resources (e.g., mmWave/THz bands, computing resources, caching) controlled by different entities. Third, with the increasing service demand and unleashed resources at the wireless edge, the number of slices running simultaneously in the network is expected to increase, requiring agile NS mechanisms. To the best of our knowledge, our NS mechanism is the first to address all of these challenges.

Recently, some works study mmWave/THz applied to traffic offloading and network slicing [8], [9], [10]. In [8], a THz wireless multi-access edge computing system for high-quality immersive VR video services is presented to minimize long-term energy consumption by jointly optimizing the downlink transmission power and viewport rendering offloading. In [9], the mmWave communications and non-orthogonal multiple access NOMA are exploited for mobile edge computing networks to enhance the performance of task offloading. In [10], a three-phase framework to price network infrastructure slices exploiting mmWave bands is proposed. Other works have applied NS to edge/fog networks for computational offloading. Xiao and Krunz [11] present an NS scheme for fog computing networks to offload computational tasks using energy harvested from the environment. Chen et al. [12] study computation offloading policies for a mobile device based on task queue state, energy queue state, and channel qualities to choose between offloading to an edge or cloud server. Sun et al. [13] suggest a hierarchical radio resource allocation for NS in fog radio access networks to share the spectrum between users and fog nodes. These works slice only one type of resource, which results in sub-optimal performance when applied to real scenarios [14].

Network slicing under traffic uncertainty has been studied in [15], and [16]. Feng et al. [15] examine the tradeoff between revenue and delay in slice admissions under traffic variations using Lyapunov optimization for a single operator and traffic class. Chien et al. [16] propose an algorithm to prevent the over-provisioning of computing and network resources based on measured statistics of typical real-world 5G services. Dynamic network slice reconfiguration and recovery schemes are proposed in [17]. However, their application to slice the wireless edge/fog will result in frequent slice reconfiguration which brings management and control overhead.

Manuscript received 25 August 2022; revised 27 April 2023; accepted 13 August 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Joshi. Date of publication 24 August 2023; date of current version 18 April 2024. This work was supported in part by the U.S. National Science Foundation under Grant CNS-2008309. *(Corresponding author: Beatriz Lorenzo.)*

The authors are with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01002 USA (e-mail: habdelhafez@umass.edu; blorenzo@umass.edu).

Digital Object Identifier 10.1109/TNET.2023.3305975

As service demands become increasingly diverse and the number of slices grows, automated solutions for dynamic slice admission and collaboration are needed. In [18], Bega et al. exploit reinforcement learning (RL) schemes to cope with dynamic slice requests by making decisions in a trial-and-error learning approach. In [14], the authors present Deep RL (DRL)-based algorithms to maximize the revenue of the InP in slicing its fixed cellular infrastructure. Our preliminary work [19] shows that augmenting the InP infrastructure with fog user equipment (FUEs) resources can significantly increase the revenue of the InP and tenants compared to existing approaches.

In this paper, we contribute to the design of efficient network slicing schemes for dynamic multi-tier network architectures:

- We develop a novel network slicing mechanism that facilitates collaboration between the Infrastructure Provider (InP) and tenants (i.e., operators) to share their resources (InP infrastructure and FUEs) and serve tenants' requests from different traffic classes. To exploit the dynamically available FUEs' resources efficiently, we proposed two schemes: 1) An E2E joint network slicing (E2E-JNS) scheme in which the InP augments its infrastructure with temporarily available FUEs resources and jointly slices the augmented infrastructure to meet high throughput and delay-tolerant traffic requirements, and 2) an E2E independent network slicing (E2E-INS) scheme in which the InP slices its infrastructure and tenants collaborate to create fog slices through sequential and fine control of their FUEs' available resources. By augmenting the InP slices with fog slices, this scheme achieves low latency and high-reliability requirements.
- To minimize the risk of non-satisfying the service request in this dynamic multi-tier network, we design a self-learning network slicing management (SELF-E2E-NS) scheme that adapts its operation to the network condition. It activates E2E-JNS scheme (mode 1) or E2E-INS scheme (mode 2) previously described. Upon receiving a request for a service class from each tenant, the orchestrator activates the mode with the lowest risk of non-satisfying the request based on the previous feedback from InP (i.e., acceptance or rejection) and tenants (i.e., probability of satisfying the SLA). In this way, our self-learning mechanism can deal with uncertain traffic demands and network dynamics.
- To find the optimal slice request admission and collaboration policy for each scheme, we present a DQ-E2E-JNS algorithm with service-aware pricing feedback based on Deep Dueling to find the InP's optimal policy under uncertainty of future slice requests and fog availability. Then, we design a distributed MAAC-E2E-INS algorithm based on a multi-agent actor-critic that incorporates a fog trading matching mechanism to find tenants' optimal association policy to buy/sell FUEs and create fog slices.
- Extensive simulations are performed to illustrate the efficiency of our NS scheme compared to other conventional schemes in the context of anticipated 6G applications. In fact, our SELF-E2E-NS scheme achieves 4.5 times higher reward and serves twice more requests than other existing approaches.

The rest of the paper is organized as follows. The system model is introduced in Section II. Section III describes the E2E-JNS scheme and the DQ-E2E-JNS algorithm. The E2E-INS scheme is described in Section IV together with MAAC-E2E-INS algorithm. Section V presents the self-learning network slicing mechanism. Simulation results are examined in Section VI, and Section VII concludes the paper.

II. System Model

A. Network Slicing Model

We consider a network slicing model, as illustrated in Fig. 1, in which each tenant $x \in \mathcal{X}$ requests slices from the InP to serve their subscribers' demands of class $c \in \mathcal{C}$. We assume that tenants serve the following anticipated 6G traffic classes [20]: Mobile Broadband Machine Type Communication (MBBMTC), i.e., class $c = 1$ services, supporting high broadband data rates along with massive connectivity; Mobile Broadband Reliable Low Latency Communication (MBBRLLC), i.e., class $c = 2$ services, offering high broadband data rates along with reliable, and low latency communication; Reliable Low Latency Machine Type Communication (RLLMTC), i.e., class $c = 3$ services, supporting massive connectivity, reliability, and low latency; and Mobile Broadband and Reliable Low Latency Machine Type Communication (MBBRLLMTC), i.e., class $c = 4$ services, supporting high data rates, reliability, low latency, and massive connectivity.

The InP owns the physical network infrastructure that supports communication services and edge/cloud computing. We assume that the InP rents its resources to tenants by jointly slicing its communication B_I, computing G_I, and storage V_I resources to serve different service classes. Each slice request of class c is associated with a vector of resources of each type $\boldsymbol{\theta}^c \triangleq [b^c, g^c, v^c]$, $\forall c \in \mathcal{C}$, where b^c, g^c, and v^c are the units of communication, computing, and storage resources in a slice of class c, with $\sum_c b^c \leq B_I$, $\sum_c g^c \leq G_I$, and $\sum_c v^c \leq V_I$. Besides, each slice of class c has a service requirement in terms of throughput T^c_{min}, delay τ^c_{max}, and reliability ρ^c_{min} [1], [2], [3], defined in the SLA by the tuple $\{\boldsymbol{\theta}^c, T^c_{min}, \tau^c_{max}, \rho^c_{min}\}$. The impact of these metrics on different applications is elaborated in Section VI. Tenants pay different prices depending on the traffic class. They can collaborate with the InP by sharing their subscribers' resources, whenever idle, as fog user equipment (FUE) resources and by selling unused slices back to the InP. The FUE resource vector $\boldsymbol{\theta}^c_{FUE} \triangleq [b^c_{FUE}, g^c_{FUE}, v^c_{FUE}]$ represents the resources from FUEs in a slice of class c.

To leverage the multi-access opportunities across the fog/edge/cloud continuum enabled by the collaboration between InP and tenants, we consider two slicing modes, as shown in Fig. 1. Mode $l = 1$, in which the InP augments its infrastructure with FUEs and jointly slices the augmented infrastructure to meet high throughput and delay-tolerant requests. The NS scheme activated in mode 1 is referred to as E2E joint network slicing (E2E-JNS), while the scheme activated in mode $l = 2$ is referred to as E2E independent NS (E2E-INS). This scheme slices InP resources and fog resources separately to track the availability of FUEs and meet high throughput, high-reliability, and low-latency requirements.

We assume that slice request arrivals of class c follow a Poisson distribution with rate λ_c and each request has a lifetime that follows an exponential distribution with the mean rate $1/\mu_c$. When a slice request arrives, the orchestrator has no information about the available resources in the network but has historical data about the responses of the InP to the

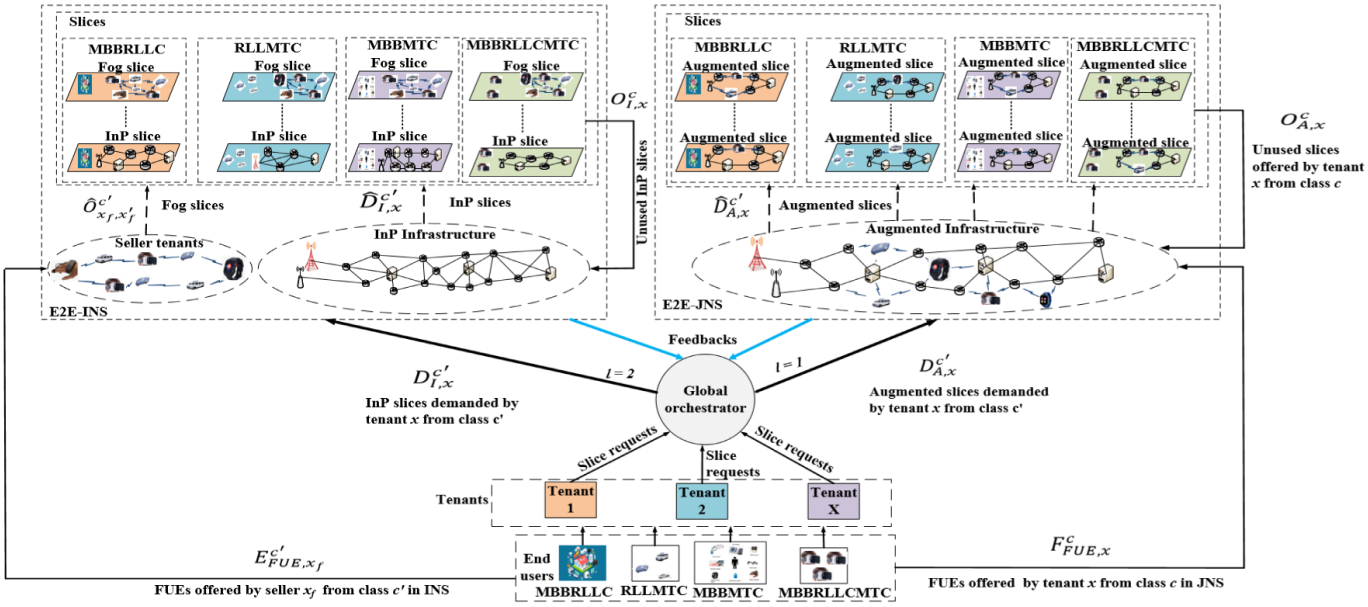


Fig. 1. Multi-mode network slicing model for dynamic network architectures.

previous slice requests (i.e., accept or reject) and the feedback from tenants (i.e., if accepted requests met the SLA). Based on the historical data, the orchestrator estimates the risk that the tenant x slice request from class c satisfies the service under scheme l when the network state is s,

$$risk^c_{x,l} = -\ln Pr[\xi^c_{x,l} = 1|s] \tag{1}$$

where $\xi^c_{x,l}$ is the acceptance probability of the slice request. The objective of the orchestrator is to select the slicing mode l to forward each request to the scheme with the minimum risk of non-satisfying the request. Our E2E slicing mechanism can develop self-learning capabilities by learning from previous experience and adjusting the slicing mode decision as the network condition and traffic evolve. More implementation details are provided in Section V.

B. Computing Model

We assume that no local computing is performed at end UEs and their traffic cannot be split, as in many practical scenarios [24]. Thus, each UE traffic is served by either an FUE or an InP node. In the independent slicing scheme, the computing decision is given by the slice type (i.e., InP slice or fog slice). Therefore, we focus our next discussion on the joint slicing scheme (i.e., augmented slices). The extension to obtain the computing delay in the former case is straightforward. We define a binary variable $\alpha^c_j \in [0,1], \forall j \in \mathcal{J}^c_A$, to indicate whether the traffic of UE j from class c given by K^c_j is computed at an FUE ($\alpha^c_j = 0$) or at an InP node ($\alpha^c_j = 1$) in the allocated slice. $\mathcal{J}^c_A$ denotes the set of UEs in an augmented slice of class c. We assume that InP nodes can compute tasks from several UEs simultaneously while FUEs can only compute one task at a time. The traffic of each UE j is processed in the accepted slice as follows:

- If the InP's decision is $\alpha^c_j = 0$, UE j directly offloads its computing task of K^c_j bits to FUE e that will allocate the required computing and storage resources to serve the task. If $\alpha^c_j = 1$, the InP serves the UE j by InP node i.
- The computation results are transmitted back to each UE. The delay in transferring the computation result back to the UE is omitted since the size of the computation outcome is much smaller than the size of the input data [25].
- We model the queuing delay at the InP nodes using an $M/M/1$ queuing model, in which each InP node i is treated as a server. The packet arrivals of class c follow a Poisson process with rate λ^c_i, and the service rate of class c follows an exponential distribution with the rate $1/\mu^c_i$. The time spent by each packet in the system is therefore given by $\frac{1}{\mu^c_i - \lambda^c_i}$.

Thus, the delay for UE j

$$\begin{aligned}(1-\alpha^c_j)\left[\frac{K^c_j}{T^c_{j,e}} + \frac{g^c_{req,j}K^c_j}{g^c_{j,e}} + \frac{\widehat{K}^c_{j,e}}{v^c_{j,e}}\right] \\ + \alpha^c_j\left[\frac{K^c_j}{T^c_{j,i}} + \frac{g^c_{req,j}K^c_j}{g^c_{j,i}} + \frac{\widehat{K}^c_{j,i}}{v^c_{j,i}} + \frac{1}{\mu^c_i - \lambda^c_i}\right] \\ \leq \tau^c_{max}, \quad \forall j \in \mathcal{J}^c_A, c \in \mathcal{C}, e \in \mathcal{E}, i \in \mathcal{I}\end{aligned} \tag{2}$$

where $\mathcal{E}$ is the set of FUEs and $\mathcal{I}$ is the set of InP nodes. If the request is served at FUE e, i.e., $(1-\alpha^c_j)$, the first term indicates the transmission delay where K^c_j is the data size and $T^c_{j,e}$ is the data rate of UE j from class c when transmitting to FUE e, which is calculated in Section VI, the second term is the computing delay at FUE e, where $g^c_{req,j}$ denotes the required computing resources (CPU cycles per bit) of UE j, $g^c_{j,e}$ is the computing resources (CPU cycles per second) allocated to UE j from FUE e. The third term is the caching delay where $\widehat{K}^c_{j,e}$ is the data size of the computation result of UE's computing task when served by FUE e and $v^c_{j,e}$ denotes the writing speed in storage (bits per second) at FUE e. Similarly, if the request is served at InP node i in a slice class c, i.e., α^c_j, the delay includes transmission, computing, caching, and queuing delay, where $T^c_{j,i}$ is the data rate of UE j from class c to InP node i, $g^c_{j,i}$ is the computing resources (CPU cycles per second) allocated to UE j from InP node i, $\widehat{K}^c_{j,i}$ is the data size of the computation result of UE j when served by InP node i, $v^c_{j,i}$ is the writing speed (bits per second) in storage at node i, and τ^c_{max} is the maximum acceptable delay of UE from class c.

TABLE I
NOTATION

Symbol	Meaning
$\mathcal{J}_A^c$, $\mathcal{J}_f^c$, $\mathcal{J}_I^c$	Set of UEs per augmented slice of class c, set of UEs per fog slice of class c, set of UEs per InP slice of class c
$\mathcal{C}_{D_A,x}$, $\mathcal{C}_{A,x}$, $\mathcal{C}$	Set of classes that demand augmented slices, set of classes with unused augmented slices, set of all classes
c, c'	Index of any traffic class with unused slices, index of any traffic class that requires slices
$\mathcal{X}_D^c$, $\mathcal{X}^c$, $\mathcal{X}$, X	Set of slice buyer tenants from class c, set of slice seller tenants from class c, set of all tenants, the total number of tenants
x, x'	Index of tenants with unused slices, index of tenants that require slices, $x \in \mathcal{X}^c$, $x' \in \mathcal{X}_D^c$
$D_{A,x}^{c'}, O_{A,x}^c, F_{FUE,x}^c$	No. of augmented slice demands from class c', augmented slice offers from class c, FUEs offers from class c by tenant x
$r_{D_A}^{c'}, r_{O_A}^{c'}, r_{FUE,x}^c, r_f^{c'}$	Price per augmented slice demanded of class c', augmented slice offered of class c', FUE of class c, fog slice of class c'
$\widehat{D}_{A,x'}^{c'}, \widehat{O}_{A,x}^c, \widehat{F}_{FUE,x}^c$	No. of augmented slice requests by tenant x' class c', augmented slice offers, FUEs offers from x and class c
$D_{I,x'}^c$,$O_{I,x}^c$, $\widehat{D}_{I,x'}^c$,$\widehat{O}_{I,x}^c$	No. of slice demands from x', slice offers from x, demands accepted from x', offers accepted from x by InP
$x_{f'}, x_f, \mathcal{X}_{D_f}, \mathcal{X}_f$	Index of fog buyer tenant and fog seller tenant, and set of fog buyer tenants, fog seller tenants
$D_{x'_f,x_f}^{c'}$, $\widehat{O}_{x_f,x'_f}^{c'}$	No. of fog slices buyer $x_{f'}$ requests from seller x_f and seller x_f offers to buyer $x_{f'}$
$\boldsymbol{\theta}_I = (B_I, G_I, V_I)$	Vector of radio, computing, and storage resources of the InP
$\boldsymbol{\theta}^c = (b^c, g^c, v^c)$	Vector of radio, computing, and storage resources per slice from class c
$\boldsymbol{\theta}_A^c$, $\boldsymbol{\theta}_{FUE}^c$, $\boldsymbol{\theta}_f^c$	Vector of resources per augmented slice, vector of FUE resources per slice, and vector of resources per fog slice of class c
T_j^c, τ_j^c, U_j^c, $p_{out,j}^c$	Throughput, delay, utility, and outage probability of UE j from class c

Moreover, the computing and storage resources allocated to all end UEs from InP node i should not exceed its capacity ($\sum_{j\in\mathcal{J}_A^c} g_{j,i}^c \le g_i^c$ and $\sum_{j\in\mathcal{J}_A^c} v_{j,i}^c \le v_i^c$). Similar constraints can be derived for serving the user at FUE e ($g_{j,e}^c \le g_{FUE,e}^c$ and $v_{j,e}^c \le v_{FUE,e}^c$), where g_i^c and $g_{FUE,e}^c$ are the available computing resources at nodes i and e, respectively, and v_i^c and $v_{FUE,e}^c$ are the storage resources at nodes i and e, respectively. The selection of the computing nodes for traffic offloading should consider that fog nodes offer lower computing delay but at a higher computing-unit cost than the InP resources. We summarize the most important notations in Table I.

III. End-to-End Joint Network Slicing (E2E-JNS)

We develop an E2E-JNS scheme in which the InP collaborates with tenants to augment its infrastructure with available FUEs' resources. We assume tenants incentivize their subscribers to share their resources whenever idle as FUEs [26]. Based on the tenants' demands and network condition, we derive the optimal long-term InP's admission and collaboration policy using DQ-E2E-JNS algorithm. Implementation details are given in Section V.

A. Tenants' Slice Demand/Offer Optimization

We assume that the arrival rate of tenant x's request from class $c \in \mathcal{C}$ is $\lambda_{x,l}^c = \lambda_x^c \xi_{x,l}^c$, where λ_x^c is the subscribers' traffic arrival rate from class c and $\xi_{x,l}^c$ is the probability that the request is forwarded to this scheme (i.e., slicing mode $l = 1$ is activated). Tenants request slices for a duration Δ. However, there will be times when slices are not fully utilized. These available resources can be temporarily sold back to the InP to serve requests from other tenants that exceed current resources −available in stock−of the InP. We denote by $\mathcal{C}_{A,x}$ the set of classes of augmented slices from tenant x currently running in the network and by $\mathcal{C}_{D_A,x} = \{\mathcal{C} \setminus \mathcal{C}_{A,x}\}$ the set of classes of augmented slices requested/demanded by tenant x, $\mathcal{C}_{D_A,x} \cap \mathcal{C}_{A,x} = \emptyset$, and $\mathcal{C}_{D_A,x}, \mathcal{C}_{A,x} \in \mathcal{C}$. Each tenant solves an optimization problem to determine the number of slices $D_{A,x}^{c'}$ needed to meet the demand for each class $c' \in \mathcal{C}_{D_A,x}$ and the number of unused slices and FUEs' offered back to the InP denoted by $O_{A,x}^c$ and $F_{FUE,x}^c$, $c \in \mathcal{C}_{A,x}$, respectively. The InP uses the latter resources to create new slices and serve requests from other tenants. The price for a slice from class c', an offered slice from class c, and a FUE from class c is $r_{D_A}^{c'}$, $r_{O_A}^c$, and r_{FUE}^c, respectively. Therefore, the optimization problem of tenant x is

$$\mathcal{P}_x(D_{A,x}^{c'}, O_{A,x}^c, F_{FUE}^c):$$

$$\underset{D_{A,x}^{c'}, O_{A,x}^c, F_{FUE}^c}{\text{maximize}} \quad \sum_{c\in\mathcal{C}_{A,x}} \left[\sum_{j\in\mathcal{J}_A^c} \Gamma_c U_j^c + O_{A,x}^c r_{O_A}^c + F_{FUE,x}^c r_{FUE,x}^c \right] - \sum_{c'\in\mathcal{C}_{D_A,x}, c'\neq c} D_{A,x}^{c'} r_{D_A}^{c'}$$

subject to

$$\begin{aligned}
&a) \quad T_j^{c'}(\boldsymbol{\theta}_j^{c'}) \ge T_{min}^{c'},\ \forall j \in \mathcal{J}_A^{c'},\ c' \in \mathcal{C}_{D_A,x} \\
&b) \quad \tau_j^{c'}(\boldsymbol{\theta}_j^{c'}) \le \tau_{max}^{c'},\ \forall j \in \mathcal{J}_A^{c'},\ c' \in \mathcal{C}_{D_A,x} \\
&c) \quad D_{A,x}^{c'} \boldsymbol{\theta}^{c'} \preccurlyeq \sum_{j\in\mathcal{J}_A^{c'}} \boldsymbol{\theta}_j^{c'},\ \forall c' \in \mathcal{C}_{D_A,x} \\
&d) \quad \sum_{j\in\mathcal{J}_A^{c'}} \boldsymbol{\theta}_j^{c'} \preccurlyeq \boldsymbol{\theta}^{c'},\ \forall c' \in \mathcal{C}_{D_A,x} \\
&e) \quad O_{A,x}^c \le N_{A,x}^c,\ \forall c \in \mathcal{C}_{A,x} \\
&f) \quad F_{FUE,x}^c \le E_{FUE,x}^c,\ \forall c \in \mathcal{C}_{A,x}
\end{aligned} \tag{3}$$

where the tenant aims to maximize the utility of its subscribers U_j^c, the number of FUEs $F_{FUE,x}^c$, and unused slices offered to the InP $O_{A,x}^c$ for each class $c \in \mathcal{C}_{A,x}$ while minimizing the number of slices requested $D_{A,x}^{c'}$, $c' \in \mathcal{C}_{D_A,x}$. The utility function of a UE j, $\forall j \in \mathcal{J}_A^c$ is $U_j^c = \left(\frac{T_j^c}{T_{min}^c}\right)^{\beta^c} / \left(\frac{\tau_j^c}{\tau_{max}^c}\right)^{\eta^c}$, where T_j^c is the achievable rate for UE j, and T_{min}^c is the minimum throughput requirement for UEs of class c. Similarly, τ_j^c is the delay for UE j and τ_{max}^c is the maximum tolerated delay and Γ_c is a scaling factor. $\beta^c, \eta^c \in [0,1]$ are mapping parameters that indicate the importance of throughput and delay for each class. Constraints (3.a) and (3.b) ensure that each UE j in a slice of class $c' \in \mathcal{C}_{D_A,x}$ achieves the service requirements previously described and $\boldsymbol{\theta}_j^{c'} = (b_j^{c'}, g_j^{c'}, v_j^{c'})$ is the vector of resources allocated to UE j. Constraint (3.c) ensures that the overall resources of each type for all slices of

class $c' \in \mathcal{C}_{D_A,x}$ from tenant x does not exceed the required resources by all UEs in the set $\mathcal{J}_A^{c'}$. Constraint (3.d) ensures that the total amount of resources allocated to all UEs in a slice of class $c' \in \mathcal{C}_{D_A,x}$ does not exceed the maximum amount of resources in a slice of the same class. Constraint (3.e) ensures that the unused slices $O_{A,x}^c$ of class $c \in \mathcal{C}_{A,x}$ in tenant x sold back to the InP do not exceed the available unused slices $N_{A,x}^c$ of the same class in tenant x. Finally, constraint (3.f) ensures that the FUEs of class c in tenant x sold to InP $F_{FUE,x}^c$ do not exceed its available FUEs $E_{FUE,x}^c$.

To solve problem (3), each tenant x calculates the utilities of their users based on the available resources at the InP. Then, they list the users in decreasing order of their utilities and group the ones who can achieve the QoS constraints for each class $c' \in \mathcal{C}_{D_A,x}$. The number of users in each group should not exceed the maximum number of users $\mathcal{J}_A^{c'}$ that can be served per slice of class c'. Next, each tenant x requests from the InP as many slices as groups of users has formed (i.e., $D_{A,x}^{c'}$ slices) per class $c' \in \mathcal{C}_{D_A,x}$, where each slice serves a group of users. The maximum number of slices needed to serve $J_x^{c'} = |\mathcal{J}_x^{c'}|$ subscribers is $D_{max,x}^{c'} = \left\lceil J_x^{c'} / |\mathcal{J}_A^{c'}| \right\rceil$. The tenant finds the optimum $D_{A,x}^{c'} \leq D_{max,x}^{c'}$ for each class c' that maximizes its overall reward. Moreover, each tenant x has $O_{A,x}^c$ unused slices of class $c \in \mathcal{C}_{A,x}$ and $F_{FUE,x}^c, c \in \mathcal{C}_{A,x}$ available FUEs. Since selling them increases the tenant's revenue, we assume the tenant sells all available resources to maximize its revenue. The complexity of this algorithm is provided in the Appendix.

B. InP's Optimal Admission and Collaboration Policy

We assume that there are two sets of tenants in the network. The set of tenants that are running slices of class c denoted by $\mathcal{X}^c$, and thus may offer them to the InP, i.e., seller tenant. The set of tenants that demand slices of class c given by $\mathcal{X}_D^c$, i.e., buyer tenant. The InP's optimal admission and collaboration policy is the one that maximizes the InP's revenue by accepting $\widehat{D}_{A,x'}^c$ slice requests from a buyer tenant $x' \in \mathcal{X}_D^c$, $\widehat{O}_{A,x}^c$ slice offers from a seller tenant $x \in \mathcal{X}^c$, and $\widehat{F}_{FUE,x}^c$ FUEs from tenant $x \in \mathcal{X}^c$ and class c as a response to the demands/offers received by each tenant as a solution to (3). The InP augments its infrastructure with $\widehat{F}_{FUE,x}^c$ FUEs and serves the slice requests by slicing its augmented infrastructure. The vector of resources per augmented slice of class c is $\boldsymbol{\theta}_A^c \triangleq [b_A^c, g_A^c, v_A^c]$ where $b_A^c = b^c + \sum_c \widehat{F}_{FUE,x}^c \, b_{FUE}^c$, $g_A^c = g^c + \sum_c \widehat{F}_{FUE,x}^c g_{FUE}^c$, $v_A^c = v^c + \sum_c \widehat{F}_{FUE,x}^c v_{FUE}^c$. The price for each augmented slice allocated to tenant $x' \in \mathcal{X}_D^c$, unused slice from tenant $x \in \mathcal{X}^c$, and FUE of class c is $r_{D_A}^c$, $r_{O_A}{}^c$, and $r_{FUE,x}^c$, respectively. Therefore, the InP's admission and collaboration optimization problem is

$$\mathcal{P}_I(\widehat{D}_{A,x'}^c, \widehat{O}_{A,x}^c, \widehat{F}_{FUE,x}^c):$$

$$\underset{\widehat{D}_{A,x'}^c, \widehat{O}_{A,x}^c, \widehat{F}_{FUE,x}^c}{\text{maximize}} \sum_{c \in \mathcal{C}} \left[\sum_{x' \in \mathcal{X}_D^c} \widehat{D}_{A,x'}^c r_{D_A}^c - \sum_{x \in \mathcal{X}^c} \widehat{O}_{A,x}^c r_{O_A}^c - \widehat{F}_{FUE,x}^c r_{FUE,x}^c \right]$$

subject to

$$a) \quad \sum_{x' \in \mathcal{X}_D^c} \sum_{c \in \mathcal{C}} \widehat{D}_{A,x'}^c \boldsymbol{\theta}^c \preccurlyeq \boldsymbol{\theta}_I + \sum_{x \in \mathcal{X}^c} \sum_{c \in \mathcal{C}} \widehat{O}_{A,x}^c \boldsymbol{\theta}^c$$

$$b) \quad \widehat{D}_{A,x'}^c \leq D_{A,x'}^{c*}, \ \forall c \in \mathcal{C}, \ \forall x' \in \mathcal{X}_D^c$$

$$c) \quad \widehat{O}_{A,x}^c \leq O_{A,x}^{c*}, \ \forall c \in \mathcal{C}, \ x \in \mathcal{X}^c$$

$$d) \quad \widehat{F}_{FUE,x}^c \leq F_{FUE,x}^{c*}, \ \forall c \in \mathcal{C}, \forall x \in \mathcal{X}^c \tag{4}$$

where constraint (4.a) ensures that the resources required to serve $\widehat{D}_{A,x'}^c$ slices from tenant x' do not exceed the current available InP resources $\boldsymbol{\theta}_I$ and additional resources from $\widehat{O}_{A,x}^c$ unused slices purchased to tenant x. The vector of resources allocated to each slice of class c is $\boldsymbol{\theta}^c = \boldsymbol{\theta}_A^c$. The InP controller assigns the resources to guarantee isolation. Constraint (4.b) limits the slices $\widehat{D}_{A,x'}^c$ allocated to tenant x' not to exceed the slices requested $D_{A,x'}^{c*}$. Constraint (4.c) means that the unused slices $\widehat{O}_{A,x}^c$ of class c bought from tenant x should not exceed the available unused slices $O_{A,x}^{c*}$. Finally, constraint (4.d) indicates that the FUEs from class c bought from tenant x $\widehat{F}_{FUE,x}^c$ should be less than the available FUEs $F_{FUE,x}^{c*}$. Recall that $D_{A,x'}^{c*}$, $O_{A,x}^{c*}$ and $F_{FUE,x}^{c*}$ are obtained by solving (3).

Tenants incentivize their subscribers to share their resources, whenever idle, as FUEs. To calculate the price $r_{FUE,x}^c$ of an FUE of class c, we assume that its task arrivals follow a Poisson distribution with an average arrival rate (λ_{TX}^c) and are independent and identically distributed (i.i.d). Thus, the probability of no task arrivals (available FUE) in Δ slots is

$$P_{TX}^c(\Delta) = e^{-\lambda_{TX}^c \Delta} \tag{5}$$

We define the price of an FUE of class c as

$$r_{FUE,x}^c = \Lambda_c * P_{TX}^c(\Delta) \tag{6}$$

which is proportional to its availability and Λ_c is a fixed price per unit of FUE's resources. The overall price decreases with Δ as the availability of the FUE decreases.

C. Modeling of InP's Decision

We formulate the InP's decision problem as a semi-Markov decision process (S-MDP) in which decisions are made once an event occurs (e.g., slice demand/offer arrival, FUE offer). The S-MDP is defined by the decision epoch, network state space, action space, transition probabilities, and reward.

1) Decision Epoch: the InP makes a decision upon receiving a request/offer from tenants. Thus, the decision epoch is the inter-arrival time between two consecutive decisions.

2) State Space: The state $\boldsymbol{s}_l$ in slicing mode $l = 1$ indicates the number of slices $\widehat{D}_{A,x'}^c$ of tenant $x' \in \mathcal{X}_D^c$ being served in the network, the number of unused augmented slices $\widehat{O}_{A,x}^c$ of tenant $x \in \mathcal{X}^c$ bought by the InP and the number of FUEs $\widehat{F}_{FUE,x}^c$ of tenant $x \in \mathcal{X}^c$ bought by the InP,

$$\boldsymbol{s}_l \triangleq \Big[\left(\widehat{D}_{A,1}^1, \widehat{O}_{A,2}^1, \widehat{F}_{FUE,2}^1 \right), \ldots, \left(\widehat{D}_{A,x'}^c, \widehat{O}_{A,x}^c, \widehat{F}_{FUE,x}^c \right), \ldots, \left(\widehat{D}_{A,X}^4, \widehat{O}_{A,1}^4, \widehat{F}_{FUE,1}^4 \right) \Big], \ \text{for } l = 1, \ c = \{1,2,3,4\} \tag{7}$$

Taking the InP's admission and collaboration constraints into consideration, the network state space is $\mathcal{S}_l \triangleq \{\boldsymbol{s}_l : (4.a) - (4.d)\}$. At the current state $\boldsymbol{s}_l$, four events can occur at the InP: a slice request arrives, an unused slice offer arrives, an FUE offer arrives to the InP, or otherwise.

3) Action Space: At state $\boldsymbol{s}_l$, the InP can take one of the following actions to maximize its long-term reward: accept the offered FUEs from tenant x ($a_{s_l} = 3$), accept the request from tenant x' ($a_{s_l} = 2$), accept the offered unused augmented

slices from tenant x ($a_{s_l}=1$), or reject any of the above ($a_{s_l}=0$). Therefore, the action space is described as follows:

$$\mathcal{A}_l \triangleq \{a_{s_l}\} = \{0,1,2,3\} \tag{8}$$

4) State Transition Probability: The slice requests from tenant x', the unused slice offers from tenant x, and the FUE offers from tenant x for each class c arrive at the network (under mode $l=1$) following a Poisson process with mean rates $\lambda^c_{x',1}$, $\widehat{O}^c_{A,x}\mu^c_{x,1}$, and $\widetilde{\lambda^c_{x,1}}$, respectively, where, $\mu^c_{x,1}$ is the rate at which the tenant x offers to sell back previously bought slices. Since the availability of fog resources is temporal and for a time less than the slice duration, we assume that the arrival and departure rates of slice requests and available fog nodes are independent. Every request from tenant x' and from class c has a lifetime that follows an exponential distribution with a mean rate $1/\mu^c_{x',1}$ and every FUE offer from class c from tenant x has availability that follows an exponential distribution with a mean rate $1/\widetilde{\mu^c_{x,1}}$. The next event can occur with rate Y_s that is described as $Y_s = \sum_{x=1}^{X}\sum_{c=1}^{C}\left(\lambda^c_{x,1}+\widehat{O}^c_{A,x}\mu^c_{x,1}+\widetilde{\lambda^c_{x,1}}+\widehat{F}^c_{FUE,x}\widetilde{\mu^c_{x,1}}\right)$ [27]. Therefore, the probability of the arrival of a slice request, an unused slice offer or an FUE offer is Y_s/Y, where $Y=\max_{s\in S} Y_s$. We assume that the occurrence rate of each event is the same and the transition epoch can be generated by a Poisson process with rate Y. The network transfers from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ ($\neq \boldsymbol{s}$) with probability $p_{\boldsymbol{s},\boldsymbol{s}'}(t)$. Therefore, the one-step transition probabilities can be written as follows

$$\overline{p}_{\boldsymbol{s},\boldsymbol{s}'}(t) = \begin{cases} (Y_{\boldsymbol{s}}/Y)\, p_{\boldsymbol{s},\boldsymbol{s}'}(t), & \boldsymbol{s}' \neq \boldsymbol{s}, \\ 1-(Y_{\boldsymbol{s}}/Y), & \text{otherwise}, \end{cases} \tag{9}$$

The probabilities $p_{\boldsymbol{s},\boldsymbol{s}'}(t)$, $t\leq 0$ are

$$p_{\boldsymbol{s},\boldsymbol{s}'}(t) = \sum_{n=0}^{\infty} e^{-Yt}\frac{Yt^n}{n!}\overline{p}^{(n)}_{\boldsymbol{s},\boldsymbol{s}'}(t), \quad \forall \boldsymbol{s}_l, \boldsymbol{s}_l' \in \boldsymbol{S}_l \tag{10}$$

Note that to derive the transition probabilities, we need information about the demand/offer arrivals and completion times, which may not be available and vary in time. For this reason, we develop DQ-E2E-INS algorithm in Section III-D to solve the optimal policy for the InP without requiring information about the environment.

5): The reward of the InP depends on the current state $\boldsymbol{s}_l \in \mathcal{S}_l$ and the action taken a_{s_l},

$$r_l(s_l, a_{s_l}) = \begin{cases} r^c_{D_A}, & \text{if a slice request arrives,} \\ & a_{s_l}=2, s_l' \in \mathcal{S}_l \\ -r^c_{O_A}, & \text{if an unused slice offer} \\ & \text{arrives}, a_{s_l}=1, s_l' \in \mathcal{S}_l \\ -r^c_{FUE,x}, & \text{if a FUE offer arrives,} \\ & a_{s_l}=3, s_l' \in \mathcal{S}_l \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

At state $\boldsymbol{s}_l \in \mathcal{S}_l$, if the slice demand/offered is accepted, i.e., $a_{s_l}=2$, the InP receives an immediate reward $r^c_{D_A}$ and the network moves to the next state s_l'. If the unused augmented slice offer is accepted, i.e., $a_{s_l}=1$, the InP pays $r^c_{O_A}$, and the network moves to the next state s_l'. If the FUE offer is accepted, i.e., $a_{s_l}=3$, the InP pays $r^c_{FUE,x}$ and the network moves to the next state s_l'. Finally, if the InP declines the demand/offer, the reward is 0. According to our S-MDP model, we can reformulate the InP's admission and collaboration problem to obtain an optimal policy $\pi_l{}^*$ that maps the state space to the action space $\mathcal{S}_l \rightarrow \mathcal{A}_l$ and maximizes the long-term average reward of the InP,

$$\max_{\pi_l} R_l(\pi_l) = \lim_{T\to\infty}\mathbb{E}\sum_{t=1}^{T}\left(r_l\left(s_l^t, \pi_l\left(s_l^t\right)\right)\right)/T, \tag{12}$$

where $r_l(s_l^t, \pi_l(s_l^t))$ is the immediate reward at decision epoch t under policy π_l. Solving this problem entails high complexity due to the network dynamics. To find the optimal policy and deal with uncertain and dynamic demands, we adopt Deep Q-learning algorithms in the sequel.

D. DQ-E2E-JNS Algorithm

We develop a *DQ-E2E-JNS* algorithm to solve the InP's admission and collaboration problem. The DQN is run in a centralized cloud at the InP and uses historical information. Through learning and building knowledge of the joint network slicing mode, the optimal policy can be obtained. The InP will forward the cloud the needed information to build the system state s_l^t as in (7). Then, the agent chooses an action a_l^t depending on the current policy. The algorithm uses the ϵ-greedy policy in choosing the action to balance the exploitation and exploration and improve the reward. Then, the algorithm measures the output in terms of reward r_l^t and moves to the next state s_l^{t+1}. The agent's experience which includes the current state, action, reward, and next state are stored in the replay memory. The experience samples are utilized to train the value function $Q(s_l, a_l)$ and target value function $\widehat{Q}(s_l, a_l)$. As described in Algorithm 1, $Q(s_l, a_l)$ is trained each step towards $\widehat{Q}(s_l, a_l)$ by minimizing the loss function at each iteration

$$\begin{aligned} Loss(w_1, w_2) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')} & \left[\left(r_l^t + \Psi \max_{a_l^t}\widehat{Q}\left(s_l^t, a_l^t; {w_1}^-, {w_2}^-\right)\right.\right. \\ & \left.\left. - Q\left(s_l^t, a_l^t; w_1, w_2\right)\right)^2\right], \end{aligned} \tag{13}$$

where w_1 and w_2 are the parameters of the primary Q-network (value function $Q(s_l, a_l)$) and ${w_1}^-$ and ${w_2}^-$ are the parameters of the target Q-network (target value $\widehat{Q}(s_l, a_l)$).

To improve the convergence of our algorithm, we adopt dueling which uses two streams of fully connected hidden layers, i.e., the values of the states $\alpha_1(s_l; w_2)$ and the advantage values of actions $L(s_l, a_l; w_1)$ to simultaneously train the learning process of the Q-learning algorithm [28]. They represent the importance of being in state s_l and the significance of executing action a_l in comparison with other potential actions, respectively. Then, the value function is calculated by combining the two values as follows

$$\begin{aligned} Q(s_l, a_l; w_1, w_2) = \alpha_1(s_l; w_2) + L(s_l, a_l; w_1) \\ - \frac{1}{|A_l|}\sum_{a_l'} L(s_l, a_l'; w_1), \end{aligned} \tag{14}$$

where $|A_l|$ is the number of all available actions.

E. Outage

Occasionally, FUEs need their resources back to transmit their traffic, interrupting the ongoing connection and affecting the availability of resources in the augmented slices and fog slices. Therefore, the probability of outage in the transmission

Algorithm 1 Dueling DQ-E2E-JNS Algorithm

Initialize: the discount factor value Ψ, experience reply buffer H to capacity Cap, epsilon ϵ, and the learning rate value Ω_t.
Initialize: the primary network Q and target network $\widehat{Q}$ with random weights w_1, w_2, and ${w_1}^- = w_1$, ${w_2}^- = w_2$, respectively.
For $t = 1$ *to* T **do**
  Tenants receive UEs' requests and find available unused slices and FUEs.
  Tenants calculate UE's expected utilities and request slices or offer unused slices or FUEs to InP as in (3).
  With probability ϵ, select a random action a_l^t, otherwise select action $a_l^t = arg\max_{a_l^t} Q^*\left(s_l^t, a_l^t; w_1, w_2\right)$
  Observe r_l^t and s_l^{t+1}based on executed action a_l^t.
  Put $(s_l^t, a_l^t, r_l^t, s_l^{t+1})$ in H
  Sample random minibatch of $(s_{l_i}, a_{l_i}, r_{l_i}, s_{l_{i+1}})$ from the experience reply buffer.
  Calculate $Q\left(s_{l_i}, a_{l_i}; w_1, w_2\right)$ as in (14)
  Update w_1, w_2 by minimizing the loss as in (13)
  Replace $\widehat{Q}$ with Q every U steps
EndFor

of UE j in an augmented slice or fog slice from class c due to unavailability of FUEs is

$$p_{out,j}^c = 1 - Pr\left[T_j^c\left(\boldsymbol{\theta}_j^c\right) \geq T_{min}^c \,\&\, \tau_j^c(\boldsymbol{\theta}_j^c) \leq \tau_{max}^c\right], \quad \forall j \in \mathcal{J}_A^c, c \in \mathcal{C} \tag{15}$$

In a similar vein, the outage probability due to using a certain radio spectrum band to serve UEs of class c is calculated by using (15). We assume that the packet arrival rate of each user j increases inversely proportional to the probability of no-outage, $\lambda_{eq,j}^c = \lambda_j/(1 - p_{out,j}^c)$. Therefore, if the transmission is unsuccessful, the users will retransmit accordingly with a rate equal to $\lambda_{eq,j}^c$.

IV. End-to-End Independent Network Slicing (E2E-INS)

To utilize FUEs' resources more efficiently and guarantee strict delay requirements, we propose a second slicing mode based on a new E2E independent network slicing scheme (E2E-INS). This scheme combines fixed InP slices created using InP's resources and fog slices consisting of available FUEs resources. The optimum slice allocation is solved in two steps. First, we determine the InP's optimal admission and collaboration policy. Then we formulate the tenants' optimization problem as a Markov game to buy/sell fog slices and solve it using a multi-agent actor-critic algorithm that incorporates matching for FUE trading. Implementation details are provided in Section V.

A. InP's Optimal Admission and Collaboration Policy

In this scheme, the InP's optimal admission and collaboration policy determines the optimal acceptance/rejection of slice requests using only InP resources and the collaboration with tenants to buy back temporarily unused slices from those previously accepted. The remaining requests from each tenant will be served by combining the InP slices with fog slices as explained in Section IV-B. Let us denote by $D_{I,x'}^{c*}$ and $O_{I,x}^{c*}$ the optimal number of slices requested and offered to the InP from class c by tenant $x' \in \mathcal{X}_D^c$ and $x \in \mathcal{X}^c$, respectively. They are obtained by solving the tenant's optimization problem in (3) when the InP charges a price $r_{D_I}^c$ per accepted slice from class c and pays a price $r_{O_I}^c$ for per unused slice from the same class. In response to this demand and offer, the InP accepts $\widehat{D}_{I,x'}^{c*}$ and $\widehat{O}_{I,x}^{c*}$ slice requests from tenant x' and unused slice offers from tenant x, respectively, by solving $\mathcal{P}_I(\widehat{D}_{I,x'}^{c*}, \widehat{O}_{I,x}^{c*}, 0)$ in (4). Recall that the vector of resources allocated per slice contains only InP's resources, $\boldsymbol{\theta}^c = \boldsymbol{\theta}_I$. Modeling the InP's decision is straightforward and is omitted in the interest of space. In fact, Algorithm 1 can solve the admission and collaboration policy for the new state vector $\boldsymbol{s}_l \triangleq \left[\{\widehat{D}_{I,1}^c, \widehat{O}_{I,2}^c\}, \ldots, \{\widehat{D}_{I,x'}^c, \widehat{O}_{I,x}^c\}, \ldots, \{\widehat{D}_{I,X}^c, \widehat{O}_{I,1}^c\}\right]$, where $l = 2$ and $c \in \mathcal{C}$. The reward function $r_l\left(s_l, a_{s_l}\right)$ equals $r_{D_I}^c$ if an InP slice request arrives and action $a_{s_l} = 2$ is taken, $-r_{O_I}^c$ if an unused InP slice offer arrives and action $a_{s_l} = 1$ is taken, or 0, otherwise and action $a_{s_l} = 0$.

B. Tenants' Fog Slicing Optimization Problem

The remaining requests $D_{I,x'}^{c*} - \widehat{D}_{I,x'}^{c*}$ from each tenant $x' \in \mathcal{X}_D^c$ will be served by using fog slices created from available FUEs from any class $c \in \mathcal{C}$ and any tenant $x \in \mathcal{X}$. Tenants serve their remaining slice requests first by utilizing their own FUEs, and if there are remaining FUEs available, they will sell them to other tenants. Therefore, a tenant becomes a fog slice seller tenant $x_f \in \mathcal{X}_f$ if it has available FUEs or a fog slice buyer tenant $x_f' \in \mathcal{X}_{D_f}$ if it has remaining slice requests. For simplicity, we refer to these tenants as sellers and buyers, respectively. The FUEs' resources will be used to create fog slices of any class c, independently of their initial class as subscribers, for as long as their resources are sufficient to serve that class. To this end, we formulate the matching problem between buyers and sellers, which is solved every time there is a new demand. A buyer x_f' is matched to a seller x_f, indicated as $\delta_{x_f', x_f} = 1$, if it is the one that can serve the highest number of requests. Otherwise, $\delta_{x_f', x_f} = 0$. At the same time, each seller is matched to the buyer that results in the highest reward. The seller slices its available FUEs' resources as fog slices to serve different classes from the buyer tenant, so the buyer will request a number of fog slices D_{x_f', x_f}^c to satisfy the requirements from its UEs of class $c \in \mathcal{C}$. The price per fog slice of class c is $r_{D_f}^c$. We assume that each buyer is matched to at most one seller, and each seller can serve at most one buyer at a time. Therefore, the optimization problem for buyer $x_f' \in \mathcal{X}_{D_f}$ is formulated as

$$\underset{\delta_{x_f', x_f}, D_{x_f', x_f}^{c'}}{\text{maximize}} \quad \sum_{x_f \in \mathcal{X}_f} \delta_{x_f', x_f} \left[\sum_c \sum_{j \in \mathcal{J}_f^c} \Gamma_c U_j^c - \sum_{c' \in \mathcal{C}_{D,x_f'}} D_{x_f', x_f}^{c'} r_f^{c'}\right] - \sum_{c' \in \mathcal{C}_{D,x_f'}} \left[\widehat{D}_{I,x_f'}^{c'*} r_{D_I}^{c'}\right] + \sum_{c \in \mathcal{C}_{x_f'}} \left[\widehat{O}_{I,x_f'}^{c*} r_{O_I}^c\right]$$

subject to

$$a) \quad T_j^{c'}(\boldsymbol{\theta}_{f,j}^{c'}) \geq T_{min}^{c'}, \quad \forall j \in \mathcal{J}_f^c \text{ or } \mathcal{J}_I^c, \forall c' \in \mathcal{C}_{D,x_f'}$$

$$b) \quad \tau_j^{c'}(\boldsymbol{\theta}_{f,j}^{c'}) \leq \tau_{max}^{c'}, \quad \forall j \in \mathcal{J}_f^c \text{ or } \mathcal{J}_I^c, \ \forall c' \in \mathcal{C}_{D,x_f'}$$

$$c) \quad \sum_{c' \in \mathcal{C}_{D,x'_f}} (D^{c'*}_{I,x'} - \widehat{D}^{c'*}_{I,x'}) \boldsymbol{\theta}^{c'} \succcurlyeq \sum_{c' \in C_{x_f}} D^{c'}_{x'_f,x_f} \, \boldsymbol{\theta}^{c'}_f$$

$$d) \quad \sum_{x_f \in \mathcal{X}_f} \delta_{x'_f,x_f} \leq 1 \tag{16}$$

where U^c_j is the utility function of UE j from class c subscribed to the buyer tenant and served by a fog slice in the set $\mathcal{J}^c_f$. U^c_j is defined as in Section III-A. $\widehat{D}^{c'*}_{I,x'_f}$ and $\widehat{O}^{c*}_{I,x'_f}$ are the optimal number of InP slices of class c' allocated to tenant x'_f and unused InP slices of class c that the same tenant sold back to the InP, respectively. Constraints (16.a) and (16.b) ensure that UE j achieves the QoS requirement when served by a fog slice from class c' with $\boldsymbol{\theta}^{c'}_f = (b^{c'}_f, g^{c'}_f, v^{c'}_f)$ FUEs' resources. Constraint (16.c) guarantees that the overall amount of fog resources in $D^{c'}_{x'_f,x_f}$ fog slices requested from tenant x_f do not exceed its remaining resources. Finally, constraint (16.d) indicates that a given buyer x'_f can only be served by one seller x_f.

Once the seller $x_f \in \mathcal{X}_f$ receives the requests from multiple buyers, the seller will maximize its reward as

$$\underset{\widehat{\delta}_{x_f,x'_f}, \widehat{O}^{c'}_{x_f,x'_f}}{\text{maximize}} \sum_{x'_f \in \mathcal{X}_{D_f}} \sum_{c' \in \mathcal{C}_{x_f}} \widehat{\delta}_{x_f,x'_f} \widehat{O}^{c'}_{x_f,x'_f} r^{c'}_f - \sum_{c \in \mathcal{C}_{D,x_f}} \left[\widehat{D}^{c*}_{I,x_f} r^{c}_{D_I} \right] + \sum_{c' \in \mathcal{C}_{x_f}} \left[\widehat{O}^{c'*}_{I,x_f} r^{c'}_{O_I} \right]$$

subject to

$$a) \quad \sum_{x'_f \in \mathcal{X}_{D_f}} \sum_{c' \in \mathcal{C}_{x_f}} \widehat{O}^{c'}_{x_f,x'_f} \, \boldsymbol{\theta}^{c'}_f \preccurlyeq \sum_{c' \in \mathcal{C}_{x_f}} E^{c'}_{FUE,x_f} \boldsymbol{\theta}^{c'}_{FUE}$$

$$b) \quad \widehat{O}^{c'}_{x_f,x'_f} \leq D^{c'}_{x'_f,x_f}, \ \forall c' \in \mathcal{C}_{x_f}$$

$$c) \quad \sum_{x'_f \in \mathcal{X}_{D_f}} \widehat{\delta}_{x_f,x'_f} \leq 1 \tag{17}$$

where the first term is the revenue obtained by selling $\widehat{O}^{c'}_{x_f,x'_f}$ fog slices to buyer x'_f, the second one is the cost of buying $\widehat{D}^{c*}_{I,x_f}$ slices from the InP at a price $r^c_{D_I}$, and the third one is the revenue obtained by selling back $\widehat{O}^{c'*}_{I,x_f}$ unused slices to the InP at price $r^{c'}_{O_I}$. Constraint (17.a) ensures that the fog resources $\boldsymbol{\theta}^{c'}_f$ allocated to $\widehat{O}^{c'}_{x_f,x'_f}$ fog slices for each class c do not exceed the current available FUEs' resources $E^{c'}_{FUE,x_f}$ in tenant x_f, where $\boldsymbol{\theta}^{c'}_{FUE}$ is the vector of FUEs' resources. Constraint (17.b) states that the fog slices sold $\widehat{O}^{c'}_{x_f,x'_f}$ should not exceed the fog slices requested $D^{c'}_{x'_f,x_f}$ per class c'. Finally, in constraint (17.c), $\widehat{\delta}_{x_f,x'_f}$ equals 1 if seller x_f serves buyer x'_f, or 0, otherwise, and seller x_f can serve only one buyer tenant. Solving the buyers' and sellers' pairing optimization problems is challenging because of the uncertainty of other sellers' policies and buyers' service demands. To overcome this non-stationary environment and the multi-agent credit assignment problem [29], a cooperative Multi-Agent Actor-Critic (MAAC) framework is implemented in which each seller learns its best policy based on centralized-training and distributed-execution design. Our algorithm implements a multi-agent dueling DQ network-based scheme combined with distributed coordinated learning to effectively learn the optimum FUE's trading policy. By distributed coordinated learning and dueling network, the learning scheme can rapidly converge to the optimum policy.

The price r^c_f that buyer x'_f pays to seller x_f per fog slice of class c is proportional to the urgency of the service and the availability of FUEs' resources in the fog slice. The ratio $\left(\frac{\tau^c_r}{\Delta}\right)$ between the remaining time slots τ^c_r from class c to complete the service and the duration of the slice Δ given in the number of time slots represent the urgency of the slice request. The probability of no task arrivals P^c_{TX} for any FUE during the duration of the urgent slice request ζ is obtained by (5). Thus, the price for a fog slice from class c is obtained as

$$r^c_f = u_c \left(\frac{\tau^c_r}{\Delta}\right) P^c_{TX}(\zeta) \tag{18}$$

where u_c is the nominal price of a fog slice from class c. This pricing function gives more priority to slice requests with less remaining slots to complete the transmission. For instance, the price of an MBBMTC, MBBRLLC or MBBRLLMTC slice request gradually decreases until the request is accepted or expires, i.e., $\tau^c_r = 0$. However, the price of an RLLMTC slice request with the lowest delay requirement has the highest priority, i.e., $\tau^c_r = 1$.

C. Modeling Multi-Agent Environment

We model the multi-agent environment using a partially observable Markov game in which each agent $x_f \in \mathcal{X}_f$ corresponds to a seller interacting with the environment. In a multi-agent environment, agents update their policies independently and simultaneously, making the environment non-stationary. The framework of Markov games admits multiple adaptive agents, i.e., $X_f = |\mathcal{X}_f|$, with competing goals. We define the Markov decision game for fog slice trading by the tuple $(\mathcal{S}, \ \mathcal{A}, r_1, \ldots, r_{X_f}, p_l, \Psi)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r_{x_f} is the reward function of agent x_f and p is the transition probability $p = Pr\{\boldsymbol{s}_{x_f}(t+1) | \boldsymbol{s}_{x_f}(t), a_1, \ldots, a_{X_f})\}$ from the current state $\boldsymbol{s}_{x_f}(t) \in \mathcal{S}$ to the next state $\boldsymbol{s}_{x_f}(t+1) \in \mathcal{S}$ when all agents take actions $\{a_{x_f} \in \mathcal{A}, x_f \in \mathcal{X}_f\}$ simultaneously, and Ψ is the discount factor, $\Psi \in [0, 1)$. All agents take their actions (select a buyer) according to a policy π_{x_f} that takes into consideration the actions of other agents. Based on the optimal policies, each seller finds the best buyer to offer its resources. Since sellers know the availability of their FUEs, we assume they are the agents in this trading game. From now on, we use interchangeably the terms "agent" and "seller" unless otherwise stated.

a) State space: The state $\boldsymbol{s}_{x_f}$ of the fog slice trading environment has two components: the number of fog slices $\widehat{O}^{c'}_{x_f,x'_f}$ of buyer x'_f from class c being served/run in the network by seller x_f and the number of remaining available FUEs $E^{c'}_{FUE,x_f}$ of seller x_f from class c,

$$\boldsymbol{s}_{x_f} \triangleq [\widehat{O}^1_{x_f,x'_f}, E^1_{FUE,x_f}, \ldots, \widehat{O}^{c'}_{x_f,x'_f}, E^{c'}_{FUE,x_f}, \ldots, \widehat{O}^C_{x_f,x'_f}, E^C_{FUE,x_f}] \tag{19}$$

The state space under each seller's resource constraints is $\mathcal{S} \triangleq \{\boldsymbol{s} : \forall \boldsymbol{s}_{x_f} \in \boldsymbol{s}, (17.a) - (17.c)\}$, where $\boldsymbol{s}$ is vector of states of all agents, $\boldsymbol{s} \triangleq [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{x_f}, \ldots, \boldsymbol{s}_{X_f}], \ \forall x_f \in \mathcal{X}_f$.

b) Action space: At the network state $\boldsymbol{s}_{x_f}$, the action of seller x_f is to choose which buyer $x'_f \in \mathcal{X}_{D_f}$ to offer the fog slices, $a_{x_f} = x'_f$, or $a_{x_f} = 0$ if the seller refuses to choose any buyer. Therefore, the action space of seller x_f is $\mathcal{A} = \left\{\boldsymbol{a} : \forall a_{x_f} \in \left\{\mathcal{X}_{D_f} \cup\right\}\right\}$, and $\boldsymbol{a} \triangleq [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{x_f}, \ldots, \boldsymbol{a}_{X_f}]$, $\forall x_f \in \mathcal{X}_f$.

c) Reward: Seller x_f receives an immediate reward $r\left(\boldsymbol{s}_{x_f}, a_{x_f}, \boldsymbol{a}_{-x_f}\right)$ after executing action a_{x_f} while other sellers executed action $\boldsymbol{a}_{-x_f}$ at network state $\boldsymbol{s}_{x_f} \in \mathcal{S}$ and the next state is $\boldsymbol{s}'_{x_f} \in \mathcal{S}$,

$$r_{x_f}\left(\boldsymbol{s}_{x_f}, a_{x_f}, \boldsymbol{a}_{-x_f}\right) = \begin{cases} \sum_{c'=1}^{C} \widehat{O}^{c'}_{x_f, x'_f} + cte, & \text{if } a_{x_f} = x'_f \text{ and } x'_f \text{ selected } x_f \\ -\hbar & \text{if } a_{x_f} = x'_f \text{ and } x'_f \text{ does not select } x_f \\ 0, & \text{if } a_{x_f} = 0 \end{cases} \tag{20}$$

where the constant $cte = -\sum_{c \in \mathcal{C}_{D,x_f}} \left[\widehat{D}^{c*}_{I,x_f} r^{c}_{D_I}\right] + \sum_{c' \in \mathcal{C}_{x_f}} \left[\widehat{O}^{c'*}_{I,x_f} r^{c'}_{O_I}\right]$ and $\hbar$ is a punishment for the agents to discourage them from offering the resources to the same buyer at the same time so FUEs' resources can be utilized efficiently, a_{x_f} is the action of seller x_f and $\boldsymbol{a}_{-x_f}$ are the actions of other sellers except x_f.

The goal of each agent is to learn a policy π_{x_f} which maximizes the expected cumulative discounted reward

$$R_{x_f}(\pi_{x_f}) = \mathbb{E}\left[\sum_{t=0}^{T} \Psi^t r_{x_f}(t)\right]. \tag{21}$$

D. Multi-Agent Actor-Critic for Fog Slice Allocation

We adopt a Multi-Agent Actor-Critic (MAAC) framework to find the optimal policy for each seller by considering the action policies of other sellers and utilizing cooperation among agents to enhance the performance. It is worth mentioning that collaboration occurs during centralized training, but agents execute their decisions distributively. To this end, the AC framework divides the agent into two parts: the actor that is responsible for selecting the action and the critic that criticizes the actor for reaching the best decision. The MAAC framework is an extension of AC. The MAAC framework for pairing buyers and sellers is shown in Fig. 2. Each seller is supported by an autonomous agent x_f. The actor is responsible for selecting a proper action based on the observed state. Then the critic evaluates the quality of the chosen action depending on the extra information on the actions and states of the other sellers. Thus, to facilitate the training process, we allow the policies to exploit this information while training. We assume that the tenant has the necessary cloud computing resources to implement the centralized training process. Agent x_f in the decentralized execution process downloads the trained weight of the actor from the cloud and loads it into its own actor x_f. The actor x_f chooses action (buyer) a_{x_f} based on the observed state $\boldsymbol{s}_{x_f}$ and receives a reward r_{x_f}. Each agent uploads to the cloud the historical information, including $(\boldsymbol{s}_{x_f}, a_{x_f}, r_{x_f})$ collected during the execution process for the next training process.

Using the framework of MAAC, the environment becomes stationary as we know the actions executed by other sellers even though their policies change in time [29]. Once the actions are known, the transition probability can be determined $Pr(\boldsymbol{s}_{x_f}(t+1) \mid \boldsymbol{s}_{x_f}(t), a_1, \ldots, a_{X_f}, \pi_1, \ldots, \pi_{X_f}) = Pr(s_{x_f}(t+1) \mid \boldsymbol{s}_{x_f}(t), a_1, \ldots, a_{X_f})$. Let $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{X_f}\}$ be the set of sellers' strategies. The critic of agent x_f takes the actions and states of the other agents to evaluate the action of its own agent using the centralized action-value function $Q^{\boldsymbol{\pi}}_{x_f}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{s}', r_{x_f} \sim E}\left[r_{x_f} + \Psi \mathbb{E}_{\boldsymbol{a}' \sim \boldsymbol{\pi}} Q^{\boldsymbol{\pi}}_{x_f}(\boldsymbol{s}', \boldsymbol{a}')\right]$, where $\boldsymbol{s}$ and $\boldsymbol{a}$ include the states and actions of all agents, respectively.

Let us extend the previous framework to deterministic policies. If we consider X_f deterministic policies (actor) with $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$ the set of all policies, we can optimize the function approximator of the centralized action-value function Q_{x_f} of agent x_f parameterized by $\Omega^Q_{x_f}$ by minimizing

$$Loss\left(\Omega^Q_{x_f}\right) = \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a}, r_{x_f}, \boldsymbol{s}'}[\Big(Q_{x_f}\left(\boldsymbol{s}, \boldsymbol{a} | \Omega^Q_{x_f}\right) - \left(r_{x_f} + \Psi Q_{x_f}\left(\boldsymbol{s}', \boldsymbol{\mu}(s') | \Omega^Q_{x_f}\right)\right)\Big)^2], \tag{22}$$

Considering the deterministic policy μ_{x_f} parameterized by $\Omega^\mu_{x_f}$ using the deterministic policy gradient scheme, we can calculate the gradient of the expected return for agent x_f

$$\nabla_{\Omega^\mu_{x_f}} \mathbb{E}\left[R_{x_f}\right] = \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a} \sim \boldsymbol{H}}\left[\nabla_{a_{x_f}} Q_{x_f}\left(\boldsymbol{s}, \boldsymbol{a} | \Omega^Q_{x_f}\right) |_{a_{x_f} = \mu_{x_f}(s_{x_f})}\right] \times \nabla_{\Omega^\mu_{x_f}} \mu_{x_f}\left(s_{x_f} \,\middle|\, \Omega^\mu_{x_f}\right) \tag{23}$$

where H is the replay buffer that contains the tuples $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')$ recording experiences of all agents. At each time the critic and actor are updated by sampling a minibatch uniformly from the buffer.

The mapping between the state and action space of the actor and the action-value function of the critic needs to be approximated by function approximators. To approximate the mapping in large state and action spaces, we incorporate deep learning into MAAC. Let $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$ and $\boldsymbol{Q} = \{Q_1, \ldots, Q_N\}$ denote the set of actor networks and critic networks of all agents, respectively, with their corresponding weights $\Omega^{\boldsymbol{\mu}} = \{\Omega^\mu_1, \ldots, \Omega^\mu_N\}$ and $\Omega^{\boldsymbol{Q}} = \left\{\Omega^Q_1, \ldots, \Omega^Q_N\right\}$. Both actor and critic networks are fully connected networks. The input of the actor network is the state observed by the agent, while the output is the chosen action. In the critic network, first we enter the states of all agents $\boldsymbol{s}$, following a fully connected layer. Then the actions of all agents $\boldsymbol{a}$ go through several fully connected layers until the output $\boldsymbol{Q}^{\boldsymbol{\pi}}_{x_f}(\boldsymbol{s}, \boldsymbol{a})$ is obtained. The MAAC-E2E-INS algorithm is described in Algorithm 2 and centralized training for the MAAC architecture is shown in Fig. 2. The MAAC framework utilizes historical data to train the deep neural networks of the actor and critic and returns the weights of actor network Ω^μ as described in the execution process in Algorithm 2. The computational complexity is derived in the Appendix.

V. Slice Management System (SELF-E2E-NS)

The orchestrator has no information about the available resources in the network and thus it builds a belief in InP resource availability. This belief is used to decide which slicing mode l to activate to serve the slice requests from each class c. The orchestrator will activate the mode with the lowest risk of non-satisfying the service based on previous decisions in which the belief of available InP resources was the same. The algorithm is described in Algorithm 3, and the belief

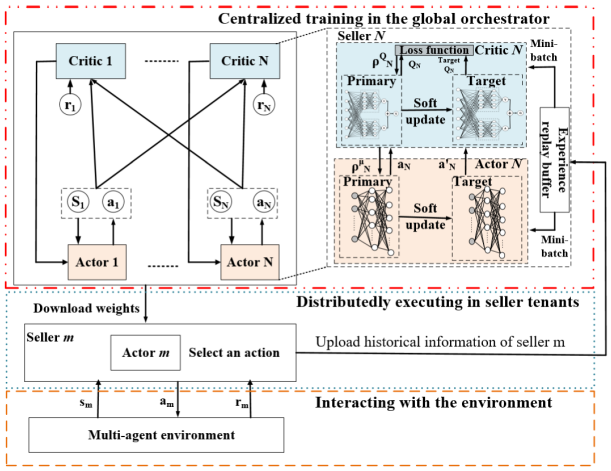


Fig. 2. Multi-agent actor-critic architecture ($N = X_f$, $m = x_f$).

Algorithm 2 MAAC-E2E-INS

Input: Critic and actor network structures.
Output: Ω^{μ} of actor network.
Training:
Initialize randomly $\boldsymbol{Q}$ and $\boldsymbol{\mu}$ with $\Omega^{\boldsymbol{Q}}$ and Ω^{μ} .
Seller receives initial states $\boldsymbol{s}^0 = \{s_1^0, \ldots, s_{X_f}^0\}$
For $t = 1$ *to* T **do**
Sellers choose actions $\boldsymbol{a}^t = \{a_1^t, \ldots.., a_{X_f}^t\}$ based on current policy where $a_{x_f}^t = \mu_{x_f}(s_{x_f}^t)$.
Sellers execute actions $\boldsymbol{a}^t$, receive rewards $\boldsymbol{r}^t = \{r_1^t, \ldots.., r_{X_f}^t\}$ and observe new states $\boldsymbol{s}^{t+1}$.
Save $(\boldsymbol{s}^t, \boldsymbol{a}^t, \boldsymbol{r}^t, \boldsymbol{s}^{t+1})$ in H
Sample random minibatch of $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')$ from the experience reply buffer.
Update critic by minimizing (22).
Update the actor policy based on (23).
End For
Input: Actor network structure, $\Omega^{\mu'}$.
Execution:
Load $\Omega^{\mu'}$ to the actor network.
Sellers receive initial observation states $\boldsymbol{s}^0 = \{s_1^0, \ldots, s_{X_f}^0\}$
For $t = 1$ *to* T **do**
Sellers choose actions $\boldsymbol{a}^t = \{a_{x_f}^t = \mu'_{x_f}(s_{x_f}^t), x_f \in \mathcal{X}_f\}$ according to current policy.
Sellers execute actions $\boldsymbol{a}^t$ and observe new states $\boldsymbol{s}^{t+1}$.
End For

calculation is detailed in the Appendix. The probability of satisfying the service $\xi_{x,l}^c$ for a tenant x requesting a slice from class $c \in \mathcal{C},\ c = \{1, 2, 3, 4\}$ using mode l is

$$\xi_{x,1}^c = \mathfrak{p}_{x,1}^c FB_{x,1}^c = \frac{\widehat{D}_{A,x}^{c*}}{D_{A,x}^{c*}} \left(1 - \sum_{j \in \mathcal{J}_A^c} p_{out,j}^c\right) \tag{24}$$

$$\xi_{x,2}^c = \mathfrak{p}_{x,2}^c FB_{x,2}^c = \frac{\widehat{D}_{I,x}^{c*} + \widehat{O}_{x_f,x_f'}^c}{D_{I,x}^{c*}} \left(1 - \sum_{j \in \{\mathcal{J}_I^c \cup \mathcal{J}_f^c\}} p_{out,j}^c\right) \tag{25}$$

where $\mathfrak{p}_{c,1}^x$ and $\mathfrak{p}_{c,2}^x$ are the acceptance probabilities and $FB_{x,1}^c$ and $FB_{x,2}^c$ are the feedback from the tenants about the service

Algorithm 3 SELF-E2E-NS

At time t, orchestrator receives D^{c*} slice requests from class c.
Obtain the belief of state of available InP resources s_{I_l} per mode l; $p_{belief}(s_{I_l}(t)) = \{y_l^*\} = \text{argmax}_{y_l}\ \{p_{belief}(y_l)\}$.
For $t' = aux_t : t-1$
  For $c = 1:4$
    If $p_{belief}(t') = y_l^*$ and $D^c(t') = D^{c*}$
      If $risk_{l=1}^c(t') < risk_{l=2}^c(t')$
        Activate mode $l = 1$.
      Else If
        Activate mode $l = 2$.
      End If
    End If
  End For
End For
Once the slice is served update history sample at t with the $risk_l^c(t)$ and $p_{belief}(t) = s_{I_l}(t)$.

satisfaction using mode 1 and 2, respectively. $\mathfrak{p}_{c,1}^x$ is obtained as the ratio between the accepted requests $\widehat{D}_{A,x}^{c*}$ by InP and the number of slice requests $D_{A,x}^{c*}$ by tenant x. $\mathfrak{p}_{c,2}^x$ is the ratio between the accepted requests by InP $\widehat{D}_{I,x}^{c*}$ and tenant x'_f $\widehat{O}_{x'_f,x_f}^c$ and the number of slice requests $D_{I,x}^{c*}$ by tenant x. The feedback $FB_{x,1}^c$ and $FB_{x,2}^c$ received from the tenants is defined as the probability of no outage in the transmission for each user $j \in \mathcal{J}_A^c$ and $j \in \{\mathcal{J}_I^c \cup \mathcal{J}_f^c\}$, respectively. The outage compromises satisfying the SLA.

The objective of the orchestrator is to select the slicing mode l that minimizes the risk of not serving the request and achieving the SLA requirement, as defined in (1), for each request of class c from tenant x as follows

$$\begin{aligned} \underset{Z_{x,l}^c}{\text{minimize}} \quad & \sum_x \sum_c Z_{x,l}^c risk_{x,l}^c \\ \text{subject to} \quad & \sum_l Z_{x,l}^c \leq 1 \end{aligned} \tag{26}$$

where $Z_{x,l}^c$ is binary indicator ($Z_{x,l}^c \in (0,1)$) in which $Z_{x,l}^c = 1$ means that the global orchestrator selects the scheme l (i.e., $= 1$ for E2E-JNS or $l = 2$ for E2E-INS) to serve the slice request from class c requested by tenant x, otherwise 0. The risk is low when the probability of non-satisfying the service is high, and vice versa. The constraint indicates that a slice request cannot be sent to both schemes simultaneously.

Once the operational mode l is activated, the slice request is forwarded to the corresponding scheme. The InP accepts or rejects the requests based on its admission and collaboration policy π_l. We have two cases:

- Request forwarded to E2E-JNS scheme ($l = 1$): If $\widehat{D}_{A,x}^{c*}$ slice requests of tenant x are accepted in the E2E-JNS scheme, tenant x cooperates with the InP by sharing $F_{FUE,x}^{c*}$ subscribers resources (FUEs) and $O_{A,x}^{c*}$ unused slices. The InP's policy π_1 takes into account the unused slices sold back by tenants to serve more requests. The local controller at the tenant has information about the location and availability of the FUEs and shares that information with the global controller (InP controller) to create augmented slices. The global controller has information about the InP available resources.

TABLE II
REQUIRED RESOURCES FOR EACH SLICE PER CLASS

Class	Radio b^c	Computing g^c	Storage v^c
MBBMTC	20 GHz	64.2 Mcyc/sec	354 Kbit/sec
MBBRLLC	20 GHz	700 Mcyc/sec	3.82 Mbit/sec
RLLMTC	800 MHz	14.28 Kcyc/sec	40 Kbit/sec
MBBRLLMTC	80 GHz	2.7 Gcyc/sec	15 Mbit/sec

TABLE III
REWARD AND PRIORITY WEIGHT PER CLASS AND EACH SLICE RESOURCE

Class	φ_I^c	$r_{D_I}^c$	φ_A^c	$r_{D_A}^c$	φ_F^c	r_{f_c}
MBBMTC	5.5	5	8.8	8	11	10
MBBRLLC	15	15	18	18	5.9	6
RLLMTC	25	18	4.2	3	5.5	4
MBBRLLMTC	8.3	10	10.8	13	6.6	8

TABLE IV
COST PER RESOURCE UNIT

Resource	Cost per resource unit
Computing	\$0.15/(Gcyc/sec)
Storage	\$0.25/(Gb/sec)
Radio (microwave band)	\$0.0009/MHz
Radio (mmWave band)	\$0.045/GHz
Radio (THz band)	\$0.01/GHz

- Request forwarded to E2E-INS scheme ($l = 2$): If $\widehat{D}_{I,x}^{c*}$ slice requests of tenant x are accepted in the E2E-INS scheme, tenant x cooperates with the InP to serve the remaining demand $D_{I,x}^{c*} - \widehat{D}_{I,x}^{c*}$ by using fog slices that contain its own and other tenants' FUEs. The local controller at the tenant informs the global controller (InP controller) to create fog slices in which FUEs act as relays and forward the transmission to the nodes in the InP slices to complete the transmission. We assume that the local controller has access to cloud computing resources to train the MAAC-E2E-INS algorithm.

By relying on feedback from tenants and InP, the orchestrator learns to activate the slicing mode with the highest probability of satisfying the request. Once the slice is served, the belief is updated with the current state information from the InP, and the risk is obtained and stored for future decisions.

VI. SIMULATION RESULTS

A. Simulation Setup

We conduct extensive simulations to illustrate the performance of our approaches and compare them with existing schemes. The simulations are implemented using TensorFlow. Unless otherwise stated, the simulation parameters are given in Tables II-V. We assume that there are four tenants in the network. The number of users of each class requiring service from each tenant is obtained by drawing a random sample from a Poisson distribution with an average number of users given in Table V. The users' locations are also chosen randomly. We consider a smart city application using the mmWave band as an example of MBBRLLC traffic, teleoperated driving using the microwave band as an example of RLLMTC, remote pervasive monitoring using the mmWave band as an example of MBBMTC and immersive VR video transmission using the THz band as an example of MBBRLLMTC. The channel model of each band is given in Table V. The reward obtained from a slice request of class c is selected based the required reliability, which results into a different amount of resources, and the availability of the slice resources in each type of slice (i.e., InP slice, augmented slice, or fog slice). The immediate

TABLE V
SIMULATION PARAMETERS

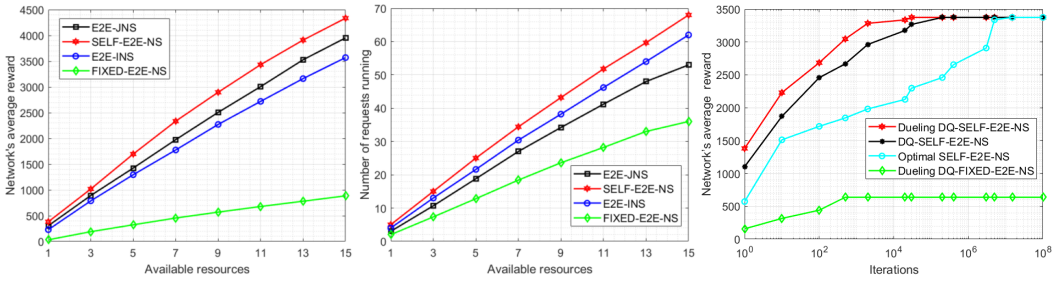
MBBMTC, MBBRLLC, RLLMTC, MBBRLLMTC	$c = 1$, $c = 2$, $c = 3$, $c = 4$
Price unused augmented slice	$\widehat{r}_{O_A}^1 = 4$, $\widehat{r}_{O_A}^2 = 9$, $\widehat{r}_{O_A}^3 = 2$, $\widehat{r}_{O_A}^4 = 6$
Price unused InP slice	$r_{O_I}^1 = 3$, $r_{O_I}^2 = 7$, $r_{O_I}^3 = 9$, $r_{O_I}^4 = 5$
Price FUEs	$\Lambda_1 = 0.13$, $\Lambda_2 = 0.26$, $\Lambda_3 = 0.0013$, $\Lambda_4 = 0.2$
Penalty	$\hbar = 1$
Mapping parameters	$\beta_1 = 0.8, \eta_1 = 0.2, \beta_2 = 0.5, \eta_2 = 0.5, \beta_3 = 0.2, \eta_3 = 0.8, \beta_4 = 0.4, \eta_4 = 0.6$
Average no. users requesting service per tenant	$N_1 = 240$, $N_2 = 120$, $N_3 = 6000$, $N_4 = 60$
Available InP resources	V_I = 265 MB/sec, $G_I = 45$ Gcycles/sec, and $B_I = 1.75$ THz
Initial InP resources	V_1 = 26.5 MB/sec, $G_1 = 4.5$ Gcycles/sec, and $B_1 = 0.175$ THz
Payload size for UE j	$K_j^1 = 250B, K_j^2 = 1kB, K_j^3 = 536B, K_j^4 = 1.25kB$
Number of cycles per bit	200
UE minimum throughput in each class	$T_{min}^1 = 1Gbps$[35], $T_{min}^2 = 4Gbps$[37], $T_{min}^3 = 40Mbps$[1], $T_{min}^4 = 2Gbps$[36]
UE maximum delay in each class	$\tau_{max}^1 = 250msec$[35], $\tau_{max}^2 = 50msec$[37], $\tau_{max}^3 = 5msec$[1], $\tau_{max}^4 = 30msec$[36]
Reliability	$\rho^1 = 99\%$[35], $\rho^2 = 99.99\%$[37], $\rho^3 = 99.999\%$[1], $\rho^4 = 99.9\%$[36]
Path loss for MBBRLLMTC service tx using THz band	$(\frac{4\pi fr dist.}{speed\ of\ light})^2\, e^{-k(fr)dist.}$ $k(fr) = \{0.0002, 0.0016\}m^{-1}$, $fr = \{0.2, 1\}$THz, W=10 GHz [30]
Path loss (dB) for MBBMTC and MBBRLLC services tx using mmWave band	$\vartheta + 10\beta log_{10}(dist.) + X_\iota$ LOS:$\vartheta = 53, \beta = 1.43, \iota = 2.24$ NLOS: $\vartheta = 29.086, \beta = 2.03, \iota = 3.04$, fr = 60 GHz, W=2.16 GHz [31]
Path loss (dB) RLLMTC service tx on microwave band	$128.1 + 37.6\log_{10}(dist.)$ [32]
Transmit power/Noise power	24 dBm / -174 dBm/Hz
Learning rate	0.01
Size of hidden layers	64
Reply buffer max. capacity	10,000
Critic/actor hidden layers	size 512, 256, 128 / 256, 64

reward from accepting a slice from class c can be calculated as follows

$$r_{RT}^c = \varphi_{RT}^c (b^c cost_b + g^c cost_g + v^c cost_v) \tag{27}$$

where φ_{RT}^c is the priority weight for a slice from class c on each type of resource (RT)- InP slice (φ_I^c), augmented slice (φ_A^c), or fog slice (φ_F^c). $cost_b$, $cost_g$, and $cost_v$ are the cost per radio unit (\$/Hz), the cost per computing unit (\$/(cyc/sec)), and the cost per storage unit (\$/(bit/sec)); respectively.

Each UE $j \in \mathcal{J}^c$ is allocated a bandwidth W_j^c such that the overall bandwidth allocated to a slice of class c is $W^c \geq \sum_{j=1}^{|\mathcal{J}^c|} W_j^c$. The bandwidth is not shared (reused) between users to avoid interference, and we assume users have several orthogonal channels available. The service requirements for each class are described in Table V for uplink transmission since both throughput and delay are more restrictive in this case. The transmission rate of UE $j, \forall j \in \mathcal{J}^c$, $c =$ MBBMTC, MBBRLLC, or MBBRLLMTC, to a base station (BS) i (or FUE) on channel υ in a slice from class c is calculated by Shannon's capacity $T_{j,\upsilon}^c = W_j^c \log_2\left(1 + \gamma_{ji,\upsilon}^c\right)$, where W_j^c is the bandwidth allocated to UE j and $\gamma_{ji,\upsilon}^c$ is the signal-to-noise ratio obtained as $\gamma_{ji,\upsilon}^c = q_{ji,y}^c h_{ji,\upsilon}^c / \sigma_{j,c}^2$ with transmission power $q_{ji,\upsilon}^c$ from UE j to BS i on channel υ, channel gain $h_{ji,\upsilon}^c$ between UE j and BS i on channel υ, and noise power $\sigma_{j,c}^2$ at UE j. On the other hand,



(a) (b) (c)

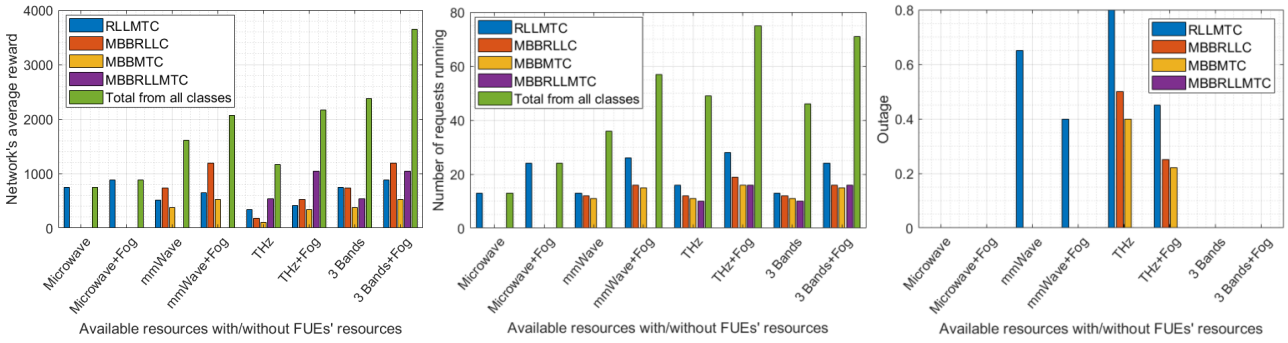
Fig. 3. (a) Network's average reward, (b) Total number of slice requests accepted in the network vs. available InP's resources (B_I = xB_1,G_I = x G_1, V_I = x V_1) when the percentage of unused slices is 20%, the number of MBBMTC FUEs, MBBRLLC FUEs, RLLMTC FUEs, and MBBRLLMTC FUEs in first seller tenant is 50, 20, 1000 and 30, respectively, and in the second seller tenant are 35, 15, 750 and 25, respectively, and (c) Network's average reward vs. no. episodes.

the data rate of UE j, $\forall j \in \mathcal{J}^c$, c = RLLMTC, taking teleoperated driving as an example, which needs high reliability and low delay service, e.g., to transmit information between vehicles, depends on the finite blocklength capacity. As the payload length is very small, the achievable rate of UE j is obtained by the short packet rate [33] $T_j^c \leq W_j^c \left\{ \log_2\left(1+\gamma_{ji,v}^c\right) - \sqrt{\frac{Ch_{ji,v}}{\Upsilon_{ji,v}}}\, Q^{-1}(\chi)\log_2 e \right\}$, where the channel dispersion of UE j is $Ch_{ji,v} = 1 - 1/\left(1+\gamma_{ji,v}^c\right)^2$, $\gamma_{ji,v}^c = q_{ji,v}^c h_{ji,v}^c/\sigma_{j,c}^2$ is the signal-to-noise ratio defined above, $\Upsilon_{ji,v}$ is the codeword block length of UE j, χ is the transmission error probability, and $Q^{-1}(.)$ is the inverse of the Gaussian Q-function. The number of arrival tasks to FUEs follows a Poisson distribution with average arrival rate λ_{TX}^c set to 0.5 task/msec for FUEs from MBBMTC, MBBRLLC and MBBRLLMTC, and 0.2 task/msec for RLLMTC. We assume that UEs in the same class require the same amount of resources, and the duration of each time slot ζ and slice Δ is 1 msec and 1 hour, respectively [34]. The deep neural network architecture is designed with the size of the hidden layers equal 64. For DQ-E2E-JNS, we use two fully-connected hidden layers. For Dueling DQ-E2E-JNS scheme, the structure of the neural network contains two fully-connected hidden layers and two streams. We use ϵ-greedy algorithm with an initial and final value of ϵ equal 1 and 0.1, respectively. The random actions are selected with probability ϵ, and the actions that maximize the $Q(\boldsymbol{s},\boldsymbol{a})$ are selected with probability 1- ϵ.

B. Performance Evaluation and Comparison

Existing works [12] that consider only one UE in making a decision to use the fog or cloud to complete their tasks or only one type of service [16] cannot be directly compared to our schemes. Therefore, we compare our results using the slicing algorithm in [14] that adopts learning to accommodate heterogeneous traffic requests in a multi-tenant scenario using a fixed InP infrastructure without cooperation between InP and tenants or fog resources. We refer to the algorithm [14] as fixed-E2E-NS. Besides we also compare the performance using exclusively E2E-JNS or E2E-INS to serve all requests without self-learning orchestration. To validate the benefits of our proposed slicing mechanism, the overall average reward of InP and tenants and the overall number of requests served are compared in Fig. 3a and 3b, respectively, to the algorithms previously described. We assume that the available resources of the InP varied in the ranges of 1xV_1 to 15xV_1, 1xG_1 to 15xG_1, and 1xB_1 to 15xB_1 where the initial InP resources, (V_1, G_1, B_1), are described in Table V. The number of requests received per class equals 20 for MBBMTC (c = 1), 18 for MBBRLLC (c = 2), 17 for RLLMTC (c = 3), and 22 for MBBRLLMTC (c = 4). The average reward increases with the InP's resources as more slice requests are accepted. Our proposed slicing mechanism outperforms the E2E-JNS and E2E-INS algorithms implemented alone. The self-learning orchestration facilitates a fair allocation of slices by forwarding the requests to each scheme with the highest probability of acceptance. In contrast, each scheme separately only accepts the requests with the highest revenue. If there are limited InP resources, our scheme achieves up to 4.5 times higher reward and serves up to twice more requests than the fixed-E2E-NS [14]. But even if the InP has abundant resources, our mechanism significantly outperforms the other schemes. By leveraging the available fog resources and leasing back the unused slices to the InP, the resources are efficiently utilized to serve new demands.

In Fig. 3b, we can see that the E2E-INS scheme accepts more slice requests than the E2E-JNS scheme, but its revenue is lower. This is because the FUEs are used to augment the InP slices in E2E-JNS to serve high-reward classes, whereas, in the E2E-INS scheme, they are utilized to create fog slices to serve low-reward classes. In Table VI, we show the number of requests accepted per class for different amounts of InP's resources. As we can see, when there are a few available resources, only requests from MBBRLLC (c = 2) and RLLMTC (c = 3) are accepted by JNS and INS, respectively, since they achieve a higher long-term reward than other classes. Once all the slice requests from MBBRLLC and RLLMTC classes are accepted, the remaining resources in the JNS and INS will be allocated to serve some slice requests from MBBMTC and MBBRLLMTC, respectively. In Fig. 3c, we show the optimality and convergence of the proposed SELF-E2E-NS mechanism when the InP slice allocation policy in both E2E-JNS and E2E-INS is obtained by using Q-learning, Deep Q-learning, Dueling Deep Q-learning, and by solving the dynamic optimization by exhaustive search. The average number of users requesting MBBMTC service/tenant, MBBRLLC service/tenant, RLLMTC service/tenant, and MBBRLLMTC service/tenant are 240, 120, 6000, and 60, respectively. Additionally, the InP resources are set to V_I = 265 MB/sec, G_I = 45 Gcycles/sec, and B_I = 1.75 THz. As we can see, the proposed SELF-E2E-NS algorithm can achieve optimal performance and outperforms the fixed-E2E-NS [14] in terms of the network's average reward in just a few iterations. Dueling obtains the reward much faster than the other algorithms and is



(a) (b) (c)

Fig. 4. (a) Network's average reward, (b) Total number of slice requests accepted in the network, (c) Outage vs available bands with/without FUEs' resources when the percentage of unused slices is 20%, the number of MBBMTC FUEs, MBBRLLC FUEs, RLLMTC FUEs and MBBRLLMTC FUEs in first seller tenant is 50, 20, 1000 and 30, respectively, and in the second seller tenant are 35, 15, 750 and 25, respectively.

TABLE VI
THE NUMBER OF SLICES ACCEPTED FROM EACH CLASS

Available resources	$c=2$	$c=3$	$c=4$	$c=1$
B_1, G_1, V_1	3	2	0	0
$3*B_1, 3*G_1, 3*V_1$	8	6	0	0
$5*B_1, 5*G_1, 5*V_1$	14	11	0	0
$7*B_1, 7*G_1, 7*V_1$	18	17	3	3
$9*B_1, 9*G_1, 9*V_1$	18	17	7	6
$11*B_1, 11*G_1, 11*V_1$	18	17	10	9
$13*B_1, 13*G_1, 13*V_1$	18	17	13	12
$15*B_1, 15*G_1, 15*V_1$	18	17	17	15

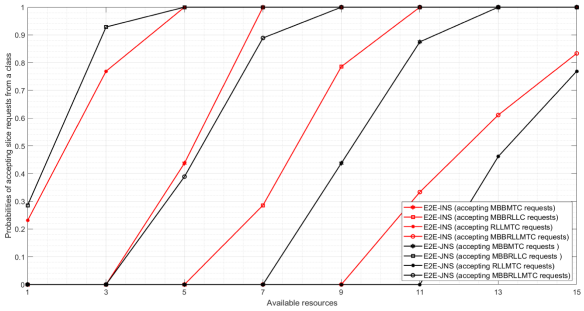


Fig. 5. Probabilities of accepting slices vs available InP's resources.

Fig. 6. Seller tenants' average reward vs episode.

more suitable for dealing with dynamic requests and the availability of FUEs. Note that the fixed-E2E-NS [14] converges faster than our proposed scheme since the state-action space of the proposed algorithm is larger than in the fixed-E2E-NS [14]. By examining the training loss, Dueling DQ-SELF-E2E-NS leads to lower and more stable loss. The converged loss values achieved in DQ-SELF-E2E-NS and the Dueling DQ-SELF-E2E-NS are around 7 and 3, respectively.

In Fig. 4, we present the reward, number of requests, and outage for different available bands and with and without FUEs' resources. When we have only a microwave band in the network with a number of available resources $B_I = 11$ GHz, $G_I = 45$ Gcycles/sec and $V_I = 265$ MB/sec, the RLLMTC slice requests are accepted, while requests from other classes are rejected. This is because there are insufficient radio resources to serve the other classes. The network's average reward increases by using the resources of the fog UEs. Since more resources are available, this leads to accepting more slice requests. If we increase the communication resources by incorporating mmWave, there are enough resources to accept requests from other classes, i.e., MBBRLLC and MBBMTC. Furthermore, if THz band is available, we can accept slices from all classes. However, the long-distance detrimental atmospheric effects in these bands lead to violating the SLA of the requests running. This is especially relevant for RLLMTC since it has the highest reliability requirement and thus achieves a lower average reward. In fact, although the number of slice requests accepted using the mmWave and THz bands increases, the outage of RLLMTC, MBBRLLC, and MBBMTC services also increases. This can be improved by utilizing fog UEs in these bands since the transmission distance becomes shorter, and thus SLA degradation decreases. When three bands are available in the network, each class will be accepted by the most suitable band, avoiding the outage and achieving a higher network average reward. Even though the number of requests running with solely THz band is comparable to the three bands, the performance of the three bands in terms of the network's average reward and the outage are better, as previously explained.

The probability of accepting a slice request for each class by the E2E-JNS and E2E-INS depends on the network condition and expected future demands. In Fig. 5, we illustrate the probabilities that each scheme accepts a slice request from each class given their immediate rewards and priorities described in Table III. When the InP has a few resources, the E2E-INS scheme accepts requests from RLLMTC and MBBMTC by using InP slices and fog slices, respectively, and rejects requests from MBBRLLC and MBBRLLMTC. In particular, the different classes can be ordered in terms of their achieved long-term reward using InP slices as: RLLMTC $>$ MBBRLLC $>$ MBBRLLMTC $>$ MBBMTC. Likewise, the order of acceptance based on the achieved long-term reward using fog slices is: MBBMTC $>$ MBBRLLMTC $>$ MBBRLLC $>$ RLLMTC. On the other hand, the E2E-JNS scheme accepts slice requests from MBBRLLC and MBBRLLMTC as they achieve the highest long-term reward and rejects requests

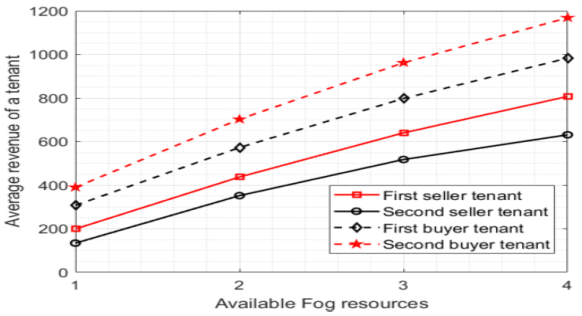


Fig. 7. Tenants' revenue vs FUE resources.

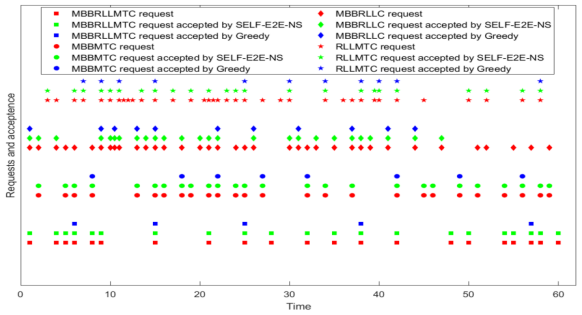


Fig. 8. Requests and accepted slices per class vs time.

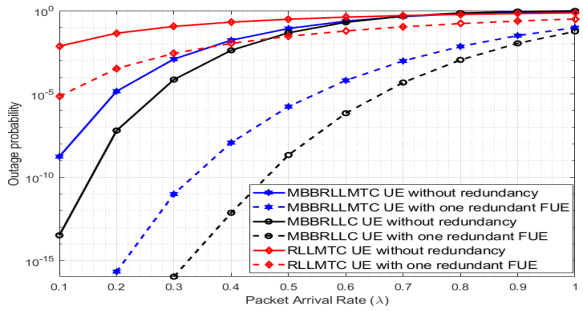


Fig. 9. Outage probability of our proposed schemes with and without redundancy versus the packet arrival rate of FUEs in packets/msec.

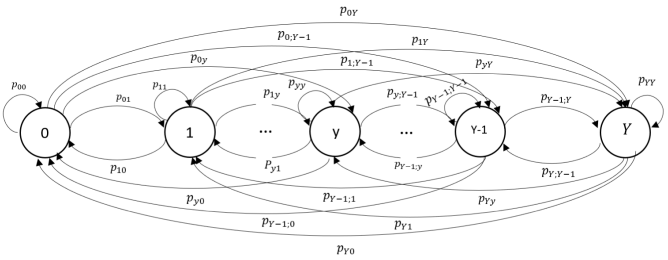


Fig. 10. Markov resource availability model.

from MBBMTC and RLLMTC. The order of acceptance using augmented slices is: MBBRLLC $>$ MBBRLLMTC $>$ MBBMTC $>$ RLLMTC. As the available InP resources increase, we can observe that the probability of accepting slice requests from other classes previously rejected increases in both schemes, following the order described above. By changing the reward value of each class and type of slice resource, we can change the acceptance rate per class and slicing mode. Therefore, based on the traffic demands of each class, the reward can be dynamically adjusted to accept different classes at different times and balance the number of slices served per class. The convergence of the MAAC-E2E-INS algorithm is shown in Fig. 6. We consider two seller and two buyer tenants. The number of idle subscribers of class MBBMTC, MBBRLLC, RLLMTC, and MBBRLLMTC in the first (second) seller tenant is 50 (35), 20 (15), 1000 (750), and 30 (25), respectively. The first (second) buyer tenant requests 6 (4) MBBMTC slices, 5 (2) MBBRLLC slices, 7 (4) RLLMTC slices, and 8 (5) MBBRLLMTC slices. We can see that MAAC-E2E-INS converges to the maximum reward in 200 episodes. In Fig. 7, we examine the impact of increasing the number of available FUEs by 2x, 3x, and 4x the previous number of idle subscribers on the tenants' revenue. The revenue of the seller tenants and buyer tenants increases with the number of available FUEs. The seller tenants can serve more slice requests by utilizing available fog slices, increasing their revenue. Simultaneously, the revenue of the buyer tenants increases because of serving more end users. We assume the first (second) buyer has 9 (12), 5 (8), 8 (9), and 10 (15) slice requests of MBBMTC, MBBRLLC, RLLMTC, and MBBRLLMTC, respectively. As can be observed, the seller who has more resources serves the buyer who has more slice requests, and the opposite is true. Therefore, the first seller serves the second buyer (red lines), and the second seller serves the first buyer (black lines).

In Fig. 8, we compare the performance of the SELF-E2E-NS mechanism with a greedy management algorithm Greedy-E2E-NS regarding the number of slice requests accepted from each class. A red shape means a slice request arrival of any class, a green shape means an accepted slice request by SELF-E2E-NS, and a blue shape means an accepted slice request by the Greedy-E2E-NS when the available InP resources (G_I, B_I, V_I) in the network are as mentioned in Table V. The Greedy-E2E-NS forwards the slice requests randomly to any scheme and ignores previous acceptance/rejection responses and the available resources in the network. As can be seen, our proposed algorithm accepts more slice requests than the Greedy-E2E-NS because the global orchestrator selects E2E-JNS or E2E-INS based on a risk model that minimizes the probability of not serving the requests and achieving the SLA requirement. The risk model takes into account the historical data that is based on the belief of available resources and the probability of achieving the SLA in each mode.

In Fig. 9, we examine the outage probability of UE from each class in our proposed algorithm due to the uncertainty of FUEs' resources in the augmented slices of the E2E-JNS and the fog slices of the E2E-INS. The probability of outage in our proposed scheme is plotted versus the packet arrival rate of FUEs λ_{TX} with different amounts of redundancy. As shown in Fig. 9, the outage probability of RLLMTC UE without redundancy is higher than the UEs in other classes. This is attributed to the high required reliability of RLLMTC UEs. On the other hand, the outage probability of MBBRLLC UE is very low without adding redundancy. MBBMTC obtained a similar performance as the latter and is omitted for clarity of presentation. Therefore, to reduce the outage probability the augmented slices must be boosted by a number of redundant FUEs to overcome the uncertainty of FUEs' resources. For $\lambda_{TX} = 0.3$, we can see that by considering one redundant FUE per UE's connection, the outage probability decreases by 2 orders of magnitude in RLLMTC, by 8 in MBBRLLMTC, and by 10 orders of magnitude in MBBRLLC. More details on how redundancy strategies can increase the robustness of fog networks are discussed in our previous work [5].

VII. Conclusion

In this paper, we have presented a self-learning slicing mechanism with two operational modes, E2E-JNS and E2E-INS, to perform end-to-end network slicing in dynamic multi-tier networks. The collaboration between InP and tenants facilitates sharing fog resources and reusing available slices to meet future demands. Our mechanism adapts to the varying availability of resources by relying on feedback from InP and tenants in terms of admission and satisfaction with the service, respectively. By learning from previous decisions, our scheme adapts its activation mode to the network conditions and minimizes the risk of non-satisfying the service request. To solve the optimal admission and collaboration policy, DQ-E2E-JNS and MAAC-E2E-INS algorithms are proposed. In the former, the InP serves the demand using augmented slices, and the admission policy is obtained by optimizing the number of fog nodes and unused slices. In the latter independent slices are created, and the InP delegates part of the service provisioning to tenants' fog slices. The simulation results show that the proposed approach significantly improves the revenue and number of requests served over fix slicing, exploits multiple spectrum bands, performs fair slice allocations, and brings substantial gains to InP and tenants.

Appendix

A. Complexity of Tenants' Slice Request/Offer Optimization

The complexity of the algorithm to solve (3) is $\mathcal{O}\left(CJ^x_{c'}m^x_{max,c'}\log(J^x_{c'}) + m^x_{max,1}{}^{\prod_{c'=2}^{C} m^x_{max,c'}}\right)$, where the first term is the complexity of ordering $J^x_{c'}$ subscribers and grouping them into $m^x_{max,c'}$ slices per class c' and the second term is the complexity of calculating the number of slice request per class c'.

B. Computational Complexity: MAAC-E2E-INS

Let $\Uparrow_k$ denote the number of neurons of the kth layer and κ^{actor} and κ^{critic} denote the number of layers in the actor and critic networks, respectively. The computational complexity of the kth layer is $\mathcal{O}(\Uparrow_{k-1}\Uparrow_k + \Uparrow_k\Uparrow_{k+1})$. During the execution phase, only the actor network is used, and thus its computational complexity is $\mathcal{O}(\sum_{k=2}^{\kappa^{\text{actor}}-1}(\Uparrow^{actor}_{k-1}\Uparrow^{actor}_k + \Uparrow^{actor}_k\Uparrow^{actor}_{k+1}))$. Both actor and critic networks are used in the training phase, and the overall computational complexity of training phase is $\mathcal{O}(\sum_{k=2}^{\kappa^{\text{actor}}-1}(\Uparrow^{actor}_{k-1}\Uparrow^{actor}_k + \Uparrow^{actor}_k\Uparrow^{actor}_{k+1}) + \sum_{k=2}^{\kappa^{\text{critic}}-1}(\Uparrow^{critic}_{k-1}\Uparrow^{critic}_k + \Uparrow^{critic}_k\Uparrow^{critic}_{k+1}))$.

C. Belief InP Resource Availability

We model the dynamics of InP resource availability using the Markov chain shown in Fig. 10. For simplicity, we assume the InP has Y resources independently of the mode. In state y, the InP has y resources allocated and $Y - y$ resources available to allocate to new requests. Every slice departure (arrival) of class c moves the Markov process to the left (right) and releases (occupies) a number of resources. The probability that m resources will be released is

$$P^+_m = \sum_{\hat{m}\in\mathcal{M}_y} \varphi_{\hat{m}} \prod_c P^{\mu^c}_{m^c_y} \tag{28}$$

where $\varphi_{\hat{m}} = 1$ if $\sum_c m^c_y\varpi^c = m$, otherwise $\varphi_{\hat{m}} = 0$, $P^{\mu^c}_{m^c_y}$ is the probability that there are m^y_c slice departures from class c at state y, ϖ^c is the amount of resources per request of class c, μ^c is the departure rate of slices of class c, and $\mathcal{M}_y$ is set of all combinations of slice departures at state y.

Similarly, the probability that n resources will be occupied as result of n^c_y request arrivals from class c at state y is

$$P^-_n = \sum_{\hat{n}\in\mathcal{N}_y} \varphi_{\hat{n}} \prod_c P^{\lambda^c}_{n^c_y} \tag{29}$$

where $\varphi_{\hat{n}} = 1$ if $\sum_c n^c_y\varpi^c = n$, otherwise $\varphi_{\hat{n}} = 0$, $P^{\lambda^c}_{n^c_y}$ is the probability of n^c_y request arrivals with arrival rate λ^c, and $\mathcal{N}_y$ is set of all combinations of slice requests of class c. Finally, the transition probability from state y to a state $y + \varrho$ is

$$P_{y,y+\varrho} = \sum_{\varrho\in S_Y} \varphi_\varrho P^+_m P^-_n \tag{30}$$

where $\varphi_\varrho = 1$ if $m - n = \varrho;\ \varrho^- < Y - y;\ \varrho^+ < y$, otherwise $\varphi_\varrho = 0$. S_Y is the set of states of available InP resources, P^+_m is the probability of releasing m resources and P^-_n is the probability of occupying n resources. The steady-state probability $\mathbf{p} = \mathbf{P}\,\mathbf{p}$, with $\mathbf{p} = [\,p(y)]$, that the system is in state y is obtained as $p(y) = p_{0y}p(0) + p_{1y}p(1) + \cdots + p_{Yy}p(Y)$, where $\sum_y p(y) = 1$, Y is the InP available resources, and $p = p_{belief}$.

References

[1] R. Xiong, C. Zhang, H. Zeng, X. Yi, L. Li, and P. Wang, "Reducing power consumption for autonomous ground vehicles via resource allocation based on road segmentation in V2X-MEC with resource constraints," *IEEE Trans. Veh. Technol.*, vol. 71, no. 6, pp. 6397–6409, Jun. 2022, doi: 10.1109/TVT.2022.3161641.

[2] N. Saeed, M. H. Loukil, H. Sarieddeen, T. Y. Al-Naffouri, and M.-S. Alouini, "Body-centric terahertz networks: Prospects and challenges," *IEEE Trans. Mol., Biol. Multi-Scale Commun.*, vol. 8, no. 3, pp. 138–157, Sep. 2022, doi: 10.1109/TMBMC.2021.3135198.

[3] L. Zhong et al., "A multi-user cost-efficient crowd-assisted VR content delivery solution in 5G-and-beyond heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 8, pp. 4405–4421, Aug. 2023, doi: 10.1109/TMC.2022.3162147.

[4] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021.

[5] B. Lorenzo, F. J. González-Castaño, L. Guo, F. Gil-Castiñeira, and Y. Fang, "Autonomous robustness control for fog reinforcement in dynamic wireless networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2522–2535, Dec. 2021.

[6] I. Kovacevic, A. S. Shafigh, S. Glisic, B. Lorenzo, and E. Hossain, "Multi-domain network slicing with latency equalization," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2182–2196, Dec. 2020.

[7] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghoumid, "A comprehensive survey on the E2E 5G network slicing model," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 49–62, Mar. 2021.

[8] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9517–9529, Oct. 2020.

[9] X. Yu, F. Xu, J. Cai, X.-Y. Dang, and K. Wang, "Computation efficiency optimization for millimeter-wave mobile edge computing networks with NOMA," *IEEE Trans. Mobile Comput.*, vol. 22, no. 8, pp. 4578–4593, Aug. 2023, doi: 10.1109/TMC.2022.3164974.

[10] J. García-Rois, B. Lorenzo, F. J. González-Castaño, F. Gil-Castiñeira, and J. Wu, "Slice allocation and pricing framework for virtualized millimeter wave cellular networks," *IEEE Access*, vol. 7, pp. 86349–86366, 2019, doi: 10.1109/ACCESS.2019.2923125.

[11] Y. Xiao and M. Krunz, "Dynamic network slicing for scalable fog computing systems with energy harvesting," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2640–2654, Dec. 2018.

[12] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[13] Y. Sun, M. Peng, S. Mao, and S. Yan, "Hierarchical radio resource allocation for network slicing in fog radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3866–3881, Apr. 2019.
[14] N. Van Huynh, D. Thai Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1455–1470, Jun. 2019.
[15] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7863–7878, Jul. 2020.
[16] H.-T. Chien, Y.-D. Lin, C.-L. Lai, and C.-T. Wang, "End-to-end slicing with optimized communication and computing resource allocation in multi-tenant 5G systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2079–2091, Feb. 2020.
[17] M. R. Raza, M. Fiorani, A. Rostami, P. Öhlen, L. Wosinska, and P. Monti, "Dynamic slicing approach for multi-tenant 5G transport networks [invited]," *J. Opt. Commun. Netw.*, vol. 10, no. 1, pp. A77–A90, Jan. 2018.
[18] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE Conf. Comput. Commun.*, Atlanta, GA, USA, May 2017, pp. 1–9.
[19] H. H. Esmat and B. Lorenzo, "Deep reinforcement learning based dynamic edge/fog network slicing," in *Proc. IEEE GLOBECOM*, Taipei, Taiwan, Dec. 2020, pp. 1–6.
[20] M. Rasti, S. K. Taskou, H. Tabassum, and E. Hossain, "Evolution toward 6G wireless networks: A resource management perspective," 2021, *arXiv:2108.06527*.
[21] 3GPP. *Radio Access Network*. Accessed: Mar. 2019. [Online]. Available: http://www.3gpp.org/specifications-groups/ran-plenary
[22] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing; A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, 2015.
[23] A. Ksentini and P. A. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5G," *IEEE Netw.*, vol. 34, no. 2, pp. 99–105, Mar. 2020.
[24] W. Xia, J. Zhang, T. Q. S. Quek, S. Jin, and H. Zhu, "Power minimization-based joint task scheduling and resource allocation in downlink C-RAN," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7268–7280, Nov. 2018.
[25] Y. Ren, A. Guo, C. Song, and Y. Xing, "Dynamic resource allocation scheme and deep deterministic policy gradient-based mobile edge computing slices system," *IEEE Access*, vol. 9, pp. 86062–86073, 2021.
[26] B. Lorenzo, A. S. Shafigh, J. Liu, F. J. González-Castaño, and Y. Fang, "Data and spectrum trading policies in a trusted cognitive dynamic network architecture," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1502–1516, Jun. 2018.
[27] R. G. Gallager, *Discrete Stochastic Processes*. London, U.K.: Kluwer Academic, 1995.
[28] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2016, *arXiv:1511.06581*.
[29] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Info. Proc. Syst.*, 2017, pp. 6379–6390.
[30] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low-latency communications for wireless VR?" *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9712–9729, Jun. 2022, doi: 10.1109/JIOT.2022.3142674.
[31] B. Chang, W. Tang, X. Yan, X. Tong, and Z. Chen, "Integrated scheduling of sensing, communication, and control for mmWave/THz communications in cellular connected UAV networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 7, pp. 2103–2113, Jul. 2022, doi: 10.1109/JSAC.2022.3157366.
[32] G. Ding, J. Yuan, G. Yu, and Y. Jiang, "Two-timescale resource management for ultrareliable and low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3282–3294, May 2022, doi: 10.1109/TCOMM.2022.3162366.
[33] Y. Zhao, X. Chi, L. Qian, Y. Zhu, and F. Hou, "Resource allocation and slicing puncture in cellular networks with eMBB and URLLC terminals coexistence," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18431–18444, Oct. 2022, doi: 10.1109/JIOT.2022.3160647.
[34] H. Zhang and V. W. S. Wong, "A two-timescale approach for network slicing in C-RAN," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6656–6669, Jun. 2020.
[35] G. Cisotto, E. Casarin, and S. Tomasin, "Requirements and enablers of advanced healthcare services over future cellular systems," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 76–81, Mar. 2020.
[36] C. Chaccour, R. Amer, B. Zhou, and W. Saad, "On the reliability of wireless virtual reality at terahertz (THz) frequencies," in *Proc. 10th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jun. 2019, pp. 1–5.
[37] D. Panno and S. Riolo, "A new centralized access control scheme for D2D-enabled mmWave networks," *IEEE Access*, vol. 7, pp. 80697–80716, 2019.

Haitham H. Esmat (Member, IEEE) received the B.S. and M.S. degrees from Helwan University, Cairo, Egypt, in 2011 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Massachusetts Amherst, Amherst, MA, USA. His current research interests include B5G and 6G mobile communication technologies, device-to-device communications, the Internet of Things, dynamic network slicing, satellite-terrestrial edge computing networks, and AI-based applications in wireless communication systems.

Beatriz Lorenzo (Senior Member, IEEE) received the M.Sc. degree in telecommunication engineering from the University of Vigo, Vigo, Spain, in 2008, and the Ph.D. degree from the University of Oulu, Oulu, Finland, in 2012. She is currently an Assistant Professor and the Director of the Network Science Laboratory, Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA. She has published more than 50 articles and coauthored two books on advanced wireless networks. The latest book *Artificial Intelligence and Quantum Computing for Advanced Wireless Networks* (Wiley, 2022), covers the enabling technologies for the definition, design, and analysis of incoming 6G/7G systems. Her research interests include AI for wireless networks, B5G and 6G network architectures and protocol design, mobile computing, optimization, and network economics. She was a recipient of the Fulbright Visiting Scholar Fellowship with the University of Florida from 2016 to 2017. She was the General Co-Chair of WiMob Conference in 2019. She serves regularly in the TPC for top IEEE/ACM conferences. She is also an Editor of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.