Towards Enabling Network Slice Mobility to Support 6G System

Miloud Bagaa[1], Diego Leonel Cadette Dutra[2], Tarik Taleb[1,3], and Hannu Flinck[4]
[1] Dep. of Communications and Networking School of Electrical Engineering, Aalto University, Espoo, Finland
[2] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[3] University of Oulu, 90570 Oulu, Finland.
[4] Nokia Standards & Technology, Helsinki, Finland.
Emails:{firstname.lastname}@aalto.fi; diegodutra@lcp.coppe.ufrj.br; hannu.flinck@nokia-bell-labs.com

***Abstract*—Even a wider set of highly critical and latency-sensitive applications with resource needs from the access network and the edge will be supported by the 6G networks. Therefore, the 6G network will deal with diversification of service platforms. Optimizing the resource consumption of network slicing on top of a shared infrastructure will become essential to keep the operating costs on an acceptable level. Each vertical, e.g., eMBBPlus, BigCom, holographic and tactile communications can run on top of network slice with specific KPIs. Different verticals can have contradicting requirements running on top of the same infrastructure. This paper investigates the orchestration of network services within a federated end-to-end network slice, which may span over multiple cloud domains as expected to be a common scenario in 6G deployments. We introduce three optimization solutions that consider two conflicting objectives, the end-to-end delay and service relocation, for orchestrating network slice. While the first solution optimizes the end-to-end delay, the second solution optimizes the service relocation. Meanwhile, the third solution leverages the bargaining game theory for achieving optimal Pareto fair trade-off configuration to optimize both objectives. The simulation results demonstrate the efficiency of the proposed solutions to achieve their main design goals**

***Index Terms*—Multi-Access Edge Computing, 6G, Network function virtualization, Network slicing**

I. Introduction

Network slicing was first introduced in 5G networks to allow network service customization by isolating resources and to support different and often times contradicting Quality of Service (OoS) needs. The main enabler for slicing is network function (NF) virtualization in combination with the efficient Network functions virtualization Orchestration (NFVO) coordinates the resources usage. in 6G the service diversification will continue as more and more service, e.g., sensing or Augmented Reality services, will be provided from edge clouds. In 6G we anticipate that NFs will be cloud native and they will be distributed across a heterogeneous fabric of clouds possibly under different administrative domains; central core cloud, multiple edge and far-edge clouds based on the service requirements. As a result the resource optimization problem needs to take into account different cloud capabilities. Network slicing enables the sharing of network resources, i.e., computing, networking, memory, and storage, of a physical infrastructure among various virtual tenants while ensuring their logical or physical isolation [1]–[3]. Henceforth, in the scope of this paper, a network slice is defined as a set of Virtual Network Functions (VNFs) instantiated and running on top of virtual environments and are interconnected using virtual network technologies [4].

Network slicing enables the deployment of complex system functionalities, allowing operators to support different use cases with contradicting requirements. The Ultra-Reliable Low latency Communications (URLLC) use case requires both ultra-low latency and very high reliability will demand a 6G-ready network slicing that leverages on the Multi-Access Edge Cloud (MEC) [5] concept. MEC brings computational resources from the traditional datacenters to the edge of the mobile network operators (MNO) enabling, stronger service delivery to the end-users and offering an ultra-short delay to near customers. By combining these emerging concepts, academia and industry developed several use cases, i.e., Augmented Reality, Autonomous Car's Communications, Smart Manufacturing, Connected Drones, and Smart Cities.

These use cases assume a high-performance and low latency machine learning solution tailor-made to the application exists, which is feasible using the computational resources of a MEC. Each instantiated slice must ensure service continuity to its end-users, bounded end-to-end network latency, and a suitable overall cost for the network slices. A service relocation would solve the mobility issue, yet this solution is complicated or even unfeasible for stateful services. Moreover, a slice could be full, underused, or empty, in which different actions, such as scaling up/down, can be applied autonomously for meeting the desired key performance indicators (KPIs) while reducing the cost [6].

This paper suggests a network orchestration system to reduce the end-to-end delay and increase network bandwidth while preventing service disruption. The end-to-end delay and high bandwidth are ensured by assigning the right amount of resources to the VNFs and placing them in the right place close to the end user. The right amount of resources helps the VNFs handle user equipments (UEs) traffic quickly (i.e., processing time) while reducing the overall cost. Meanwhile, placing the VNFs close to the UEs helps to reduce the network delay (i.e., propagation delay). Various service relocation techniques are employed to assign the correct quantity of resources to the VNFs and place them at the right places, including VNF scaling up/down, VNF creation/deletion, or VNF migration. Unfortunately, the service relocation comes with an avoidable overhead on service availability. In fact, by relocating services, the ongoing connections with the UEs can be interrupted.

We introduce a network slice configuration solver that uses optimization solutions as an initial step followed by a theoretical game theory approach to guarantee and enforce a reduced end-to-end latency for a service while maintaining an assured system bandwidth, and reducing service relocation for the various Network Slice Mobility (NSM) patterns. We are particularly interested in providing and managing both intra-cloud and inter-cloud domain network services and how to provide network coverage tailored to a slice's requirements, where each cloud has a well-defined set of instance flavors. Each one of these flavors defines the amount of computational, storage, memory, and network resources available to the owner of an instance of that given flavor. Moreover, these flavors usually are discrete points in the configuration space, e.g., 6 vCPUs, 12 GiB, 200 GiB storage, and 50 Mbps. We have suggested three different solutions to orchestrate the network slice while dealing with the two conflicting objectives: end-to-end delay and service relocation. The ultimate goal is to relocate the services as minimum as possible while reducing the end-to-end delay. The UEs should be optimally assigned to different services and scaling up/down those services without harming the UE's Quality of experience (QoE). We expect that the multi-administrative domains supporting different cloud flavors will be a key characteristic of the forthcoming 6G network architecture [7]. We can reduce the overall deployment cost of a federated network slices by leveraging on multi-administrative and technological domain network slices consisting of an optimally positioned minimal number of high-quality VNFs at the lowest costs. This further reduces the deployment and maintenance costs of federated network slices in terms of their Capital Expenditures (CAPEX) and Operating Expenditures (OPEX) while also reducing the time needed to plan, deploy and maintain network slices.

In the first objective, we aim to reduce the end-to-end delay and maximize the use of local bandwidth by placing services (i.e., VNFs) close to the users. In this regard, we have suggested the first solution, named delay-aware network service management (DELAY-NSM), that aims to reduce the end-to-end delay and increase the bandwidth. When the users move from a location to another, this solution aims to reduce the end-to-end delay by relocating services. Unfortunately, the service relocation comes with an avoidable overhead on service interruption. DELAY-NSM solution aims to reduce the end-to-end delay while considering the service relocation as a constraint. It aims to reduce the end-to-end delay while service relocation (i.e., interruption) does not exceed a specific, service-defined, threshold.

Meanwhile, the second solution, resource relocation aware network service management (R-NSM), favorites the second objective, service relocation. It aims to prevent service relocation by keeping the services at the same location with the same configuration as much as possible. However, this strategy harms the end-to-end delay due to UE mobility. This solution aims to minimize the service relocation as much as possible while considering the end-to-end delay as a constraint. To achieve a Pareto optimal configuration that simultaneously satisfies both objectives, we have suggested FT-NSM leverages the bargaining game. FT-NSM solution aims to optimize both goals and find a fair trade-off between them by leveraging the bargaining game – a type of cooperative game – to optimize and find a fair trade-off between them. In this solution, we considered two objectives, as two players that would like to barter goods. We have used Kalai and Smorodinsky Bargaining Model (KSBS) for designing the FT-NSM solution.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes the proposed architecture, our network model, and illustrates the problem formulation. Sections IV and V describe the proposed framework and solutions, while, Section VI analyses the performance evaluations of the proposed solutions. Finally, Section VII concludes the paper.

II. Related Work

Lin et al. [8] proposed a theoretical model that analyzes the impact of bandwidth on the total time and downtime of live virtual machine (VM) migration. They devised an initial deterministic model and extending it to handle pages' dirtying frequency. Lee et al. [9], studied the problem of scheduling low-priority tasks in a system with most of the resources already assigned to high-priority ones. They employed a Markov Decision Process (MDP) to an online scheduling policy archiving an optimal bandwidth-latency trade-off for parallel low-priority tasks migration. Gao and Rouskas [10] worked on the network configuration problem using the links and virtual nodes to ensure network load balancing. Their proposed model consists of two steps, the first step uses integer linear programming (ILP) for selecting the virtual nodes, while the second uses a Markov chain for selecting new substrate nodes for migrating virtual nodes.

The VNF placement problem is a classic optimization problem widely investigated in the literature. Piao and Yan [11] propose an approach for virtual machine placement and its migration while considering the time data transfers. Meanwhile, the authors in [12] have focused on how to map the VMs to bare-metal servers while reducing resource utilization in terms of CPU and memory. Somani et al. [13] have suggested a solution that considers performance isolation in terms of CPU memory and network. This solution considers contention properties, collocated VM's, and VM behavior when placing these VMs. The suggested algorithm considers maximizing the benefit and resource utilization under different service level agreements (SLA) [14].

Taleb and Ksentini [15] have proposed an algorithm for placing mobility-anchor gateways (i.e. SGW) that minimizes the SGW relocation occurrences. Their solution finds a trade-off between the minimization of UE handoff and the number of SGWs to be deployed. Meanwhile, the authors, in [16], aim to place virtual PGWs in the appropriate locations. The proposed solution also optimizes the association of UEs to different virtual PGWS. Bagaa et al. [17], leverage the coalition formation

Table I: Summary of Notations.

Notation	Description	Notation	Description
$\mathcal{S}$	A specified slice to would be created in the network.	r_{γ}	The current round.
$r_{\gamma-1}$	The previous round.	Γ	The technological and administrative cloud domains.
λ_i^{φ}	The required resources $\varphi \in \Omega$ to handle the traffic of UE $i \in \mathcal{N}$.	$\mathcal{L}_{\delta}$	The set of flavors can be used by the instances in the cloud $\delta \in \Gamma$.
Λ_{ℓ}^{φ}	The amount of resources $\varphi \in \Omega$ offered by the flavor $\ell \in \mathcal{L}_{\delta}$ in the cloud $\delta \in \Gamma$.	$\mathcal{F}_{\delta,\varphi}$	The amount of resources $\varphi \in \Omega$ available in the cloud $\delta \in \Gamma$.
$\mathcal{V}_1$	The set of VNF instances that are already running during the period r_{γ}. Formally, $\mathcal{V}_1$ can be defined as follows: $\mathcal{V}_1 = \bigcup\limits_{\delta \in \Gamma} \hat{\eta}(\delta)$.	$\mathcal{V}_2$	The set of VNFs that can be created during the period r_{γ}. Where, each VNF should handle at least one UE, the cardinality of $\mathcal{V}_2$ should be lower or equal to the cardinality of UEs $\mathcal{N}$. Formally, $\lvert\mathcal{V}_2\rvert \leq \lvert\mathcal{N}\rvert$.
$\beta^{j}_{\delta_1,\delta_2}$	The cost of intercloud network, in terms of end-to-end delay and communication bandwidth, for migrating a VNF $j \in \mathcal{V}_1 \cup \mathcal{V}_2$ from cloud $\delta_1 \in \Gamma$ to the cloud $\delta_2 \in \Gamma$. The end-to-end delay of a VNF j is also related to its flavor ℓ.	$\hat{\mathcal{X}}$	A matrix of boolean parameters from the previous period $r_{\gamma-1}$. Use by the Optimal Solution Algorithm in the active period r_{γ}. Formally, $\forall i \in \mathcal{N}, \forall j \in \Gamma, \forall k \in \eta(j), \hat{\mathcal{X}} = 1$ iff i is served by the VNF instance k in the cloud network j in the previous round $r_{\gamma-1}$. Otherwise, $\hat{\mathcal{X}} = 0$
α	The required time for establishing routes between a UE and its assigned VNF.	$\psi_{j,\ell}$	The fee for running a VNF instance in the cloud network j using the flavor $\ell \in \mathcal{L}_j$
σ_{δ}	The cost for scaling up/down at the cloud $\delta \in \Gamma$, in terms of end-to-end delay and communication bandwidth.	$\hat{\eta}(\delta)$	The set of VNF instances that are running in the cloud network $\delta \in \Gamma$ during the previous period $r_{\gamma-1}$.
$\mathcal{R}^{t}_{i,\delta}$	The network propagation service of type t between a UE $i \in \mathcal{N}$ and a cloud network $\delta \in \Gamma$. E.g., the propagation delay between the user i and cloud δ.	Ψ	The set of network services considered by the paper including end-to-end delay, bandwidth, and network jitter.
r^{t}_{ℓ}	The computation of service t using the flavor ℓ, i.e., computation speed of a VNF using the flavor ℓ	$\Theta_{\ell,\varphi}$	The amount of resources $\varphi \in \Omega$ would be used by the flavor $\ell \in \mathcal{L}_{\delta}$ in the cloud $\delta \in \Gamma$.
$\mathcal{F}_{\text{Th}}$	The maximum service relocation threshold tolerated for each UE $i \in \mathcal{N}$ in the network.	ψ_{Th}	The maximum cost that is allowed for managing different services in the network.
ϖ_{Th}	The maximum end-to-end delay tolerated in the network.	Ω	The set of resources types, such as CPU, RAM, and Disk.
ω_{δ}	The required time to launch a VNF in the cloud $\delta \in \Gamma$.	$\Phi(\mathcal{S})$	A function that returns the requirements of the slice $\mathcal{S}$.

game technique for hosting 4G and 5G core networks over a federated cloud. They modeled the cloud networks as players in the coalition game for hosting core network instances.

Li et al. [18] proposed a model to find the optimal solution using CPLEX for the Network Function Virtualization-Resource Allocation (NFV-RA) problem, which they named Homogeneous Link Mapping (HLM). Their model aimed to minimize the overall deployment cost. While their proposal is a solution for the Service Function Chaining (SFC) resource allocation problem it overlooked the network slicing paradigm. Zheng and Cai [19] proposed an energy-aware placement for virtual machines on cloud environments. Moreover, through live migration, they reduce the overall energy consumption and the deployment cost of VMs. They evaluated their proposal using HPC-based traces, which are unable to account for the users' QoE and the impact of the migration-induced downtime.

Wen et al. [20], proposed an ILP model to solve the Network Function Consolidation (NFC) problem using a greedy heuristic. They consider a typical three-layer architecture in which they introduce the notion of the application layer as well as the VNF layer and the underlying network layer. They evaluated their proposed solution using both random and real network topologies, their results show a 32% reduction of VNFs for large-scale deployments through consolidation. However, the network slicing paradigm imposes additional hurdles to their approach as sharing of a VNF may be infeasible given the users' SLA. Hawilo et al. [21] proposed mixed-integer linear programming (MILP) optimization model for the VNF orchestration problem that takes into account the migration or re-instantiation of a VNF. As a result, they achieved minimal downtime for the VNFs while minimizing SFC's delays with their proposal.

Ahmad et al. [22] have proposed the use of a game theory-based solution to control uplink power in 5G to solve the co-channel interference problem exacerbated by the use of femtocells on these networks. Huang et al. [23] proposed a game theory solution to the interference problem in a Device-to-Device (D2D) communication scenario when the UE links utilize common resources of multiple cells and only know their transmission parameter. Poularakis et al. [24] have proposed a game theory-based solution to the placement problem on MECs that takes into account the services routing requirements. Our work shares with the aforementioned publications the use of game theory to solve an optimization problem, however, instead of focus on the statically schedule mobile operator resources, our focus is on how to find feasible destination resources that enable us to move the slice's VNFs without violating their SLAs.

We highlight that the previous works disregarded the network slicing, which allows more flexibility to the user, albeit at the cost of a more complex optimization model. This paper differs from them as it considers the network slicing orchestration, augmented with different actions (i.e., service relocation and VNFs management) to ensure the SLA while optimizing the cost. The VNF management includes the creation, migration, and resizing by either scaling up or down. We have proposed three solutions. While the first two solutions leverage the MILP model for optimizing the end-to-end delay and service relocation, respectively, our third one leans on the theory of bargaining games to produce an optimal Pareto fair trade-off configuration that optimizes both the end-to-end delay and service relocation.

III. Network model

A. System model and scope

Based on the system architecture depicted in Figure 1, we consider a system comprised of multiple technological and administrative cloud domains Γ as is anticipated for multi-cloud 6G deployments, a global orchestrator, henceforth denominated the E2E Slicer, an NFVO (Network Function Visualization Orchestrator) [4], multiple gNBs/6gNBs, a dedicated VNFM (Virtual Network Function Manager), and MA (Monitoring Agent) per cloud platform. Per cloud MAs are needed not only for supporting slice mobility in multi-cloud 6G environment, but also to implement data collection and analytics for AI/ML. The system setups consists of four layers: i) User equipment or user plane; ii) Radio Access Network (RAN) layer and transport network; iii) Physical infrastructure layer that consists of a set of clouds; iv) Virtual network function layers consisting of different network slices. Each network slice consists of VNFs connected together, forming an SFC. The aforementioned Global Orchestrator, E2E Slicer, uses the clouds' domains APIs to manage the life-cycle of the computational resource while leveraging the NFVO to manage the services running on top of those resources, using this approach abstract the specific control features of the cloud provider, focus on the behavior expected by the slice owner. A network slice orchestrated over these domains may be designed to be highly available and reliable, thereby fulfilling the characteristics of a Critical Communication (CriC) slice as described in [25].

MA's primary role is to monitor the network metrics (e.g., service availability, end-to-end delay, and bandwidth) and report them to the global orchestrator. Our framework is orthogonal to any technology, as we can employ any available infrastructure monitoring and logging tools in MA. For instance, MA to gather and report the network metrics can leverage: i) infrastructure monitoring, such as Nagios; ii) metric collection tool, such as Prometheus; iii) Centralized log monitoring system using Splunk and Elasticsearch; iv) User-friendly data visualization like Grafana. Different notations used in this paper are summarized in Table I.

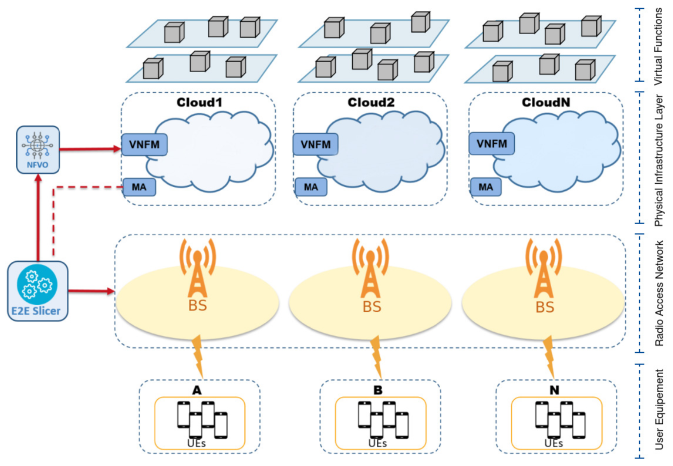
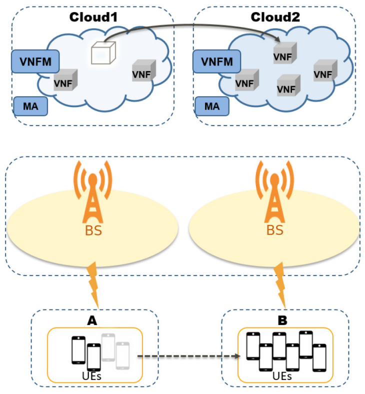
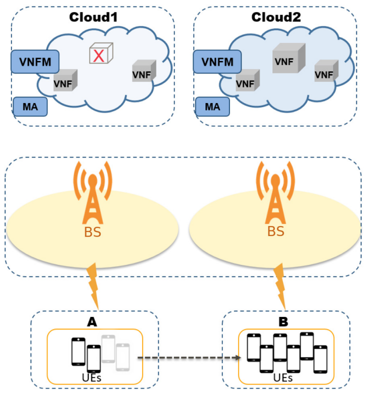


Figure 1: High-level Architecture.

The E2E Slicer orchestrated the VNFs in Figure 1 so that only one instance of a VNF of a given network slice can serve a UE, whereby each VNF has a maximum number of UEs it can concurrently handle. This limit allows the slice owner to ensure an SLA based on the slice's requirements, where this limit arises from previous profiling of a given VNF on the virtualized resource. In an unlikely event of an unforeseen shutdown of a VNF, there is a possibility to relocate UEs receiving network services from the switched-off VNF to another in the same/another cloud environment. Similarly, it is possible to migrate a VNF from one administrative domain to another. Likewise, depending on the need, a VNF can be scaled up or down according to our setup's consideration, while for slower reconfigurations instances, we can allocate or deallocate a VNF. Furthermore, to guarantee the SLA of any particular use case, every cloud platform has a dedicated MA that constantly monitors its resource consumption and its VNFs' performance and reports it to the E2E Slicer to enable its decision algorithm. Finally, the slice enforces the UEs' end-to-end latency and bandwidth, accounting for measurements of their Radio Access and Core networks.



(a) VNF Migration Scenario



(b) VNF Scaling Scenario

Figure 2: Architectures showing different VNF processing scenarios

The E2E Slicer uses the data to forecast the running context of the UE within the next epoch $r_{\gamma+1}$ based on their current states in the active epoch r_γ. E2E Slicer will compute one of three possible slice re-configurations to enforce the users' SLA: i) the relocation of UE with a cost of α from one serving

VNF to another. This cost is the time taken for the system to establish new routes between the UE and its new serving VNF. Using an SDN-like technology, we can establish the routes between different hosts (UE and its service) either using overlay/underlay networking. However, to establish these new SDN routes, a time α is required; $ii)$ the UE's VNF is migrated from one cloud $\delta_1 \in \Gamma$ to another one $\delta_2 \in \Gamma$ with a cost $\beta_{\delta_1,\delta_2}$. The latter consists of the time required for transferring UE's VNF from cloud δ_1 to δ_2. $\beta_{\delta_1,\delta_2}$ has a relation on the bandwidth and delay between those clouds; $iii)$ the UE's VNF is scaled up/down within the same cloud $\delta \in \Gamma$ with a cost σ_δ. The latter consists of the required time to scale a VNF. Furthermore, as the scope of this work lies within a single federated slice and we are interested especially in the orchestration and performance monitoring of the individual VNFs that collectively combine to provide network coverage tailored a slice, the routing of the underlying user services between the different domains are unaccounted in our proposed model.

B. Service relocation

As any service relocation harms the user QoE, we should prevent them to its utmost to avoid long-time service degradation. Henceforth, we describe different service relocation parameters that can interrupt the traffic of UEs. We denote by **VNF relocation**, whenever the VNF serving a UE changes between two consecutive epochs, this can be caused because the VNF was destroyed or relocated due to UE's mobility, arrival or departure. When a UE moves from one location to another, another VNF that ensures the required QoE and end-to-end delay (SLA) can serve it. The newly assigned VNF may already exist in the cloud, or we should deploy it to serve that UE. It is worth noting that we must establish the routing path between the clients and a newly created VNF, which can also take a considerable time. As different service relocation actions would take a specified amount of time, the E2E Slicer must consider them. We denote by ω_δ the required time to launch a VNF in the cloud $\delta \in \Gamma$. We leverage SDN technology to dynamically reestablish new routes to UEs that are currently consuming network services is possible. However, the service relocation could harm UE's ongoing sessions in terms of end-to-end delay and bandwidth of communication as, aforementioned, we assume that the time required for establishing routes between a UE and its assigned VNF is constant, and we denote it by α. Moreover, the relocation time equals α if we already assigned the UE to an existing VNF. Otherwise, if we create a new VNF, the relocation time equals to $\alpha + \omega_\delta$.

For the sake of readability we mix the physical infrastructure and virtual functions layers together in Figure 2. We denote by **VNF migration** a process whenever a VNF is migrated between various clouds as depicted in Figure 2(a). This scenario arises whenever a set of UEs currently been served by a VNF on a cloud δ_1 moves to a region best served by VNFs on cloud δ_2. At that particular epoch, if all VNFs in cloud δ_1's are unable to serve an extra user and it takes longer to instantiate a new VNF in cloud δ_2 than to migrate the currently UEs' VNF from cloud δ_1 to cloud δ_2. Moreover, the end-to-end delay and cost of communication bandwidth for migrating a VNF from cloud δ_1 to cloud δ_2 is denoted by $\beta_{\delta_1,\delta_2}$. We model the migration time as a constant $\beta_{\delta_1,\delta_2}$. This represents an upper bound of the service interruption time, as the current technologies support live-migrating both containers and VMs, with minimum downtime (service interruption). Moreover, we must note that if we had modeled this time as a random variable, as currently, we are assuming a worst-case, the performance of our solutions would only improve, albeit at the cost of solving a more complex problem.

Finally, we denote by **VNF scaling** the scenario where we change the computational resource configuration of one VNF serving a set of UEs in the region of cloud δ_2 that is executing, where the time of scaling up/down at a cloud δ_2 is denoted by σ_{δ_2}. Moreover, this scenario allows continuous operating of the VNF during the scaling operations, as advances in the Linux kernel allow it to dynamically reconfigure its computational resources for both VMs and containers, where the latter also has the advantage of a low memory and CPU footprint while delivering fast bootstrap and higher performance at the cost of shared kernel instance among all the containers. After a success migration of a service (e.g., VNFs) as depicted Figure 2(a), later, more new users can also be connected and served by this VNF. At that point, there is a choice between two options: $i)$ Creating a new VNF instance and migrate some UEs connection to the new VNF. Also, the new UEs can be oriented to the new VNF; $ii)$ Scaling the resources of the VNF in order to support the new UEs without affecting their SLA. Figure 2(b) presents the VNF scale scenario after its migration from cloud 1 to cloud 2. While the migration of a VNF from a cloud to another affects the ongoing sessions of the connected users, the service scaling should happen transparently, especially if Ceph storage is adapted.

IV. Main idea and problem formulation

A. Problem statement

In this paper, we aim to reduce the end-to-end delay and increase service availability while keeping the computational cost reasonable. To ensure the end-to-end delay and the high bandwidth, the VNFs should be assigned the right amount of resources and placed in the right place close to the end-used. We employed various service relocation techniques, e.g., VNF scaling up/down, VNF creation/deletion, or VNF migration, to reduce the cost and ensure the end-to-end delay. Unfortunately, the service relocation comes with an avoidable overhead on service availability. In fact, by relocating services, the ongoing connections with the UEs can be interrupted.

The end-to-end delay and service relocation are two conflicting objectives. The aim is to relocate the services as minimum as possible while reducing the end-to-end delay. The UEs should be optimally assigned to different services and scaling up/down those services without harming the UE's QoE. In the first objective, we aim to reduce the end-to-end delay and increase the bandwidth communication by relocating VNFs close to the users and assigning them the right amount of resources. Unfortunately, the service relocation comes with an avoidable overhead on service availability. The second objective is service availability by reducing the service relocation as much as possible. The aim is to keep the services at the same location with the same configuration as much as possible. However, this strategy harms the end-to-end delay due to UE mobility.

B. Main idea

We denote by $\mathcal{N}$ the set of UEs interested by the slice $\mathcal{S}$, where UEs may move between regions and are under the control of an $\mathcal{MA}$ that gets their locations. $\mathcal{N}$ consists of two sets ($\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$), whereby $\mathcal{N}_1$ is the set of UEs interested by the slice $\mathcal{S}$ already existing in the previous round $r_{\gamma-1}$ while $\mathcal{N}_2$ is the set of new arrival UEs. The $\mathcal{MA}$ reads users' mobility data from different access points, more precisely their handoff process, and then feeds it to the Slice Orchestrator (**SO**). We denote by Ψ the set of network requirements needed by the slice $\mathcal{S}$, e.g., end-to-end delay, bandwidth, and QoE. We denote by $\Phi_t(\mathcal{S})$ a function that returns the requirements of the slice $\mathcal{S}$ in terms of type $t \in \Psi$. We denote by $\mathcal{R}^t$ the amount of network resources t (i.e., delay or bandwidth) between the clouds and UEs. The delay and bandwidth offered by different cloud δ to UE i vary according to: i) The used underline network connecting i to δ; ii) The locations of δ and i. We denote by $\mathcal{R}^t_{i,\delta}$ the resource type t between a UE $i \in \mathcal{N}$ and a cloud $\delta \in \Gamma$. The proposed framework is periodically run, whereby at each epoch r_i, the decisions are taken based on: i) The information received from the previous epoch r_{i-1} about the states of different clouds and services; ii) The UE mobility information received from $\mathcal{MA}$. Let r_γ and $r_{\gamma-1}$ denote the current and previous epochs, respectively. $\mathcal{L}_\delta$ denotes the set of flavors that are available at each cloud δ. Γ and $\mathcal{L}_\delta$ capture the heterogeneity of 6G networks not seen in 5G deployments.

C. Variables and parameters definition

1) Parameters definition: In what follows, we summarize the different parameters used by the solutions, at the current epoch r_γ, from the previous epoch $r_{\gamma-1}$. First, we define the parameter $\hat{\mathcal{X}}_{i,k}$ transferred from the previous period that defines the VNF $j \in \hat{\eta}(\delta)$ hosted a the cloud δ that serves the UE i.
$\forall i \in \mathcal{N}_1, \forall \delta \in \Gamma, \forall j \in \hat{\eta}(\delta)$:

$$\hat{\mathcal{X}}_{i,j} = \begin{cases} 1 & \text{Iff } i \text{ is served by the VNF instance } j \text{ in the} \\ & \text{cloud network } \delta \text{ in the previous round } r_{\gamma-1} \\ 0 & \text{Otherwise} \end{cases}$$

Second, we define the parameter $\hat{\mathcal{Y}}_{j,\ell}$ transferred from the previous period that sets the flavor of a VNF j prior period.
$\forall \delta \in \Gamma, \forall j \in \hat{\eta}(\delta), \forall \ell \in \mathcal{L}_\delta$:

$$\hat{\mathcal{Y}}_{j,\ell} = \begin{cases} 1 & \text{Iff the VNF } j \text{ uses the flavor } \ell \text{ in the previous} \\ & \text{round } r_{\gamma-1} \\ 0 & \text{Otherwise} \end{cases}$$

Note that both parameters $\hat{\mathcal{X}}$ and $\hat{\mathcal{Y}}$ have fixed values taken from the previous periods, with each VNF used in these prior rounds using one flavor. The following equation shows that each running VNF from the previous round $r_{\gamma-1}$ has only one flavor:

$$\forall \delta \in \Gamma, \forall k \in \hat{\eta}(\delta) : \sum_{\ell \in \mathcal{L}_\delta} \hat{\mathcal{Y}}_{k,\ell} = 1 \qquad (1)$$

Third, we define the parameter $\hat{\mathcal{Z}}_{j,\delta}$ transferred from the previous period that defines the cloud $\delta \in \Gamma$ of a VNF $j \in \mathcal{V}_1$.
$\forall \delta \in \Gamma, \forall j \in \hat{\eta}(\delta)$:

$$\hat{\mathcal{Z}}_{j,\delta} = \begin{cases} 1 & \text{Iff the VNF } j \text{ is hosted at the cloud } \delta \text{ during the} \\ & \text{previous round } r_{\gamma-1} \\ 0 & \text{Otherwise} \end{cases}$$

2) Variables definition: In this subsection, we define the different variables used in this paper. Let $\mathcal{X}_{i,j}$ be a variable that defines the VNF j that should serve the UE i in the current period r_γ. In what follows, we define $\mathcal{X}_{i,j}$ as follow:
$\forall i \in \mathcal{N}, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2$:

$$\mathcal{X}_{i,j} = \begin{cases} 1 & \text{Iff } i \text{ is served by the VNF instance } j \text{ in the} \\ & \text{current round } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

Let $\mathcal{Y}_{j,\ell}$ be a variable that specifies the flavor ℓ that should be used by the VNF j in the current period r_γ. In what follows, we define $\mathcal{Y}_{j,\ell}$:
$\forall \delta \in \Gamma, \forall \ell \in \mathcal{L}_\delta, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2$:

$$\mathcal{Y}_{j,\ell} = \begin{cases} 1 & \text{Iff the VNF } j \text{ uses the flavor } \ell \text{ in the current} \\ & \text{round } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

Let $\mathcal{Z}_{j,\delta}$ be a variable that specifies the cloud $\delta \in \Gamma$ whereby the VNF $\forall j \in \mathcal{V}_1 \cup \mathcal{V}_2$ should be hosted:

$$\mathcal{Z}_{j,\delta} = \begin{cases} 1 & \text{Iff the VNF } j \text{ is hosted at the cloud } \delta \text{ round } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

Let $\mathcal{T}_j$ be a variable that shows if a VNF $j \in \mathcal{V}_1 \cup \mathcal{V}_2$ would stay running for the current period r_γ. Formally, the E2E slicer destroys an unwanted VNF:

$$\mathcal{T}_j = \begin{cases} 1 & \text{Iff the VNF } j \text{ should stay running in the current} \\ & \text{period } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

D. Constraints related with VNFs and UEs

In this subsection, we define the main constraints related to VNFs and UEs.

1) Constraints related to UEs: The following constraint shows that we serve each UE with one VNF.

$$\forall i \in \mathcal{N} : \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \mathcal{X}_{i,j} = 1 \qquad (2)$$

While constraint (3) ensures that a VNF should only stay running in the next round if it will serve at least one user in the future.

$$\forall j \; \mathcal{V}_1 \cup \mathcal{V}_2 : \mathcal{T}_j \leq \sum_{i \in \mathcal{N}_1 \cup \mathcal{N}_2} \mathcal{X}_{i,j} \qquad (3)$$

The E2E Slicer must preserve the requirements of the slice $\mathcal{S}$ for all the UEs.
$\forall t \in \Psi, \forall i \in \mathcal{N}_1 \cup \mathcal{N}_2$:

$$\underbrace{\sum_{\delta \in \Gamma, \ell \in \mathcal{L}_\delta} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} r^t_\ell \times \mathcal{X}_{i,j} \times \mathcal{Y}_{j,\ell} +}_{(4.a)}$$
$$\underbrace{\sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \mathcal{R}^t_{i,\delta} \times \mathcal{X}_{i,j} \times \mathcal{Z}_{j,\delta}}_{(4.b)} \rhd_t \Phi_t(\mathcal{S}) \qquad (4)$$

The $\rhd_t$ operation specifies the network requirement, e.g., the end-to-end delay between a user and a cloud. For instance, $\rhd_t$ can denote the operation $\leq$ in case of the end-to-end delay, while it should be $\geq$ for the case of end-to-end bandwidth.

While the part (4.a) shows the computational service of the VNF j using the flavor ℓ, e.g., computation delay, the part (4.b) represents propagation service between the user i and the cloud δ, e.g., propagation delay. However, the constant (4) is not linear. We replace this inequality to make that constraint linear using the following variables and constraints:

Let $\xi^{\delta}_{i,j}$ be a variable that shows if the VNF of a UE $i \in \mathcal{N}_1 \cup \mathcal{N}_2$ will be hosted at the cloud $\delta \in \Gamma$:

$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2:$

$$\xi^{\delta}_{i,j} = \begin{cases} 1 & \text{Iff the VNF of the user } i \text{ will be hosted in the cloud } \delta. \\ 0 & \text{Otherwise} \end{cases}$$

Let also $\zeta^{\ell}_{i,j}$ a decision Boolean variable that shows if the VNF j of a UE i uses the flavor ℓ.

$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall \ell \in \mathcal{L}_\ell:$

$$\zeta^{\ell}_{i,j} = \begin{cases} 1 & \text{Iff the VNF of } j \text{ of the user } i \text{ uses the flavor } \ell. \\ 0 & \text{Otherwise} \end{cases}$$

Also, we replace the constraint (4) with the following constraint:

$\forall t \in \Psi, \forall i \in \mathcal{N}_1 \cup \mathcal{N}_2:$

$$\sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \left(\mathcal{R}^t_{i,\delta} \times \xi^{\delta}_{i,j} + \sum_{\ell \in \mathcal{L}_\delta} r^t_\ell \times \zeta^{\ell}_{i,j} \right) \rhd_t \Phi_t(\mathcal{S}) \quad (5)$$

The following constraint ensures that the VNF of a UE $i \in \mathcal{N}_1 \cup \mathcal{N}_2$ should be hosted only at one cloud.

$$\forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall i \in \mathcal{N}_1 \cup \mathcal{N}_2: \quad \xi^{\delta}_{i,j} \geq \mathcal{X}_{i,j} + \mathcal{Z}_{j,\delta} - 1 \quad (6)$$

$$\forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall i \in \mathcal{N}_1 \cup \mathcal{N}_2: \xi^{\delta}_{i,j} \leq \mathcal{X}_{i,j} \quad (7)$$

$$\forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall i \in \mathcal{N}_1 \cup \mathcal{N}_2: \xi^{\delta}_{i,j} \leq \mathcal{Z}_{j,\delta} \quad (8)$$

$$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2: \sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \xi^{\delta}_{i,j} = 1 \quad (9)$$

Meanwhile, the following constraints ensure that a VNF j of a user i should use only one flavor.

$$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall \ell \in \mathcal{L}_\ell: \quad \zeta^{\ell}_{i,j} \geq \mathcal{X}_{i,j} + \mathcal{Y}_{j,\ell} - 1 \quad (10)$$

$$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall \ell \in \mathcal{L}_\ell: \quad \zeta^{\ell}_{i,j} \leq \mathcal{X}_{i,j} \quad (11)$$

$$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall \ell \in \mathcal{L}_\ell: \quad \zeta^{\ell}_{i,j} \leq \mathcal{Y}_{j,\ell} \quad (12)$$

$$\forall i \in \mathcal{N}_1 \cup \mathcal{N}_2: \sum_{\delta \in \Gamma, \ell \in \mathcal{L}_\ell} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \zeta^{\ell}_{i,j} = 1 \quad (13)$$

2) Constraints related to VNFs: The following equality ensures that a VNF would be hosted in a cloud if and only if it is activated. Moreover, an activated VNF should be hosted only on one cloud network.

$$\forall j \in \mathcal{V}_1 \cup \mathcal{V}_2: \sum_{\delta \in \Gamma} \mathcal{Z}_{j,\delta} = \mathcal{T}_j \quad (14)$$

Any VNF should use only one flavor if it is activated. For this reason, we add the following equation:

$$\forall j \in \mathcal{V}_1 \cup \mathcal{V}_2: \sum_{\delta \in \Gamma} \sum_{\ell \in \mathcal{L}_\delta} \mathcal{Y}_{j,\ell} = \mathcal{T}_j \quad (15)$$

Also, a VNF $j \in \mathcal{V}_1 \cup \mathcal{V}_2$ should use only one flavor $\ell \in \mathcal{L}_\delta$ from the cloud $\delta \in \Gamma$ where it is deployed. Otherwise, if it is undeployed in δ, it should not use any flavor.

$\forall j \in \mathcal{V}_1 \cup \mathcal{V}_2, \forall \delta \in \Gamma:$

$$\sum_{\ell \in \mathcal{L}_\delta} \mathcal{Y}_{j,\ell} \leq \mathcal{Z}_{j,\delta} \quad (16)$$

We define the constraints related to the resources available at each cloud $\delta \in \Gamma$. For each resource $\varphi \in \Omega$, we should ensure that the number of resources $\varphi \in \Omega$ used by all the VNFs hosted at the cloud $\gamma \in \Gamma$ do not exceed the capacity of that cloud in terms of that resources $\mathcal{F}_{\delta,\varphi}$.

$\forall \varphi \in \Omega, \forall \delta \in \Gamma:$

$$\sum_{j \in \mathcal{V}} \sum_{\ell \in \mathcal{L}_\delta} \Theta_{\ell,\varphi} \times \mathcal{Y}_{j,\ell} \leq \mathcal{F}_{\delta,\varphi} \quad (17)$$

The last constraints ensure that the number of resources used by the users of a VNF $j \in \mathcal{V}$ that uses the flavor $\ell \in \mathcal{L}_\delta$ should never exceed its capacity.

$\forall \varphi \in \Omega, \forall j \in \mathcal{V}:$

$$\sum_{i \in \mathcal{N}} \lambda^{\varphi}_i \times \mathcal{X}_{i,j} \leq \sum_{\delta \in \Gamma} \sum_{\ell \in \mathcal{L}_\delta} \Lambda^{\varphi}_\ell \times \mathcal{Y}_{j,\ell} \quad (18)$$

E. Constraints related to relocation time overhead

In this sub-section, we define the service relocation cost of each UE $i \in \mathcal{N}$. Let $\mathcal{F}_i$ denote the service relocation cost of a UE $i \in \mathcal{N}$. We varied the values of $\mathcal{F}_i$ according to different parameters. In what follows, we summarize the different possible scenarios of $\mathcal{F}_i$:

- **Use-case** (1) **VNF scaling up/down:** If the VNF of a UE i hosted at the cloud $\delta \in \Gamma$ has been scaled up or down, then the service of that UE can be affected for a specified amount of time. In this case, the relocation cost of that user would be defined as follow: $\mathcal{F}_i = \sigma_\delta$;
- **Use-case** (2) **A UE i relocated to another VNF without service mobility:** In this case, there are three scenarios:
  - UE i uses another existing VNF, in this case, the relocation cost exactly equals the cost to establish the new routes between that UE and his new VNF. Thus, $\mathcal{F}_i = \alpha$;
  - UE i uses a new existing VNF at cloud $\delta \in \Gamma$ that made a shrinking by either scaling up or down. In this case, the worst relocation cost equals the cost for scaling up/down that VNF plus the cost of establishing the new routes between that VNF and that UE. Thus, $\mathcal{F}_i = \alpha + \sigma_\delta$;

  - UE i is affected to a new VNF that will be created in the current round r_γ at the cloud $\delta \in \Gamma$. In this case, the relocation cost is $\mathcal{F}_i = \alpha + \omega_\delta$.
- **Use-case** (3) **A service mobility between two clouds:** In this case, we have two scenarios:
  - The VNF $j \in \eta(\delta_2)$ of UE i is moved from the cloud $\delta_1 \in \Gamma$ to another cloud $\delta_2 \in \Gamma$. In this case, the relocation cost equals to the cost of moving that VNF from the cloud δ_1 to the cloud δ_2 in addition to the relocation cost for establishing the routes between the UE i and its VNF. In this case, the relocation cost is $\mathcal{F}_i = \alpha + \beta^j_{\delta_1,\delta_2}$;
  - The VNF $j \in \eta(\delta_2)$ of UE i is moved from the cloud δ_1 to another cloud δ_2, as well as it performs scale up/down. In this case, the relocation cost is $\mathcal{F}_i = \alpha + \beta^j_{\delta_1,\delta_2} + \sigma_{\delta_2}$.

In what follows, we define a general formula (39) relocation time that captures all the possible above scenarios. To do so, we first define the different variables that would be used in that formula. In what follows, we define the variables $\mathcal{A}_i$, $\mathcal{B}^\delta_i$, $\mathcal{C}^\delta_i$ and $\mathcal{D}^j_{\delta_1,\delta_2}$ for $i \in \mathcal{N}, \forall\delta \in \Gamma$ and $\delta_1, \delta_2 \in \Gamma$, respectively:

Definition of the variable $\mathcal{A}_i$:

In what follows, we define the Boolean variable $\mathcal{A}_i$.

$\forall i \in \mathcal{N}$:

$$\mathcal{A}_i = \begin{cases} 1 & \text{Iff the VNF of a UE } \forall i \in \mathcal{N} \text{ has been changed in} \\ & \quad \text{the current round } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

As explained before, we defined $\mathcal{A}_i$ to take into account the amount α when the UE i has changed his VNF or used a new one. Formally, for each UE $i \in \mathcal{N}$, we have two cases:

- A new UE: In this case, $\mathcal{A}_i$ should equal one. Formally, we define $\mathcal{A}_i$ for $\forall i \in \mathcal{N}_2$ as follow:
$$\forall i \in \mathcal{N}_2 : \mathcal{A}_i = 1 \tag{19}$$
- Existing UE: In this case, $\mathcal{A}_i$ would be considered if the UE has changed his VNF between the previous $r_{\gamma-1}$ and current period r_γ, respectively. We need to consider the existing and the new VNFs $\forall j \in \mathcal{V}_1 \cup \mathcal{V}_2$, and check if the UE i has changed his VNF by using another VNF that can be either exiting or a new one $\mathcal{V}_1 \cup \mathcal{V}_2$. For this reason, $\mathcal{A}_i$, for $i \in \mathcal{N}_1$ is defined using the two following equations:
$$\forall i \in \mathcal{N}_1, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2 : \mathcal{A}_i \geq \hat{\mathcal{X}}_{i,j} - \mathcal{X}_{i,j} \tag{20}$$
$$\forall i \in \mathcal{N}_1, \forall j \in \mathcal{V}_1 \cup \mathcal{V}_2 : \mathcal{A}_i \leq 2 - \hat{\mathcal{X}}_{i,j} - \mathcal{X}_{i,j} \tag{21}$$

From (2), a UE i should use only one VNF. We have mainly two cases. In the first case, a UE will change his VNF, in that case, for a specified $j \in \mathcal{V}_1$, we will have $\hat{\mathcal{X}}_{i,j} = 1$ and $\mathcal{X}_{i,j} = 0$. Note that the UE i should use another VNF $j\prime \in \mathcal{N}$, while the VNF j can be either kept running or destroyed in the next round ($\mathcal{T}_j = 0$). From (20), $\mathcal{A}_i$ should equal to 1 in that case. In the second case, whereby the UE uses the same VNF j in both periods (i.e, the previous and the current periods), $\mathcal{A}_i$ should equal to zero. If $\hat{\mathcal{X}}_{i,j} = \mathcal{X}_{i,j} = 1$, for a specified j, then from (21), $\mathcal{A}_i$ should equal to 0.

Definition of the variable $\mathcal{B}^\delta_i$:

In what follows, we define $\mathcal{B}^\delta_i$ that specifies if a VNF of a UE $i \in \mathcal{N}$ in the cloud $\delta \in \Gamma$ made a shrinking by either scaling up or down.

$\forall i \in \mathcal{N}, \forall\delta \in \Gamma$:

$$\mathcal{B}^\delta_i = \begin{cases} 1 & \text{Iff the VNF selected by the UE } i \text{ in the cloud } \delta \\ & \quad \text{has been made scale up/down in } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

For computing $\mathcal{B}^\delta_i$, we consider only the VNFs that already running in the previous round $r_{\gamma-1}$ ($\forall k \in \hat{\eta}(\delta)$). Formally, $\mathcal{B}^\delta_i$ would be computed as follows:

- A new UE: In this case, $\mathcal{B}^\delta_i$ should be equal to zero. Formally, $\mathcal{B}^\delta_i$, for $\forall i \in \mathcal{N}_2$ and $\forall\delta \in \Gamma$, is defied as follow:
$$\forall\delta \in \Gamma, \forall i \in \mathcal{N}_2 : \mathcal{B}^\delta_i = 0 \tag{22}$$
- Existing UE: In this case, $\mathcal{B}^\delta_i$ would be considered iff the VNF of that UE has made scale up/down between the previous $r_{\gamma-1}$ and the current period r_γ, respectively. A scale up/down of a VNF j of a UE i has changed its flavor ℓ in the current period. Formally, the VNF j has made a scale up/down iff: $\hat{\mathcal{Y}}_{j,\ell} - \mathcal{Y}_{j,\ell} = 1$. In what follows, $\mathcal{B}^\delta_i$, for $\forall i \in \mathcal{N}_1$ and $\delta \in \Gamma$, is defined as follows:

  $\forall i \in \mathcal{N}_1, \forall\delta \in \Gamma, \forall\ell \in \mathcal{L}_\delta, \forall j \in \mathcal{V}_1$:
$$\mathcal{B}^\delta_i \geq \mathcal{Z}_{j,\delta} \times \mathcal{X}_{i,j} \times (\hat{\mathcal{Y}}_{j,\ell} - \mathcal{Y}_{j,\ell}) \tag{23}$$

  However, the inequality (23) is not linear. The constraint (23) would be transformed to linear optimization as follows:

  $\forall i \in \mathcal{N}_1, \forall\delta \in \Gamma, \forall\ell \in \mathcal{L}_\delta, \forall j \in \mathcal{V}_1$:
$$\mathcal{B}^\delta_i \geq \mathcal{Z}_{j,\delta} + \mathcal{X}_{i,j} + \mathcal{Y}_{j,\ell} - \hat{\mathcal{Y}}_{j,\ell} - 2 \tag{24}$$

  We will also add the following inequality for defining $\mathcal{B}^\delta_i$:

  $\forall i \in \mathcal{N}_1, \forall\delta \in \Gamma, \forall\ell \in \mathcal{L}_\delta, \forall j \in \mathcal{V}_1$:
$$\mathcal{B}^\delta_i \leq 4 - \mathcal{Z}_{j,\delta} - \mathcal{X}_{i,j} - \hat{\mathcal{Y}}_{j,\ell} - \mathcal{Y}_{j,\ell} \tag{25}$$

  From (24), $\mathcal{B}^\delta_i$, for a UE $i \in \mathcal{N}_2$ and a cloud $\delta \in \Gamma$, should equal 1 iff the VNF j of that UE is hosted at that cloud its flavor has changed in comparing to the previous period. While from (25), $\mathcal{B}^\delta_i$ should equal 0 if the VNF has used the same flavor as the previous period (i.e., $\hat{\mathcal{Y}}_{j,\ell} = \mathcal{Y}_{j,\ell} = 1$).

We also add the following constraint that ensures that the VNF of a UE can scale a maximum of one time during a period.
$$\forall i \in \mathcal{N} : \sum_{\delta\in\Gamma} \mathcal{B}^\delta_i \leq 1 \tag{26}$$

Definition of the variable $\mathcal{C}^\delta_i$:

In what follows, we define $\mathcal{C}^\delta_i, \forall i \in \mathcal{N}$ and $\forall\delta \in \Gamma$, which is a variable that shows if a UE is assigned to a new VNF. Formally, $\mathcal{C}_i$ is defined as follow: $\forall i \in \mathcal{N}, \delta \in \Gamma$:

$$\mathcal{C}^\delta_i = \begin{cases} 1 & \text{Iff the UE } i \in \mathcal{N} \text{ uses a new VNF in the cloud } \delta \\ & \quad \text{during the current round } r_\gamma \\ 0 & \text{Otherwise} \end{cases}$$

The following constraint ensures that a UE will use at most one new VNF during a period.

$$\forall i \in \mathcal{N} : \sum_{\delta \in \Gamma} \mathcal{C}_i^{\delta} \leq 1 \tag{27}$$

Moreover, we add the following constraints for tacking into account that the variable $\mathcal{C}_i^{\delta}$. We have the two following constraints:

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 : \mathcal{C}_i^{\delta} \geq \mathcal{X}_{i,j} \times \mathcal{Z}_{j,\delta} \tag{28}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma : \mathcal{C}_i^{\delta} \leq \sum_{j \in \mathcal{V}_2} \mathcal{X}_{i,j} \times \mathcal{Z}_{j,\delta} \tag{29}$$

From (28), $\mathcal{C}_i^{\delta}$ equals to 1 iff the UE i uses a new VNF $j \in \mathcal{V}_2$ (i.e., $\mathcal{X}_{i,j} = 1$) in the cloud $\delta \in \Gamma$ (i.e, $\mathcal{Z}_{j,\delta} = 1$). While, from (29), $\mathcal{C}_i^{\delta} = 0$ iff the UE i does not use any new VNF $j \in \mathcal{V}_2$ in the cloud $\delta \in \Gamma$. However, equations (28) and (29) are not linear. To convert the constraints (28) and (29), we replace these constraints by the following constraints and variables:

First, we add the following variables:

$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 :$

$$\Upsilon_{i,j}^{\delta} = \begin{cases} 1 & \text{Iff the UE } i \in \mathcal{N} \text{ uses the new VNF } j \text{ in the} \\ & \quad \text{cloud } \delta \text{ during the current round } r_{\gamma} \\ 0 & \text{Otherwise} \end{cases}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 : \Upsilon_{i,j}^{\delta} \leq \mathcal{X}_{i,j} \tag{30}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 : \Upsilon_{i,j}^{\delta} \leq \mathcal{Z}_{j,\delta} \tag{31}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 : \Upsilon_{i,j}^{\delta} \geq \mathcal{X}_{i,j} + \mathcal{Z}_{j,\delta} - 1 \tag{32}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma, \forall j \in \mathcal{V}_2 : \mathcal{C}_i^{\delta} \geq \Upsilon_{i,j}^{\delta} \tag{33}$$

$$\forall i \in \mathcal{N}, \forall \delta \in \Gamma : \mathcal{C}_i^{\delta} \leq \sum_{j \in \mathcal{V}_2} \Upsilon_{i,j}^{\delta} \tag{34}$$

Definition of the variable $\mathcal{D}_{\delta_1,\delta_2}^{j}$:

In what follows, we define $\mathcal{D}_{\delta_1,\delta_2}^{j}$ for $\delta_1, \delta_2 \in \Gamma$ and $j \in \mathcal{V}_1$. Formally, $\mathcal{D}_{\delta_1,\delta_2}^{j}$ shows if the VNF j will migrate from the cloud δ_1 to δ_2.

$\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2, \forall j \in \eta(\delta_1) :$

$$\mathcal{D}_{\delta_1,\delta_2}^{j} = \begin{cases} 1 & \text{Iff the VNF } j \text{ will migrate from the cloud } \delta_1 \\ & \quad \text{to the cloud } \delta_2 \text{ during the current round } r_{\gamma} \\ 0 & \text{Otherwise} \end{cases}$$

We have also the following constraints:

$$\forall \delta_1 \in \Gamma, \forall j \in \eta(\delta_1) : \sum_{\delta_2 \in \Gamma, \delta_1 \neq \delta_2} \mathcal{D}_{\delta_1,\delta_2}^{j} \leq 1 \tag{35}$$

$\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2, \forall j \in \eta(\delta_1) :$

$$\mathcal{D}_{\delta_1,\delta_2}^{j} \leq \hat{\mathcal{Z}}_{j,\delta_1} \tag{36}$$

$$\mathcal{D}_{\delta_1,\delta_2}^{j} \leq \mathcal{Z}_{j,\delta_2} \tag{37}$$

$$\mathcal{D}_{\delta_1,\delta_2}^{j} \geq \hat{\mathcal{Z}}_{j,\delta_1} + \mathcal{Z}_{j,\delta_2} - 1 \tag{38}$$

From (36), (37) and (38), $\mathcal{D}_{\delta_1,\delta_2}^{j} = 0$ iff $\hat{\mathcal{Z}}_{j,\delta_1} = 0$ or $\mathcal{Z}_{j,\delta_2} = 0$, which means that the VNF j is not migrated from the cloud δ_1 to the cloud δ_2. Formally, the VNF j would not migrated from δ_1 were not in the previous period ($\hat{\mathcal{Z}}_{j,\delta_1} = 0$), as well as it can not migrate to a new cloud δ_2 unless it is located on that cloud in the current period ($\hat{\mathcal{Z}}_{j,\delta_1} = 0$). Also, $\mathcal{D}_{\delta_1,\delta_2}^{j} = 1$ iff $\hat{\mathcal{Z}}_{j,\delta_1} = 1$ and $\mathcal{Z}_{j,\delta_2} = 1$, which means that the VNF j should migrate from the cloud δ_1 to the cloud δ_2.

Relocation time overhead:

The following constraint represents the general formula of relocation time that captures all the aforementioned scenarios.

$\forall i \in \mathcal{N} :$

$$\mathcal{F}_i = \alpha \times \mathcal{A}_i + \sum_{\delta \in \Gamma} \sigma_{\delta} \times \mathcal{B}_i^{\delta} + \sum_{\delta \in \Gamma} \omega_{\delta} \times \mathcal{C}_i^{\delta} + \sum_{\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2} \sum_{\forall j \in \eta(\delta_1)} \beta_{\delta_1,\delta_2}^{j} \times \mathcal{D}_{\delta_1,\delta_2}^{j} \times \mathcal{X}_{i,j} \tag{39}$$

However, the equation (39) is not linear due to the part ($\mathcal{D}_{\delta_1,\delta_2}^{j} \times \mathcal{X}_{i,j}$). We change this equality to linear optimization by replacing the equation (39) with the following variables and constraints:

$\forall i \in \mathcal{N}, \forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2 :$

$$\mathcal{E}_{i,j}^{\delta_1,\delta_2} = \begin{cases} 1 & \text{Iff the VNF } j \text{ of } i \text{ should be migrated from} \\ & \quad \delta_1 \text{ to } \delta_2. \\ 0 & \text{Otherwise} \end{cases}$$

$\forall i \in \mathcal{N} :$

$$\mathcal{F}_i = \alpha \times \mathcal{A}_i + \sum_{\delta \in \Gamma} \sigma_{\delta} \times \mathcal{B}_i^{\delta} + \sum_{\delta \in \Gamma} \omega_{\delta} \times \mathcal{C}_i^{\delta} + \sum_{\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2} \sum_{\forall j \in \eta(\delta_1)} \beta_{\delta_1,\delta_2}^{j} \times \mathcal{E}_{i,j}^{\delta_1,\delta_2} \tag{40}$$

$$\forall i \in \mathcal{N}, \forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2, \forall j \in \eta(\delta_1) : \mathcal{E}_{i,j}^{\delta_1,\delta_2} \geq \mathcal{D}_{\delta_1,\delta_2}^{j} + \mathcal{X}_{i,j} - 1 \tag{41}$$

$$\forall i \in \mathcal{N}, \forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2, \forall j \in \eta(\delta_1) : \mathcal{E}_{i,j}^{\delta_1,\delta_2} \leq \mathcal{D}_{\delta_1,\delta_2}^{j} \tag{42}$$

$$\forall i \in \mathcal{N}, \forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2, \forall j \in \eta(\delta_1) : \mathcal{E}_{i,j}^{\delta_1,\delta_2} \leq \mathcal{X}_{i,j} \tag{43}$$

V. Proposed solutions

In this paper, we propose three unique solutions with different objectives, delay-aware network service management (DELAY-NSM), resource relocation aware network service management (R-NSM), and fair trade-off for network service management solution (FT-NSM). Our DELAY-NSM solution goal is to reduce the services' network end-to-end delay through instantiation, migration, or both of these services. R-NSM goal is to reduce the service relocation of different services to mitigate service interruption. Finally, FT-NSM aims to find a fair Pareto optimal configuration for optimizing the end-to-end delay and the service relocation. Those solutions are integrated into the NFVO to guarantee an agile and smooth service mobility orchestration.

A. DELAY-NSM: Delay-aware network service management solution

In this subsection, we describe DELAY-NSM that aims at minimizing the end-to-end delay. Basically, the end-to-end delay would be minimized by reducing the end-to-end delay $\mathcal{R}_{i,\delta}^{\text{DELAY}}$ between any UE $i \in \mathcal{N}$ and cloud $\delta \in \Gamma$. However, we can extend our proposed solution easily by considering any other type of resources $t \in \Psi$ by reducing the resource $\mathcal{R}_{i,\delta}^{t}$ between any UE i and cloud δ. We formulate the DELAY-NSM as follow:

$$\textbf{OP1} : \min \max_{i \in \mathcal{N}} \sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \Big(\mathcal{R}_{i,\delta}^{\text{DELAY}} \times \xi_{i,j}^{\delta} + \sum_{\ell \in \mathcal{L}_\delta} r_{\ell}^{\text{DELAY}} \times \zeta_{i,j}^{\ell} \Big) \tag{44}$$

S.t, (2), (3), (5), (6), (7), (8), (9), (10), (11), (12), (14), (15), (16), (17), (18), (19), (20), (21), (22), (24), (25), (26), (27), (30), (31), (32), (33), (34), (35), (36), (37), (38), (40), (41), (42), (43),

$$\forall i \in \mathcal{N} : \mathcal{F}_i \leq \mathcal{F}_{\text{Th}} \tag{45}$$

$$\sum_{j \in \mathcal{V}} \sum_{\delta \in \Gamma} \sum_{\ell \in \mathcal{L}_\delta} \psi_{j,\ell} \times \mathcal{Y}_{j,\ell} \leq \psi_{\text{Th}} \tag{46}$$

$\mathcal{F}_{\text{Th}}$ denotes the maximum service relocation threshold tolerated for each UE $i \in \mathcal{N}$ in the network, with ψ_{Th} denoting the maximum cost that is allowed for managing different services in the network.

B. R-NSM: Resource relocation aware network service management solution

In this subsection, we describe our R-NSM solution, which aims at minimizing the service relocation, including system disruption and end-users' QoS. R-NSM goal is to ensure the required end-to-end services including, delay, bandwidth, and jitter, while preventing the service relocation, e.g., service mobility, and vertical/horizontal scaling, that harms the SLA. We formulate the R-NSM solution as follow:

$$\textbf{OP2} : \min \max_{i \in \mathcal{N}} \alpha \times \mathcal{A}_i + \sum_{\delta \in \Gamma} \sigma_\delta \times \mathcal{B}_i^{\delta} + \sum_{\delta \in \Gamma} \omega_\delta \times \mathcal{C}_i^{\delta} + \sum_{\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2} \sum_{\forall j \in \eta(\delta_1)} \beta_{\delta_1,\delta_2}^{j} \times \mathcal{E}_{i,j}^{\delta_1,\delta_2} \tag{47}$$

S.t,

(2),(3), (5), (6), (7), (8), (9), (10), (11), (12), (14), (15), (16), (17), (18), (19), (20), (21), (22), (24), (25), (26), (27), (30), (31), (32), (33), (34), (35), (36), (37), (38), (41), (42), (43), (46),

$$\forall i \in \mathcal{N} : \sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \Big(\mathcal{R}_{i,\delta}^{\text{DELAY}} \times \xi_{i,j}^{\delta} + \sum_{\ell \in \mathcal{L}_\delta} r_{\ell}^{\text{DELAY}} \times \zeta_{i,j}^{\ell} \Big) \leq \varpi_{\text{Th}} \tag{48}$$

C. FT-NSM: Fair trade-off network slice orchestration solution

Our proposed solutions, DELAY-NSM and R-NSM, aims to optimize one objective, either the end-to-end delay or service relocation while, immolating the other. DELAY-NSM solution goal is to place the VNFs close to the UEs to the end-to-end delay and increase the bandwidth communication. However, as the users move around, this solution relocates VNFs more often, harming the service availability. On another side, the R-NSM solution favorites the service relocation objective. It aims to prevent service relocation by keeping the services at the same location with the same configuration as much as possible. However, this strategy harms the end-to-end delay due to UE mobility.

As these objectives are conflicting and we are unable to ensure both at the same time. We design FT-NSM to optimize both goals and find a fair trade-off between them addressing these conflict objectives. FT-NSM leverages the bargaining game – a type of cooperative game – to optimize and find a fair trade-off between the two goals. We modeled the two objectives as two players that would like to barter goods. Following, we introduce the concept of the bargaining game, whereby explaining the Nash bargaining Model (NBS) and Kalai and Smorodinsky Bargaining Model (KSBS), we can derive the FT-NSM solution.

1) Cooperative games: Cooperative games, especially the bargaining game, have been widely used in the literature to find Pareto optimal solutions, whereby there are no other alternative configurations whose objectives would have better outcomes. In the bargaining game, the players (i.e., objectives) would attain either the most convenient point when negotiation succeeds or disagreement point when negotiation fails. Let $\mathcal{P}$ denote the vector of payoffs of the two players (i.e., end-to-end delay and service relocation). Formally, we define $\mathcal{P}$ as a set of expected utility functions of players according to their played strategies. Let $\mathcal{P} = \{(u_1(x), u_2(x)), x = (x_1, x_2) \in X\}$, such that X is the set of the two players' strategies while $u_1(x)$ and $u_2(x)$ are the utility functions of the two players, respectively.

John Nash has suggested Nash Bargaining Model [26] that ensures a fair trade-off (i.e., Nash Equilibrium) between two players in a collaborative game. Nash has proven that the obtained configuration ensure the following axioms: $i)$ Feasibility; $ii)$ Pareto Optimality; $iii)$ Independence of irrelevant alternatives; $iv)$ Invariance under change of location and scale. Kalai and Smorodinsky [27] have proven that the Nash bargaining model (NBS) [26] has some limitations for ensuring fairness between the two players, and then they provide a new game, named KSBS, that ensures a fair Pareto optimal solution. KSBS is a cooperative game with a non-transferable utility that we measured in non-comparable units. KSBS enhances fairness by sharing the same utility fraction r among the two players. KSBS preserves all the axioms of NSB except the independence of irrelevant alternatives. Also, the KSBS game adds an extra axiom called monotonically. Moreover, the KSBS game needs: $i)$ $\mathcal{P}$ vector payoffs; $ii)$ The disagreement point $d = (u_1^d, u_2^d) \in \mathcal{P}$ that represents the pair of utility whereby the two players fail to achieve an agreement; iii) The ideal point for both players $x^b = (u_1^b, u_1^b)/x^b \in \mathcal{P}$ that refers to the best utility function that both players can achieve separately without bargaining. The aim of KSBS game is to find a fair and Pareto reasonable point, $(u_1^*, u_2^*) = f(\mathcal{P}, d, b) \in \mathcal{P}$ that satisfies both players. Kalai and Smorodinsky have proven that the unique fair Pareto optimal solution that satisfies KSBS's axioms is the solution of the following optimization problem:

$$\max r \tag{49}$$

s.t

$$(u_1(x), u_2(x)) \in \mathcal{P} \tag{50}$$

$$r = \frac{u_1(x) - u_1^d}{u_1^b - u_1^d} \tag{51}$$

$$r = \frac{u_2(x) - u_2^d}{u_2^b - u_2^d} \tag{52}$$

2) FT-NSM description: FT-NSM solution finds fair Pareto optimal configuration for optimizing both the end-to-end delay and service relocation by leveraging KSBS bargaining game. In FT-NSM, both objectives end-to-end delay and service relocation are considered as two players that would like to barter goods. Let $b = (u_{\mathcal{D}}^b, u_{\mathcal{R}}^b)$ and $d = (u_{\mathcal{D}}^d, u_{\mathcal{R}}^d)$ the ideal and threat points of KSBS game, respectively. As aforementioned, DELAY-NSM, defined using the (**OP1**), (resp., R-NSM defined by (**OP2**)) aims to optimize the end-to-end delay (resp., service relocation) without considering the other parameter. Hence, the points $b = (u_{\mathcal{D}}^b, u_{\mathcal{R}}^b)$ and $d = (u_{\mathcal{D}}^d, u_{\mathcal{R}}^d)$ can be computed from the following equations (53) and (54) and by substituting the values of the variables derived from the optimizations (**OP1**) and (**OP2**), respectively.

$$\max_{i \in \mathcal{N}} \alpha \times \mathcal{A}_i + \sum_{\delta \in \Gamma} \sigma_\delta \times \mathcal{B}_i^\delta + \sum_{\delta \in \Gamma} \omega_\delta \times \mathcal{C}_i^\delta + \sum_{\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2} \sum_{\forall j \in \eta(\delta_1)} \beta_{\delta_1, \delta_2}^j \times \mathcal{E}_{i,j}^{\delta_1, \delta_2} \tag{53}$$

$$\max_{i \in \mathcal{N}} \sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \left(\mathcal{R}_{i,\delta}^{\text{DELAY}} \times \xi_{i,j}^\delta + \sum_{\ell \in \mathcal{L}_\delta} r_\ell^{\text{DELAY}} \times \zeta_{i,j}^\ell \right) \tag{54}$$

While $u_{\mathcal{R}}^d$ and $u_{\mathcal{D}}^b$ are obtained by substituting the values of the variables obtained by solving (**OP1**) in equations (53) and (54), respectively. $u_{\mathcal{R}}^b$ and $u_{\mathcal{D}}^d$ would be achieved by substituting the values of the variables obtained by solving (**OP2**) in equations (53) and (54), respectively. Let $(u_{\mathcal{R}}^*, u_{\mathcal{D}}^*) \in \mathcal{P}$ the fair Pareto optimal configuration. FT-NSM solution is formulated as follows:

$$\max r$$

s.t

(2),(3), (5), (6), (7), (8), (9), (10), (11), (12), (14), (15), (16), (17), (18), (19), (20), (21), (22), (24), (25), (26), (27), (30), (31), (32), (33), (34), (35), (36), (37), (38), (41), (42), (43), (46), (48),

$\forall i \in \mathcal{N}$:

$$\alpha \times \mathcal{A}_i + \sum_{\delta \in \Gamma} \sigma_\delta \times \mathcal{B}_i^\delta + \sum_{\delta \in \Gamma} \omega_\delta \times \mathcal{C}_i^\delta + \sum_{\forall \delta_1, \delta_2 \in \Gamma, \delta_1 \neq \delta_2} \sum_{\forall j \in \eta(\delta_1)} \beta_{\delta_1, \delta_2}^j \times \mathcal{E}_{i,j}^{\delta_1, \delta_2} \leq u_{\mathcal{R}}^* \tag{55}$$

$\forall i \in \mathcal{N}$:

$$\sum_{\delta \in \Gamma} \sum_{j \in \mathcal{V}_1 \cup \mathcal{V}_2} \left(\mathcal{R}_{i,\delta}^{\text{DELAY}} \times \xi_{i,j}^\delta + \sum_{\ell \in \mathcal{L}_\delta} r_\ell^{\text{DELAY}} \times \zeta_{i,j}^\ell \right) \leq u_{\mathcal{D}}^* \tag{56}$$

$$r = \frac{u_R^* - u_R^d}{u_R^b - u_R^d} \tag{57}$$

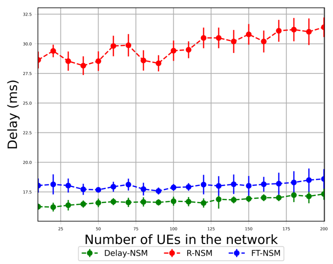
$$r = \frac{u_D^* - u_D^d}{u_D^b - u_D^d} \tag{58}$$

VI. Performance evaluation

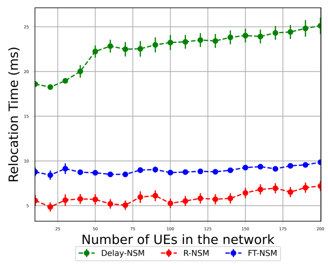
We evaluated our proposal using the Gurobi [28] optimizer as an optimization solver tool. We developed our simulation environment using the Python Language and Networkx library. This work handles network slice mobility under various objectives by considering the service relocation besides the end-to-end delay, focusing on how to find new feasible configuration/deployment solutions on top of MECs, complementing research works (i.e., Nikaein et al. [29]) that aimed at implementing the Network Edge support. We evaluated our solutions using a discrete-time simulation, where at each time slot, an event happens, such as the mobility of a UE from one location to another. Moreover, we split the 2D plane where the UEs move into different regions, each with a set of eNB/gNB and MEC hosts. We model the mobility of UEs using a random walk process [30]–[36], and we only use one solution (i.e., Delay-NSM, R-NSM, and FT-NSM) at each simulation execution. We choose to use the random walk mobility pattern because it is the worst-case scenario for any proposal aiming to compute network configurations in a dynamic environment. Our framework is orthogonal to any mobility patterns and quickly adapts to network changes. The framework needs only information about the new location of UEs that should get from the RAN (i.e., gNB). As in our work, we could use any information available about the UEs' mobility behavior to speculative precompute new configurations solutions. We have leveraged pymobility[1] source code to implement the random walk mobility of different UEs.

We verify each simulation step by checking if the result violates the UEs' SLA regarding its end-to-end delay. If so, we execute one of our solutions (i.e., DELAY-NSM, R-NSM, or FT-NSM) to decide to perform service relocation to assign the right amount of resources to the VNFs and place them at the right places, including VNF scaling up/down, VNF creation/deletion, or VNF migration. To provide network

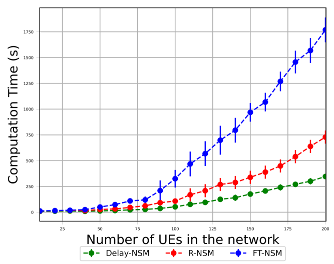
[1]https://github.com/panisson/pymobility



(a) Delay comparison under the diverse proposed approaches.



(b) Relocation comparison under the diverse proposed approaches.



(c) Computational cost comparison for the diverse proposed approaches.

Figure 3: Performance of the proposed solutions as a function of number of UEs.

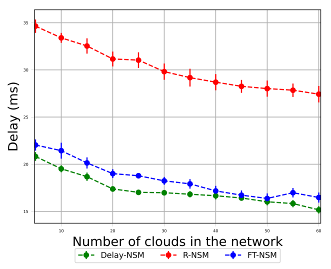
configuration within a reasonable time, we have executed the three solutions on UEs batches. We organized the UEs in batches, where we configured the VNFs of UEs that belong to the same batch simultaneously using one of the proposed solutions. In contrast, we configure the VNFs of UEs that belong to different batches sequentially. This helps to reduce the number of variables in the models of the three solutions, which has a positive impact on the execution time. In the simulation, we have considered the size of the batch equals to 30 UEs.

Our simulations randomly generate the environment's components, i.e., UEs, Edge-Clouds, VNFs, network slice, and their resources requirements, to reproduce a realistic environment. In the simulation, we have conducted two sets of experiments. Firstly, we have varied the number of UEs while fixing the number of clouds in the network by 40 and eNB/gNB by 50. Secondly, we have changed the number of clouds while fixing the number of UE by 100 and eNB/gNB by 50. In all the simulation scenarios, while α has been fixed by $4ms$, $\beta^{j}_{\delta_1,\delta_2}$, σ_δ and ω_δ are randomly selected from an interval $[4ms, 8ms]$. In all the simulations, we have set: 1) The maximum tolerated end-to-end delay in each network slice ϖ_{Th} by $30ms$; 2) The maximum cost budget for managing the network services ψ_{Th} is fixed by $1200units$; 3) The maximum service relocation threshold $\mathcal{F}_{\text{Th}}$ is fixed by 300; 4) We set the minimum and maximum propagation delay between an eNB/gNB and a cloud have to $5ms$ and $60ms$, respectively. 5) We have used 8 flavors $\mathcal{L}$ in the network, each of which has different resources (i.e., RAM, CPU, and Disk) and costs. We repeat each simulation 80 times, then compute the mean and the 95% confidence interval of the delay and the relocation time in milliseconds, as well as the time required (i.e., complexity) to execute each solution in seconds.

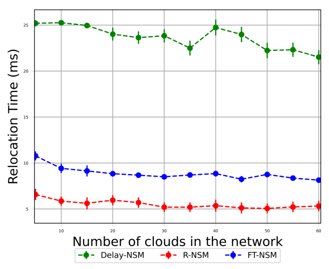
We evaluate how the number of UEs impacts our proposed solutions as presented in Figure 3. Figure 3(a) presents the results of our simulations for Delay-NSM, R-NSM, and FT-NSM, presenting in its Y-axis the communication delay between Edge-Clouds and UEs and in the X-axis the number of UEs. Figure3(a) shows that the delay for each solution varies over a fixed interval, allowing us to conclude that our solutions are indifferent to the number of UEs. In contrast to Delay-NSM and FT-NSM, the number of UEs has a slightly negative impact on R-NSM. We explain this as follows, the increase in the number of UEs raises the likelihood that UEs move away from their original running services, which harms the delay. In contrast to the R-NSM solution, Delay-NSM and FT-NSM optimize the delay by relocating the services more often to follow the UE mobility. The R-NSM solution's main objective is to prevent service relocation by keeping the services at the same location as much as possible. Thus, an increase in the number of UEs hurts the performances in terms of delay in the R-NSM solution. However, among these solutions, the delay does vary significantly, between 16 ms and 17.5 ms to Delay-NSM, 29 ms and 32 ms to R-NSM, and FT-NSM the communication delay moderates around 18 ms. The obtained results demonstrated the efficiency of the FT-NSM solution in terms of delay. It achieves almost a similar delay like Delay-NSM, whose objective is to reduce only the delay.

Figure 3(b) shows how the number of UEs impacts the relocation time of our solutions, with the Y-axis showing the relocation time of VNFs in ms while keeping the same representation for the X-axis. As expected, Delay-NSM has the worst results when relocation time is the factor to optimize with its relocation time going from $18ms$ to $25ms$. R-NSM solution reported a relocation time between $5ms$ and $7.5ms$, and FT-NSM solution between Delay-NSM and R-NSM solutions with a relocation time varying from 8 to $10ms$. From this figure, we observe that the R-NSM solution has the best performance in terms of delay. R-NSM solution's main target objective is to reduce the service relocation while the delay overhead never exceeds the threshold. In contrast to R-NSM and FT-NSM solutions, the number of UEs hurts the service relocation in the Delay-NSM solution. The Delay-NSM solution aims to minimize the delay while the service relocation does not exceed a specific threshold. The more UEs are, the higher likelihood that those UEs go far away from their original service becomes. In this case, the Delay-NSM solution will relocate the services following UE mobility, which harms the service relocation. The FT-NSM's results for both the delay and the relocation experiments show the possibility of finding a slice configuration that guarantees both an acceptable delay and relocation time when increasing the number of UEs. The obtained results demonstrate the efficiency of FT-NSM for achieving a fair Pareto optimal configuration that optimizes both objectives. FT-NSM has almost similar performances to Delay-NSM in terms of delay and performs like R-NSM in service relocation.

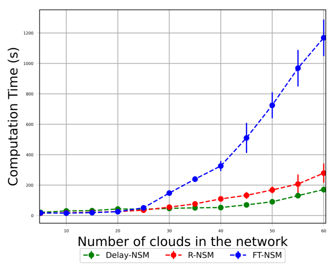
Figure 3(c) presents the computational cost of our proposed solutions, the Y-axis is the required time in seconds to solve



(a) Delay comparison under the diverse proposed approaches.



(b) Relocation comparison under the diverse proposed approaches.



(c) Computational cost comparison for the diverse proposed approaches.

Figure 4: Performance of the proposed solutions as a function of number of clouds.

the optimization problem on our computational resource, a dual Intel Xeon E5-2680 with 256 GB of main memory running Ubuntu 16.04, and the X-axis is the number of UEs, as before, we bached the UEs in our simulation. Our results show that the computational cost increases faster for FT-NSM than for Delay-NSM and R-NSM. Both Delay-NSM and R-NSM show a linear behavior to their computational cost. Meanwhile, FT-NSM has an interesting behavior caused by our batch size, as we have a fixed cost until passing 75 UEs, and then show a linear increase to its cost in the steady-state. These results allow us to conclude that finding a trade-off when increasing the number of UEs became rapidly costly in terms of computational time, which will impact the slice's performance. Furthermore, the increase in the number of UEs causes Edge-Clouds to saturate, which incurs an increased computational cost for FT-NSM.

Figure 4 depicts the impact of the number of clouds on the proposed solutions R-NSM, Delay-NSM, and FT-NSM. Figure 4(a) shows the effect of the number of clouds on the delay. The first observation that we can draw from this figure is that the number of clouds positively impacts all the delays. Increasing the number of clouds would create more opportunities for the users to host their services on the clouds that can offer better end-to-end delay. We also observe that both Delay-NSM and FT-NSM solutions have better performances compared to the R-NSM solution. While Delay-NSM and FT-NSM solutions aim to optimize the delay, the R-NSM solution reduces only service relocation. For instance, Delay-NSM and FT-NSM solutions achieve almost similar delays between $15ms$ and $22ms$ with a slight superiority of the Delay-NSM solution. Meanwhile, R-NSM solution archives delay between $27.5ms$ and $35ms$. The obtained results demonstrate the efficiency of FT-NSM for achieving almost similar performance like Delay-NSM whose first objective is the minimization of the delay.

Figure 4(b) depicts the impact of the number of clouds on the service relocation. The first observation that we can draw from this figure is that the number of clouds does not significantly impact service relocation. We also observe that both R-NSM and FT-NSM have similar performances, which are better than those achieved by the Delay-NSM solution. From this figure, R-NSM has a service relocation time between $5ms$ and $6.5ms$, whereas the service relocation time in FT-NSM is between $8ms$ and $10.5ms$. In contrast, the Delay-NSM solution has service relocation between $21ms$ and $25ms$. The obtained results demonstrated the superiority of R-NSM in terms of service relocation. Also, they show that the FT-NSM solution has similar performances to the R-NSM solution. We conclude that the FT-NSM solution succeeded in finding a fair Pareto optimal configuration that simultaneously optimizes the delay and service relocation.

Figure 4(c) presents the computational cost of our proposed solutions, the Y-axis is the required time in seconds to solve the optimization problem on our computational resource, and the X-axis is the number of clouds, in this scenario, we keep the number of UEs fixed in 100. Our results show that the computational cost increases faster for FT-NSM than for Delay-NSM, and R-NSM presents a linear behavior to their computational cost. Meanwhile, FT-NSM results demand a more careful analysis of its computational cost behavior regarding the number of clouds. Closer scrutiny of our results show three different behaviors caused by the batch size initially, we have a fixed cost, it then shows a faster linear increase in the 25 to 40 clouds interval, and finally, its sets on its steady-state behavior as we can see from 40 to 60 clouds, batching the UE allowed us to avoid an exponential computational cost. These results allow us to conclude that when increasing the number of clouds, as for UEs, finding a trade-off became rapidly costly in terms of computational time, which impacts the slice's performance.

VII. Conclusion

Unlike the previous generations that use the "one-fit-all" concept, the 6G system will be designed to consider diverse, with often-conflicting objectives and requirements in a multi-cloud, multi-administration environment. The network slicing concept evolve to play a central role in offering the agility and customizability for the 6G system. 6G use cases will have very demanding requirements, such as high reliability with a block error rate, i.e., BLER, lower than 10^{-9}, and low latency order of $1\ ms$. In this paper, we designed three solutions that aim to efficiently orchestrate the network slice, reducing latency, and increasing reliability. Our first solution aims to reduce the end-to-end delay, while the second one optimizes the service relocation. Finally, FT-NSM employs a bargaining game theory that provides a fair Pareto optimal configuration for optimizing the end-to-end delay and service relocation. The obtained results demonstrate the efficiency of each solution for achieving its main design goals.

REFERENCES

[1] G. T. 28.530, "5G; management and orchestration; concepts, use cases and requirements." [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3273

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[3] P. Schneider, C. Mannweiler, and S. Kerboeuf, "Providing strong 5g mobile network slice isolation for highly sensitive third-party services," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.

[4] E. NFV, "https://www.etsi.org/committee/nfv."

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[6] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, September 2013.

[7] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6G architecture to connect the worlds," *IEEE Access*, vol. 8, pp. 173 508–173 520, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3025032

[8] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," in *IEEE INFOCOM*, April 2014, pp. 574–582.

[9] T. He, S. Chen, H. Kim, L. Tong, and K. Lee, "To migrate or to wait: Bandwidth-latency tradeoff in opportunistic scheduling of parallel tasks," in *INFOCOM*, March 2012, pp. 2871–2875.

[10] L. Gao and G. N. Rouskas, "Virtual network reconfiguration with load balancing and migration cost considerations," in *IEEE Conference on Computer Communications (INFOCOM)*, April 2018, pp. 2303–2311.

[11] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *2010 Ninth International Conference on Grid and Cloud Computing*, 2010, pp. 87–92.

[12] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *2013 IFIP/IEEE IM*, 2013, pp. 177–184.

[13] G. Somani, P. Khandelwal, and K. Phatnani, "Vupic virtual machine usage based placement in iaas cloud," *abs/1212.0085*, 2012.

[14] W. Shi and B. Hong, "Towards profitable virtual machine placement in the data center," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 2011, pp. 138–145.

[15] T. Taleb and A. Ksentini, "Gateway relocation avoidance-aware network function placement in carrier cloud," in *16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '13, Barcelona, Spain, November 3-8, 2013*. ACM, 2013, pp. 341–346.

[16] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *IEEE WCNC*, 2014, pp. 2402–2407.

[17] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for the creation of efficient virtual core network slices in 5g mobile systems," *IEEE JSAC*, vol. 36, no. 3, pp. 469–484, 2018.

[18] H. Li, L. Wang, X. Wen, Z. Lu, and L. Ma, "Constructing service function chain test database: An optimal modeling approach for coordinated resource allocation," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[19] X. Zheng and Y. Cai, "Dynamic virtual machine placement for cloud computing environments," in *2014 43rd International Conference on Parallel Processing Workshops*, Sept 2014, pp. 121–128.

[20] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *IEEE ICC*, May 2016, pp. 1–6.

[21] H. Hawilo, M. Jammal, and A. Shami, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, Sept 2017, pp. 1–6.

[22] I. Ahmad, Z. Kaleem, R. Narmeen, L. D. Nguyen, and D.-B. Ha, "Quality-of-service aware game theory-based uplink power control for 5g heterogeneous networks," *Mobile Networks and Applications*, vol. 24, no. 2, pp. 556–563, 2019.

[23] J. Huang, C. Xing, Y. Qian, and Z. J. Haas, "Resource allocation for multicell device-to-device communications underlaying 5g networks: A game-theoretic mechanism with incomplete information," *IEEE TVT*, vol. 67, no. 3, pp. 2557–2570, 2018.

[24] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *IEEE INFOCOM*, 2019, pp. 10–18.

[25] G. T. 22.862, "Feasibility study on new services and markets technology enablers for critical communications, rel.14, jun. 2016."

[26] J. F. Nash, "The Bargaining Problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.

[27] E. Kalai and M. Smorodinsky, "Other Solutions to Nash's Bargaining Problem," *Econometrica*, vol. 43, no. 3, pp. 513–518, 1975. [Online]. Available: http://www.jstor.org/stable/1914280

[28] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2019. [Online]. Available: http://www.gurobi.com

[29] N. Nikaein, X. Vasilakos, and A. Huang, "Ll-mec: Enabling low latency edge applications," in *IEEE CloudNet*, 2018, pp. 1–7.

[30] Kuo-Hsing Chiang and N. Shenoy, "A 2-d random-walk mobility model for location-management studies in wireless networks," *IEEE TVT*, vol. 53, no. 2, pp. 413–424, 2004.

[31] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "A random walk around the city: New venue recommendation in location-based social networks," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, 2012, pp. 144–153.

[32] M. Dehghani Soltani, A. A. Purwita, Z. Zeng, C. Chen, H. Haas, and M. Safari, "An orientation-based random waypoint model for user mobility in wireless networks," in *IEEE ICC Workshops*, 2020, pp. 1–6.

[33] S. Shao, A. Khreishah, and M. Ayyash, "Evaluating the feasibility of random waypoint model for indoor wireless networks," *Internet Technology Letters*, vol. n/a, no. n/a, p. e214.

[34] N. Pillay and H. Xu, "Large intelligent surfaces: Random waypoint mobility and two-way relaying," *International Journal of Communication Systems*, vol. 33, no. 14, p. e4505, 2020, e4505 dac.4505.

[35] V. Yajnanarayana, H. Rydén, and L. Hévizi, "5g handover using reinforcement learning," in *IEEE 5GWF*, 2020, pp. 349–354.

[36] D. Basu, A. Jain, R. Datta, and U. Ghosh, "Optimized controller placement for soft handover in virtualized 5g network," in *IEEE WCNCW*, 2020, pp. 1–8.