871780 — MonB5G — ICT-20-2019-2020

Deliverable D5.3
Final report on AI-driven MonB5G energy efficiency techniques

Document Summary Information

Grant Agreement No	871780	**Acronym**	MonB5G
Full Title	Distributed Management of Network Slices in beyond 5G		
Start Date	01/11/2019	**Duration**	42 months
Project URL	https://www.monb5g.eu/		
Deliverable	D5.3 - Final report on AI-driven MonB5G energy efficiency techniques		
Work Package	WP5		
Contractual due date	M34	**Actual submission date**	30/08/2022
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	CTTC		
Responsible Author	Luis Blanco (CTTC), Hatim Chergui (CTTC)		
Contributions from	Luis Blanco (CTTC), Engin Zeydan (CTTC), Hatim Chergui (CTTC), Farhad Rezazadeh (CTTC), Anestis Dalgkitsis (IQU), Vasiliki Vlahodimitropoulou (OTE)		

Revision history

Version	Issue Date	Complete(%)	Changes	Contributor(s)
V0.1	01/06/2022	1%	First template with the ToC	Luis Blanco
V0.9	02/08/2022	95%	First draft ready for review	All
V1.0	30/08/2022	100%	Reviewed deliverable	All

Disclaimer

Copyright message

TABLE OF CONTENTS

List of Figures

List of Tables

List of Acronyms

Acronym	*Description*
3GPP	*Third Generation Partnership Project*
AE	Analytic Engine
AE-F	Analytic Engine Function
AE-S	Analytic Engine Sublayer
AI	Artificial Intelligence
BSS	Business Support System
CLA	Closed-loop Automation
CNF	Cloud Native function
DE	Decision Engine
DE-F	Decision Engine Function
DE-S	Decision Engine Sublayer
EEM	Embedded Element Manager
eMBB	Enhanced Mobile Broadband
eTOM	Enhanced Telecom Operations Map
ETSI	European Telecommunications Standards Institute
ECA	Event Condition Action
ENI	Experiential Networked Intelligence
FCAPS	Fault, Configuration, Accounting, Performance, Security
ISM	In-Slice Management
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LCM	Lifecycle Management
ML	Machine Learning
MANO	Management and Orchestration
MaaS	Management as a Service
MAN-F	Management Function
mMTC	Massive Machine Type Communications

MEO	MEC Orchestrator
MNO	Mobile Network Operator
MLaaS	MonB5G Layer as a Service
MS	Monitoring System
MS-F	Monitoring System Function
MS-S	Monitoring System Sublayer
MEC	Multi-access Edge Computing
NFVO	Network Function Virtualization Orchestrator
NSD	Network Service Descriptor
NSO	Network Service Orchestrator
NSP	Network Service Provider
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnetwork Management Function
NST	Network Slice Template
NSSI	Network sub-Slice Instance
NGMN	Next Generation Mobile Networks
NFVI	NFV Infrastructure
OAI	Open Air Interface
ONAP	Open Network Automation Platform
OPEX	Operational Expenditure
OSM	Open-Source MANO
OSS	Operation System Support
PaaS	Platform as a Service
PoC	Proof of Concept
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
SON	Self-Organizing Network
SLA	Service Level Agreement

SFL	Slice Functional Layer
SML	Slice Management Layer
SM	Slice Manager
uRLLC	Ultra-Reliable Low-Latency Communication
VIM	Virtual Infrastructure Manager
VNF	Virtual network Function
VNFM	Virtual network Function Manager
ZSM	Zero-touch network and Service Management

1 Executive summary

This deliverable presents MonB5G final energy efficiency techniques which cover Task 5.3. It starts by highlighting the energy management and orchestration architectural building blocks. They are divided between inter-domain management and orchestration (**IDMO**) and domain-specific **DMO** which can dynamically deploy management functions that interact with Infrastructure Domain Manager (IDM) to achieve **energy-saving** and **cost-effective** infrastructure management. Specifically, one of the most appealing use cases is the possibility to deploy additional infrastructure management and orchestration functions (IOMF) for resource consumption estimations and resource groupings to maximize resource utilization and **achieve energy-saving goals**.

On the other hand, AI-driven zero-touch closed-loop management is adopted. It is based on the three administrative elements of monitoring system (MS), analytic engine (AE) and decision engine (DE), wherein feedback interfaces are leveraged to reconfigure MS, AE and DE to fulfill energy efficiency and scalability objectives along with network automation and service management. In this regard, MS, AE, and DE are instantiated at each technological domain and for each slice. This allows for several energy-aware **decentralized sustainable artificial intelligence (AI) techniques** to achieve MonB5G energy-efficiency vision in MS, AE, and DE. They rely on a key design principle, wherein the energy cost for running multiple **distributed** local computation tasks is much lower than transmitting raw monitoring data. This is justified by the high transmit power over, e.g., fiber transport links compared with the cloud central processing unit (CPU) power consumption that depends on the product of its extremely small capacitance (in the order of octillions) and its number of cycles required for computing one data sample.

Categorically, the **MS** has been designed in such a way to minimize the measurement load by adding an internal memory called *common online memory store* (COMS) and using the concept of *sampling loops* to collect monitoring data. Moreover, to reduce the transmission overhead and thereby the underlying energy consumption, a cross-domain constrained federated learning (FL)-based **AE** is introduced which makes the analysis and prediction task more than **x10 energy-efficient** by dramatically reducing the amount of raw data exchanged between local AEs and the end-to-end AE and resulting in more scalability to support a massive number of concurrent slices. Based on slice traffic analysis. Finally, distributed cross-domain multi-agent Deep Reinforcement Learning (DRL)-based **DE**s are considered to perform cross-domain joint slice VNF placement and energy control by incorporating the energy cost into the DE multi-objective reward function.

The deliverable covers the following aspects:

- MonB5G reference architecture for energy management,
- AI/ML-driven energy management through MS/AE/DE,
- MonB5G energy-efficient techniques with cross-domain Decentralized Energy Efficient DE,
- Energy efficiency at RAN and Edge with cross-domain statistical FL-based decentralized AEs,
- Stochastic FL-based policy for scalable AE and DE energy-aware techniques.

2 Introduction

2.1 Scope of the deliverable

MonB5G aims at the design of a novel zero-touch management and orchestration (MANO) framework by deeply leveraging the distribution of operations together with beyond state-of-the-art AI algorithms. In this respect, MonB5G develops a hierarchical, automated, and data-driven network management system that incorporates energy efficiency as a key feature to orchestrate a high number of parallel network slices in a sustainable way.

This deliverable provides the final report of MonB5G contributions for a zero-touch distributed x10 energy-efficient MANO compared to a centralized approach. First, the deliverable introduces the MonB5G reference architecture for energy management. Then it details the role of the three administrative elements---MS, AE, and DE---in the AI-driven energy-efficiency. In addition, cross-domain energy-aware AEs and DEs are presented, which range from scalable constrained federated learning resource prediction/allocation to multi-agent deep reinforcement learning intelligent slice reconfiguration.

2.2 Structure of the deliverable

The deliverable covers the following energy efficiency aspects:

Subsection	Description	Domain
3.1	MonB5G reference architecture for energy management,	Cross-Domain
3.2	AI/ML-driven energy management through MS/AE/DE	Cross-Domain
4 .1	MonB5G energy-efficient techniques Decentralized Cross-domain Energy Efficient DE	Cross-Domain
4.2.1	Statistical FL-Based decentralized AE	RAN and Edge
4.2.2	Stochastic FL-based policy for scalable AE	RAN and Edge
4.3	DE energy-aware technique	RAN and Cloud

3 MonB5G reference architecture for energy management

3.1 Energy management architecture and components

Energy efficiency has always been a significant aspect of network development, driven by both economic and environmental considerations. Although B5G networks support many new and existing energy-saving features that network operators can utilize to reduce network energy consumption, the optimization of resources consumption and energy usage for sustainable mobile networks is a challenging subject.

5G-era networks will be much more energy efficient on a per-bit basis than 4G. However, they will transmit significantly more bits due to **mobile data traffic growth**, as well as the expanding variety and increasing complexity of 5G applications, which will impose additional demands through a greater number of cell sites powered by energy-hungry antennas. Consequently, 5G network operators could face up to 2-3 times higher energy costs versus 4G [1].

By introducing greater levels of virtualization, automation, and software-defined networking (SDN), 5G promises to significantly reduce OPEX. A 5G network and beyond that is built on a software-driven, orchestrated architecture will reduce the OPEX per site because of a more flexible allocation of resources [2].

Energy is a significant B5G network cost. Consequently, energy management is one of the largest cost optimization targets for all network vendors and operators. In B5G mobile networks, network virtualization, automation, and artificial intelligence (AI) will be the key tools for energy consumption reduction. AI capabilities can be used at different layers of the 5G Network and will improve the operators' understanding of the energy perspective and identify the root causes of inefficiencies. ML-based techniques make it possible to reduce energy consumption in all network elements without impacting QoE. A data-driven AI-based energy-saving solution on RAN, predicts low traffic periods and shuts down resources at exactly the right time, and reduces the waste of the energy during the peak traffic hours.

Network slices are simply segments of virtual computing and connectivity resources that are configured and provisioned for specific services based on their requirements and features. Network slices have a wide range of requirements in terms of resources, quality objectives, and lifetime. Slice deployment algorithms that are efficient are crucial for lowering network operator costs and energy usage while also offering better service to users. The performance of cloud computing and network slicing depends on the efficient allocation of virtual resources and the optimal placement of Virtualized Network Functions (VNFs) composing these [3].

In addition to the effective allocation of network resources and the satisfaction of user demands, the performance and scalability of the orchestration, which is responsible for the deployment, modification, and termination of network slices, is incredibly important to slice tenants and network operators. A robust OSS and BSS with automated business and operational processes is required to efficiently manage network slices and enable OPEX savings. The most important network functions can be dynamically created, quickly deployed, and managed automatically throughout the entire service lifecycle using programmable and flexible 5G networks, advanced AI (Artificial Intelligence), and Service Level Agreement (SLA) based orchestration. The goal of automated deployment methods is to schedule hardware resources optimally and quickly provision network services to fulfill SLAs while maximizing system utilization. This lowers the costs and energy consumption of the network.

One of the main targets addressed by MonB5G is to reduce energy usage. MonB5G aims to offer all required techniques to enable energy consumption reduction. Numerous energy-aware artificial intelligence (AI) solutions are proposed, which are built on the MonB5G distributed network slicing architecture.

The MonB5G architecture intends to provide a new framework for scalable and automated management and orchestration of network slices on a massive scale. The proposed MonB5G framework's core features align with a number of ETSI ZSM standards that have already been specified. The MonB5G architecture consists of static and dynamic components that support procedures for self-healing, self-configuration, self-optimization (including energy efficiency), and security of network slices.

MonB5G divides the management system into a number of management subsystems, by distributing monitoring, analysis, and decision-making processes across multiple technological domains and components. AI is used at numerous levels to achieve specific management objectives and to reduce interactions across architectural components, through hierarchical closed-loop controls. The implementation of AI/ML algorithms across all technological domains (e.g., RAN, cloud, edge, core) enables resource allocation optimization, sustainable deployment, and operation, among other objectives, and creates a highly adaptive, scalable, and **energy-efficient** network.

In MonB5G the management and orchestration functions are divided between inter-domain management and orchestration (**IDMO**) and Domain specific **DMO**(s) management and orchestration. IDMO is a centralized element with extensive slice management and orchestration decision capabilities. DMO can be seen as a combination of resource-oriented Operations Support Systems (OSS) / Business Support Systems (BSS) and an orchestrator. Domain Manager and Orchestrators (DMOs) can dynamically deploy management functions that interact with IDM to achieve **energy-saving** and **cost-effective** infrastructure management. The framework introduces the **MonB5G portal** that allows Slice Tenants (infrastructure providers, orchestrator operators, template providers, and VNF providers) to request slice deployment based on their chosen slice templates. It provides communication between tenants and system operators for network slice service exposure, negotiation, ordering, and LCM.

NFV allows the separation of communication services from dedicated hardware. This separation allows network services, known as Virtualized Network Functions (VNFs), to be hosted on existing hardware, which simplifies and improves service deployment and management for providers, increases flexibility, and leads to more efficient and scalable resource utilization and lower costs. One of the most difficult technical challenges is ensuring that VNFs are properly placed in hosting infrastructures. This placement significantly influences the network's performance, reliability, and operating costs. Likewise, the performance of network slices' orchestration processes is crucial for efficient utilization of resources during slice lifetime and for slice provisioning in a timeframe accepted by verticals (slice tenants), which is also a key feature in the context of energy savings. Long slice termination or deployment time contributes to insufficient resource utilization and can cause excessive energy loss [4].

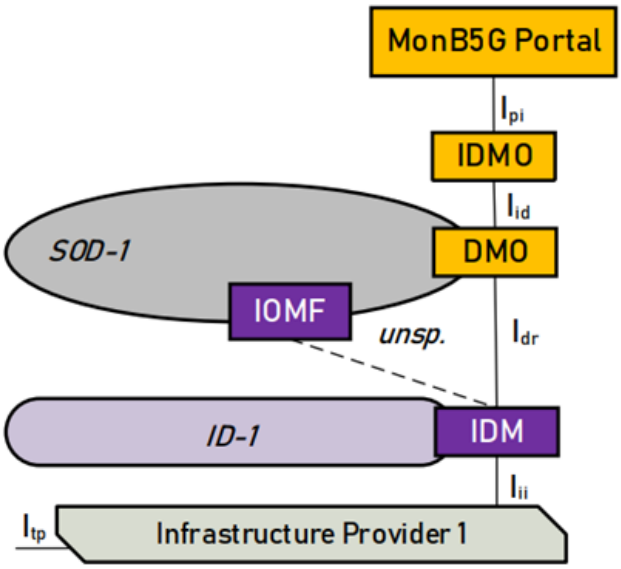


Figure 1. Orchestrated management functions of Infrastructure Provider

As a result, the suggested architecture implies that the infrastructure also needs management, and this management will benefit from the Infrastructure's programmability. To that aim, we have proposed a new management entity named Infrastructure Domain Manager (**IDM),** which can be seen in Figure 1. IDM is responsible for the infrastructure management of specific orchestration domains. NFVI Agent, Energy Consumption Agent, Resource Brokering Support, Infrastructure Operator Portal, and Infrastructure-oriented OSS and BSS should be included in the minimal implementation of IDM in the MANO case.

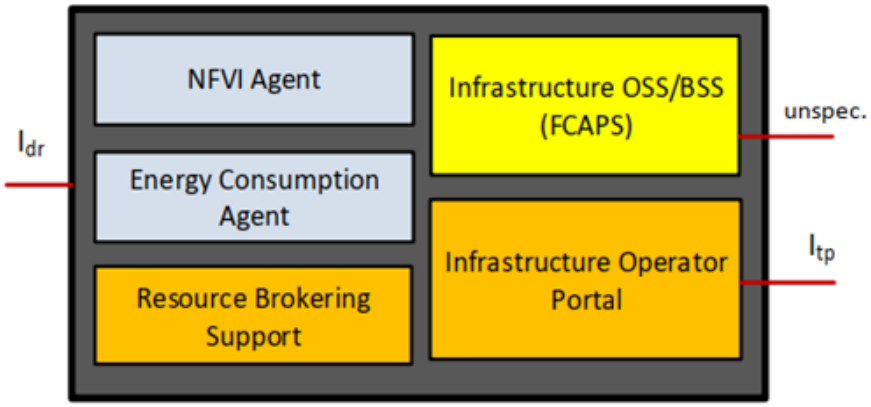


Figure 2. Minimal implementation of IDM

The Infrastructure Provider can use the MonB5G portal to request the deployment of additional infrastructure orchestrated management functions known as **IOMFs** through IDM's interface with the Infrastructure Provider. The IOMF functions can perform a wide range of functionalities that can optimize infrastructure utilization efficiency and achieve effective infrastructure management. The infrastructure management system can deploy its services using the MonB5G mechanisms similarly to how slices are deployed. The framework is **aware of the energy costs** associated with infrastructure resources and can optimize them. Utilizing IOMF for resource consumption estimations and resource groupings to maximize resource utilization and **achieve energy-saving goals** is one of the most promising use cases. The IOMF functions are orchestrated in a way that is consequently, they must be customized specifically for each IDM type [5].

The deployment of multiple network slices requires the network to reserve enough resources for each specific slice instance according to its requirements. In terms of management of Network Slices, we use the concept of In-Slice Management, (ISM). In-Slice Management involves embedding a specialized management function (in the form of a VNF) into slices, which exposes interfaces to both the Orchestrator and the slice owner, enabling the latter to implement its own slice instance management operations. [6].

IDMO, DMO(s), and ISM(s) are utilized with a closed-control loop that adds intelligence to the management and orchestration functions through AI-based capabilities to reach zero-touch management objectives. The proposed closed-control loops include the necessary mechanisms and algorithms mainly relying on AI/ML to assist IDMO, DMO(s), and ISM(s) to achieve self-management, self-configuration, self-adaptation, and performance optimization including energy efficiency. As a result of the implementation of the in-slice management, the central OSS/BSS is slice-agnostic and the orchestrator is primarily focused on resource-oriented operations. The In-Slice Manager can request the modification of a slice or new resource allocation by the interaction with the orchestrator. This method increases the orchestrator's scalability and simplifies the integration of a slice. Furthermore, the slice management is programmable with the aid of AI techniques and KPI prediction.

IDMO is analogous to 3GPP Network Slice Management Function (NSMF) [7] and manages the LCM of end-to-end network slices. It provides complete slice management and orchestration decision capabilities and performs global actions for network-wide, slice-to-slice, and domain-to-domain optimizations.

The DMO of a Slice Orchestration Domain (SOD) is in charge of orchestrating slices in the domain and managing the domain's resources. DMO can be viewed as a combination of resource-oriented OSS/BSS and MANO orchestrator in the NFV MANO-orchestrated domain. Slice admission, LCM, and resource sharing are the most common operations of DMO. The OSS/BSS component of DMO is an AI-driven resource management platform. Through a distributed messaging network, it communicates with IDMO, slices, and infrastructure.

The dynamic components of the architecture are slices that contribute to the overall management of the MonB5G platform. The MonB5G slice template is composed of two different layers that can be orchestrated individually for improved flexibility, efficiency, and scalability, the Slice MonB5G Layer (SML) and the Slice Functional Layer (SFL). SML can be regarded as an embedded OSS/BSS with AI-based MAPE management at the slice level. The concept of in-slice management with embedded intelligence results in self-managed slices and **reduces the amount of data exchanged between the slices and the architecture's external management components**. Slice Functional Layer (SFL) consists of a set of virtual functions that compose the network slice to be deployed and offers the "core" functionality of the slice.

The SML is further divided into four different sublayers, the Monitoring System (MS), Analytic Engines (AEs), and Decision Engines (DE), as well as the Actuators (ACTs) sublayers. The Monitoring Subsystem Sublayer (MS Sublayer) is responsible for collecting, and processing of critical information on the system's operation with certain granularities. The Analytic Engines Sublayer (AE Sublayer) is composed of multiple Analytic Engines that use the information provided by the MS Sublayer to detect and react to various events, related to FCAPS operations (Fault, Configuration, Accounting, Performance, and Security). The Decision Engines Sublayer (DE Sublayer) is composed of multiple Decision engines that are responsible for data-driven decision-making. The Actuators Sublayer (ACT Sublayer, ACT-S) is responsible for converting the decisions of DEs into multiple atomic reconfiguration-related operations that simplify the reconfiguration and reduces the traffic between DE and reconfigured node(s).

MS/AE and DE form the closed-control loop of the ETS ZSM reference architecture [8] and assist IDMO, DMOs, and ISM in enhancing resource allocation and network sustainability. All the components of the architecture are using interfaces between them for the management of the network slices. The role of the defined feedback interfaces between DE and AE, DE and MS, and AE and MS are to reconfigure MS and AE to fulfill energy efficiency and scalability objectives along with network automation and service management.

In the following section, we provide more details about the role of the triplet MS/AE/DE in the energy and infrastructure management of 5G Network and beyond.

3.2 AI/ML-driven energy management through MS/AE/DE

Adoption of a highly scalable, automated infrastructure that can be adjusted on demand is required for energy optimization and operational expenditure minimization of 5G networks and beyond. The use of monitoring, analysis, real-time infrastructure provisioning, dynamic orchestration, and data-driven decisions provides a zero-touch energy efficient network.

The MonB5G framework provides End-to-End (E2E) service and network management automation across multiple domains and network slices through the development of hierarchical closed-control loops. MonB5G also utilizes the cognitive capabilities provided by ETSI ENI, including AI/ML algorithms, intent policies, and SLA management, to increase the scalability and effectiveness of service delivery and reduce operational expenditures (OPEX).

Monitoring System (MS), Analytical Engine (AE), and Decision Engine (DE) compose the closed-control loop with AI capabilities to accomplish energy efficiency, as well as the objectives of zero-touch management and network automation. Monitoring data are collected, processed, and continuously analyzed to provide information and insights. Using these data, decisions are taken, and the results are delivered back to the network, where performance is continuously monitored.

3.2.1 MONITORING SYSTEM (MS)

MonB5G MS captures the current operation status at multiple levels of the management hierarchy (node, slice, domain, and inter-domain). The distributed MS agents are designed to manage the tightest metric sampling loops in their technological area, hence significantly reducing the requirement for data transfer and minimizing communication overhead generated by the monitoring system itself. The analysis and decision functions are located closer to MS in order to avoid the transfer of raw data. MS periodically transmits monitoring data to AE, which analyses the data and gives the necessary analysis output to the DE.

Monitoring of performance KPIs applicable to different parts of the network infrastructure provides very powerful and holistic information for 5G and beyond Networks. MS monitors KPIs that capture the **OPEX savings** and **reduction in managerial overhead** due to zero-touch slice MANO. The advanced SML with the AI-powered capabilities of AE/DE for enhanced flexibility and resilience, as well as the real-time monitoring of MS enable efficient and dynamic inter-domain resource allocation. The framework is informed of the energy costs of infrastructure resources and can act to optimize them.

The gathered monitoring data can be passed directly to DE and AE or might be kept in a database (internal memory) for later examination such as a Time-Series Database (TSDB) for measurement load minimization. This memory block allows the MS, AE, and DE to avoid implementing energy-intensive synchronization whenever information needs to be updated and transmitted between these blocks. The Common online memory store (COMS) is responsible for the storing and retrieval of shared historical, configuration, and operational data between the MS-AE-DE triplet and other closed network loops. Thus, DE and AE can be more flexible in terms of processing duration without reducing the granularity with which MS can collect monitoring data from the controlled systems.

3.2.2 ANALYTIC ENGINE (AE)

Analytical Engines (AEs) in MonB5G are distributed in the different parts of the management system as the Monitoring System. Using Federated-Learning, AE can significantly make data analysis and prediction more energy efficient in real time by substantially reducing the quantity of raw data exchanged between local AEs and the E2E AE. Federated Learning is also used for decentralized resource estimation to maintain low SLA violations. This innovative function of AE enables decentralized resource allocation in network slices while ensuring extremely low SLA violation rates.

The suggested Statistical FL was moved closer to the network's distributed monitoring nodes in a way that greatly decreases the interchange of raw data and transfers only a subset of AI model parameters for coordination or collaboration [9]. Statistical FL can decrease the communication overhead by more than a factor of 10 when compared to centralized constrained deep learning schemes as demonstrated in deliverable D3.1. This overhead is further decreased by a selection policy restricting the number of agents that cooperate throughout the FL task, hence promoting sustainability in a situation involving extensive network slicing.

MonB5G improves AI-based traffic load prediction by introducing context-aware loss into the learning mechanism of AE. Forecasting the volume of traffic is critical for a variety of subsequent activities, including resource allocation and admission control. This AE feature can predict traffic load for all technological domains and includes a wider span of control for resource over- and under-allocation penalties and resource reallocation settings. By ensuring that the appropriate quantity of resources is supplied to a network slice at the appropriate time, it decreases the probability of SLA violations and ensures QoS/QoE.

3.2.3 DECISION ENGINE (DE)

Network slicing, which allocates network resources according to users' specific requirements, is a key feature to fulfill the diversity of service requirements in 5G and beyond. The operator must decide in advance how much resources must be allocated to the various slices to ensure that the capacity is used as effectively as possible. During the decision-making process, energy efficiency and optimal infrastructure utilization must be considered apart from network performance KPIs.

Decision Engine (DE) is one of the core components of the MonB5G architecture, and the closed-control loop is designed to achieve zero-touch administration of a large number of network slices in Beyond 5G networks. Admission control, intra-slice orchestration, and inter-slice orchestration are the three main management tasks of the network slice life cycle process that we have addressed using of DEs. During the lifecycle of network slices, the Decision Engine (DE) regulates the decision-making process of the closed-control loop, by relying heavily on artificial intelligence (AI) and the outputs of the Analytical Engine (AE) from data processing [10]. AI can play a significant role in zero-touch networks that reduce complexity by managing the vast amount of network and service parameters. Recent AI innovations have demonstrated that more accurate decision models may be constructed using decentralized architectures such as Multi-Agent DRL and Federated DRL [11][12].

Within the framework of the MonB5G project, we suggested various deep reinforcement learning (DRL) schemes for network slice-related problems that can perform energy-aware configuration decisions. RL can be trained to maximize the energy efficiency in the network by placing VNFs on the same physical machine, when possible, which will reduce the quantity of VNF migrations. According to traffic conditions, RL is used to decide when to scale up or down, instantiate, terminate, or even duplicate a specific VNF, VNF chain, or slice, and to switch off the servers. All layers of the MonB5G architecture employ AI/ML models for network slicing that are scalable and sustainable.

DEs use Decentralized Deep Reinforcement Learning strategies (e.g., Multi-Agent DRL, Federated DRL) to perform cross-domain energy-aware VNF and SFC placement in 5G service-customized network slices. The energy cost is integrated into the DE multi-objective reward function, along with latency. DEs choose the ideal compromise solution to achieve a balance between energy efficiency and SLAs. Distributed AI enables the local processing of management information, consequently decreasing the exchange of management information between entities. Local DE can make AI-driven decisions based on the intelligence obtained from a local analytic engine. Moreover, cross-domain re-configuration is performed when the assignment becomes too complex, and the local information is insufficient.

In order to accomplish the goal of end-to-end DE, the distributed AI-driven solutions at several levels of the management hierarchy (node, slice, domain, and inter-domain) leverage communication interfaces across domains permitting resource brokering and energy-efficient operations.

DE leverages DMO-exposed APIs to implement the chosen decisions. DE is required to interact with IDMO in order to make global decisions, whereas for local decisions, it communicates with DMO and ISM. Cross-domain operations between local DEs (i.e., DEs of each technological domain) or with end-to-end DEs are managed by the IDMO, whereas inter-slice DE operations are managed by the DMO.

4 MonB5G energy-efficient techniques

4.1 Decentralized Cross-Domain Energy Efficient DE

The decentralized Energy-Efficient DE utilizes the distributed notion of network domains to operate locally. This action enables parallelism and avoids unnecessary VM migrations between the domains or costly re-orchestration of the entire slice. An entity disconnected from the decision function, called the Auction Mechanism, is introduced to enable inter-domain VNF migration. It operates with multiple agents, eliminating a centralized point of failure as the Auction Mechanism module can be instantiated anywhere in the network.

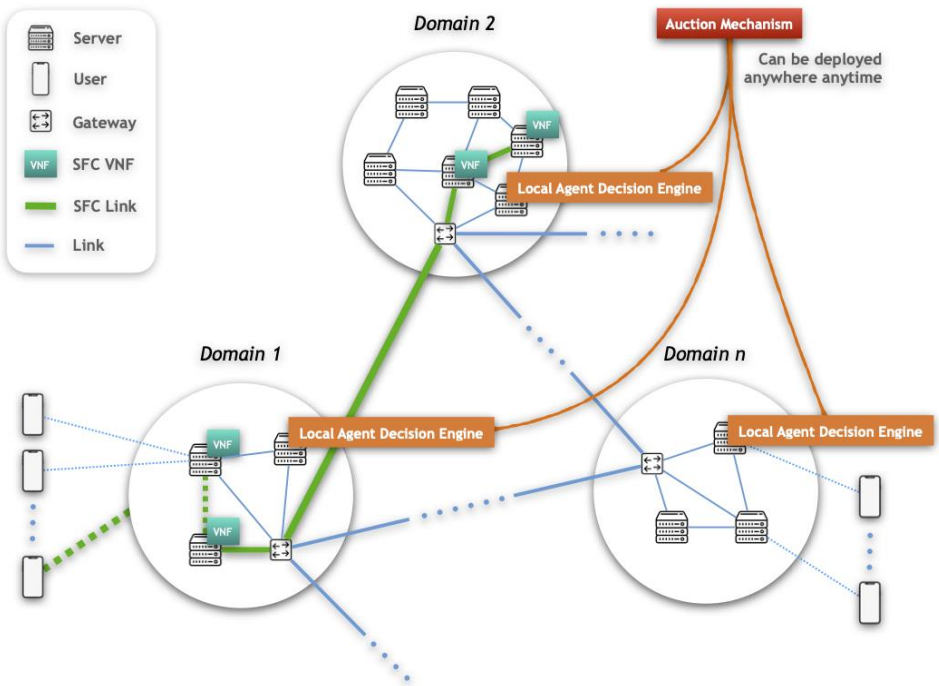


Figure 3. Slice graph DE deployment example

The Energy Consumption Model

The total energy consumed by the network during the operation of the slice branches into two distinct segments.

First, the energy consumed by the utilization of computational resources E_m while hosting the VM m on a network node is calculated based on the work of Mao, et. al in [10] as defined below:

$$E_m = \phi_s^{cpu} \mu C D_s F^2,$$

where F indicates the computational capacity of the node measured in CPU cycles per second, C stands for the CPU cycles required for computing one data sample at each CPU core, ϕ_s^{cpu} is the number of utilized

CPU cores by the VM and D_s represents the amount of processed data expressed in bits. The constant μ expresses the effective switched capacitance of the CPU architecture.

Second, the link energy consumption E_{uv} can be calculated as the data sent between the VMs hosted in servers u and v, divided by the data rate and multiplied by the power of the link transmitted optical power ρ_{uv}. The transmission energy is calculated as follows:

$$E_{uv} = \rho_{uv} \frac{t_u}{R_u},$$

where the variable R_u expresses the transmission data rate of a network node u in Gigabits per second, ρ_{uv} is the optical transmit power of link uv in dBm and t_u stands for the amount of transmitted data of a server expressed in bits.

The total slice energy consumption E_s that we attempt to optimize for the operation of the slice in the current iteration is defined as follows:

$$E_s = \sum_{uv \in E} + \sum_{u_s v_s \in E_s} + E_{uv} z_{uv}^{u_s v_s} + \sum_{m \in M} + \sum_{u_s \in V_s} + E_m z_m^{u_s}.$$

The Auction Mechanism Module

We also introduce the Auction Mechanism, a system that enables inter-domain VM migration in a distributed multi-domain network. As shown in figure 4, the Auction Mechanism enables scalability and parallel operation.

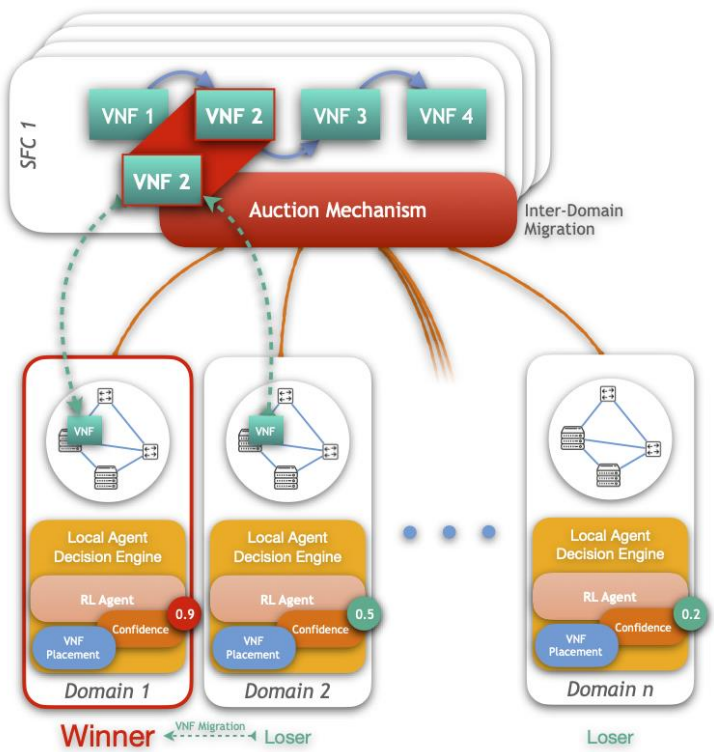


Figure 4. Overview of the multi-domain and distributed Auction Mechanism.

The operation of the Auction Mechanism can be described in the following steps:

1. **Auction Initiation:** The Auction Mechanism chooses the next slice VM and initiates an auction with the distributed domains.
2. **Distributed Operation:** The distributed RL agents of the domains generate their local action and the corresponding Confidence Vector with respect to the local placement for the VM in auction. The Confidence Metric of each domain is then sent to the Auction Mechanism, ensuring minimum data transfers.
3. **Global Operation:** The Auction Mechanism receives the Confidence Metric of each domain and selects the highest bidder or the domain with the maximum Confidence Metric as a candidate to receive the VM currently in auction. The Auction Mechanism notifies the candidate domain with an acknowledgment.
4. **Orchestration:** If the candidate domain is different from the current domain that hosts the VM in the auction, the inter-domain migration is initiated. Otherwise, the domain agent performs an intra-domain migration to the node with the highest Confidence Metric of the local Confidence Vector with a much lower cost in terms of both energy, time and overall cost. If the VM is already instantiated in the same node, the procedure of migration is declined.
5. **Iteration:** The procedure is repeated indefinitely.

It is apparent that the Auction Mechanism acts as the *auctioneer* and it is only responsible for the inter-domain communication making it non-essential for the local domain orchestration. The Auction Mechanism can be deployed quickly at any node of the network eliminating the single point of failure in the system.

Results Evaluation

In this section, we conduct a simulation study with a diverse variety of scenarios on a realistic multi-domain network to prove the performance superiority of the proposed scheme.

The performance of the proposed SCHE2MA solution is compared with two references from the literature scenarios:

- **Centralized RL:** An RL-based orchestration algorithm located in a central location, which is a common baseline in the related research literature, such as in [11], [12], and [13]. The central orchestration algorithm overlooks the entire network as opposed to our proposed distributed orchestration scheme, where the VNFs are serially placed and the VNFs are migrated to the node with the highest action value.
- **Static Placement:** A typical VNF placement strategy, which is adopted by many providers [14]. In this strategy the VNF placement is static and the VNFs remain hosted in the initial node throughout the experiment.

Results Analysis

The performance of the baseline scenarios is normalized to the SCHE2MA performance, and the plots show the relative gain or loss for each metric. The analysis shows the performance of SCHE2MA in both average energy consumption and average service latency. The energy consumption curves of all figures are

normalized based on the SCHE2MA performance to improve legibility. The values are expressed in millijoules (mJ) under the curve of SCHE2MA.

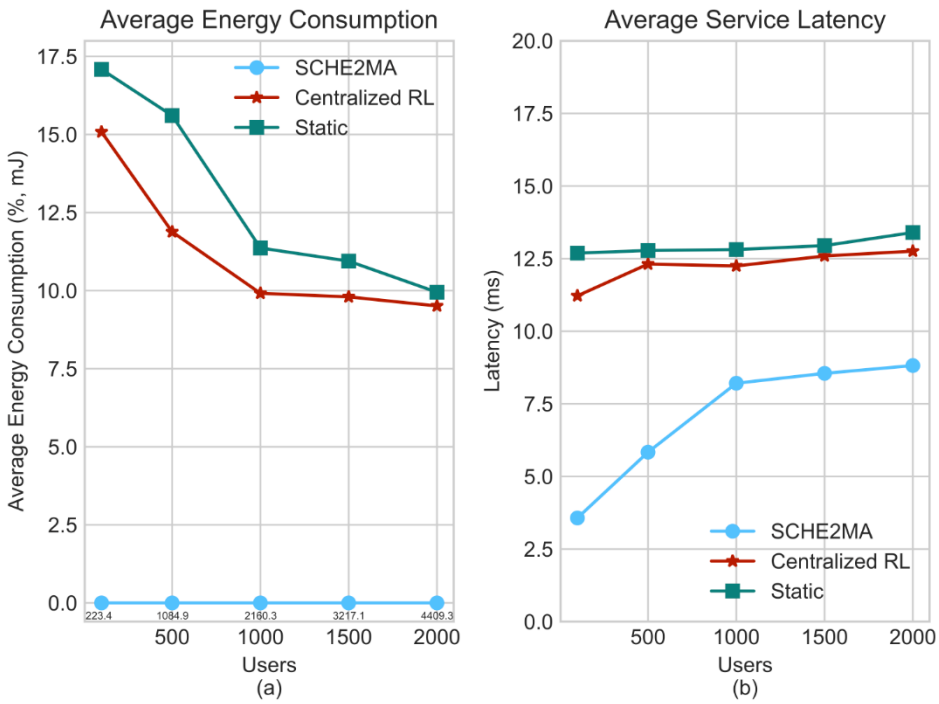


Figure 5. (a) Average energy consumption improvement expressed in percentages, compared to the proposed solution. (b) Average service latency per number of users in multiple traffic scenarios. Lower is better for both figures.

In Figure 5(a), we depict the average energy consumption of the examined network of 500 simulations for a varying number of users, that is normalized based on the SCHE2MA performance (%,mJ). We observe that the energy consumption increases almost linearly with the number of users due to the massive number of transmissions. The reason is that introducing more users to the network generates additional requests that consume more energy during each transmission. Therefore, the overall energy consumption of the network is higher. The proposed solution can maintain lower energy consumption in all scenarios, reaching almost 17.1% reduction in the case of 100 users. The reason for this behaviour is the ability of SCHE2MA to cluster VNFs into the servers, minimizing the costly communication between servers.

Figure 5(b) presents the performance of the most critical metric in URLLC services, the average service latency. We observe that the average service latency increases due to insufficient computing resources in servers within the domains as the number of active users grow. However, it has to be noted that SCHE2MA outperforms both baselines by offering a 103.4% reduction in latency for the case of 100 users without increasing the energy consumption, which is a considerable performance improvement while also maintaining lower energy consumption than both baselines. That is possible due to VNF clustering in servers,

which minimizes the number of transmissions in physical media. SCHE2MA demonstrates a clear indication of its ability to conceive better VNF placements that satisfy the latency and energy consumption trade-off.

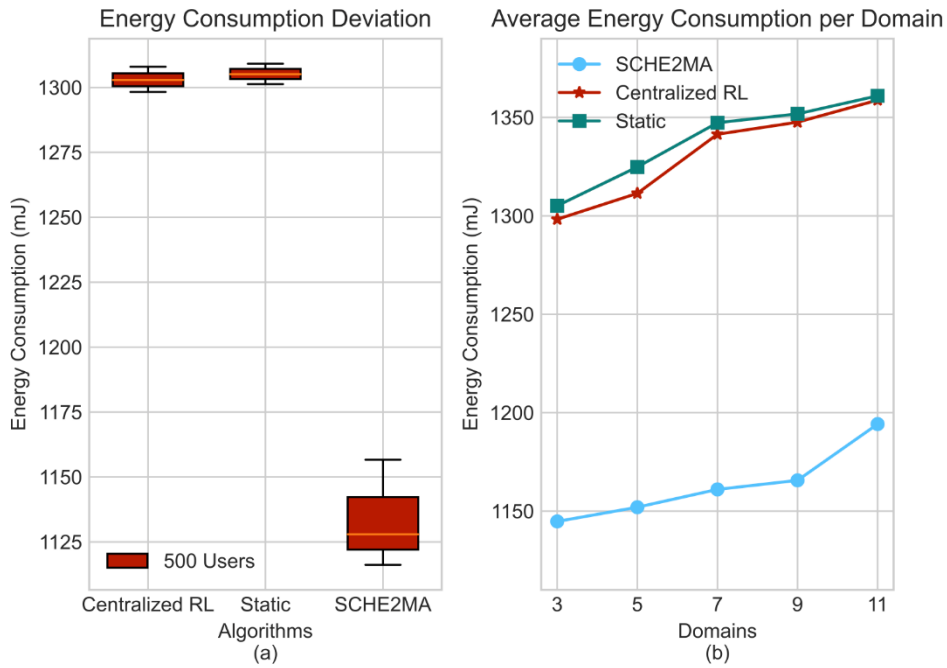


Figure 6. (a) Energy consumption deviation in a 3-domain network with 500 users and 25 SFCs. (b) Average energy consumption in multi-domain network configurations for 500 users and 25 SFCs. Lower is better for both figures.

Figure 6(a) presents how the energy consumption fluctuates during the operation of each algorithm, specifically for the scenario with 3 domains and 500 users in a simulation cycle. We observe that the maximum difference in energy consumption is 15.91% between the Static solution and SCHE2MA. The reason for this is that, as can be seen in Fig. 5a SCHE2MA tends to consolidate multiple SFC VNFs in hosts *id-est* hosts 2 and 5, to minimize both energy consumption and latency by turning physical link connections into virtual that yield minimal losses. In Figure 6 (b), we plot the average energy consumption per domain to evaluate the scalability of the given solutions. We observe that the energy consumption steadily increases as we introduce more domains into the network, hence increasing the number of data that need to be considered when planning a VNF placement. It can be seen that the Centralized RL fails to converge due to the larger state space. SCHE2MA can reduce the energy consumption by 14.85% compared to the baseline solutions with 500 users. This is due to the flexibility and scalability of SCHE2MA's distributed architecture where the decision-making takes place locally in multi-domain agents that communicate through the Auction Mechanism, dividing and sharing that way the immense problem space.

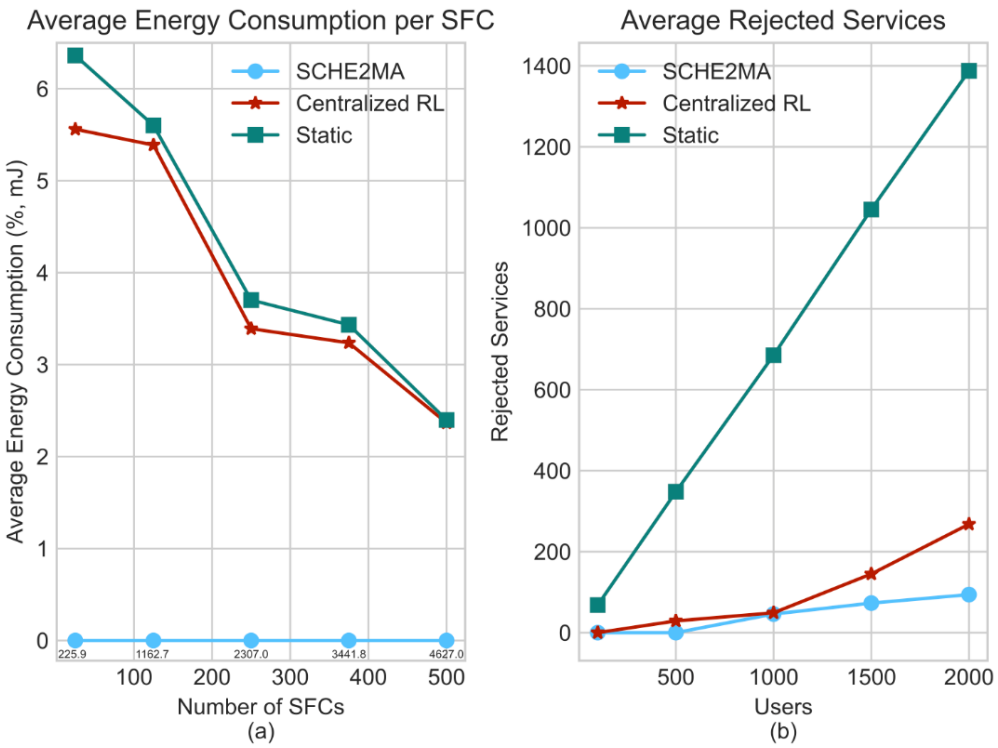


Figure 7. (a) Average energy consumption by the number of SFCs in the network for 500 users. (b) Average number of rejected services for a 3-domain network. Lower is better for both figures.

Figure (a) outlines the average energy consumption per SFC deployed in the network. We observe that in the case of 25 SFCs, the average energy consumption per SFC of SCHE2MA is reduced by 6.36% compared to the Static solution. The reason is that compared to the baseline scenarios, SCHE2MA is capable of operating with less energy, as we have previously discussed and analysed in Figure 5. Figure (b) illustrates the average number of rejected services in a 3-domain scenario with a varying number of users. When the number of users increases in a network with finite resources, the number of rejected services increases. Given that the SCHE2MA can re-organize the VNFs, several

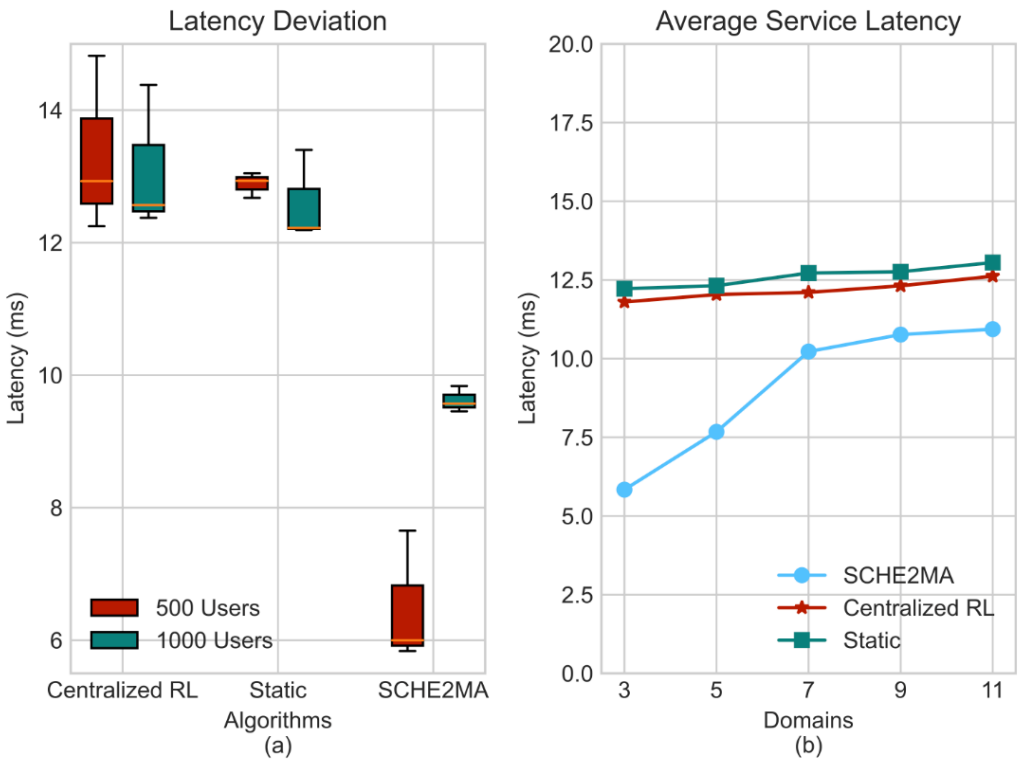


Figure 8. (a) Latency deviation per algorithm for both 500 and 1000 user traffic scenarios in a 3-domain network. (b) Average service latency of 500 users in a 3-domain network. Lower is better for both figures.

Figure 8 illustrates how the service latency oscillates during the operation of each algorithm for the scenarios with 500 and 1000 users. We observe that SCHE2MA can achieve 73.52% less service latency than the baseline scenarios in the case of 5 domains, depicted in the right-hand Figure. This is possible by devising VNF placements that minimize the number of transmissions through local intra-domain orchestration. The Centralized RL is hugely affected by the number of users, as the deviation in the figure suggests.

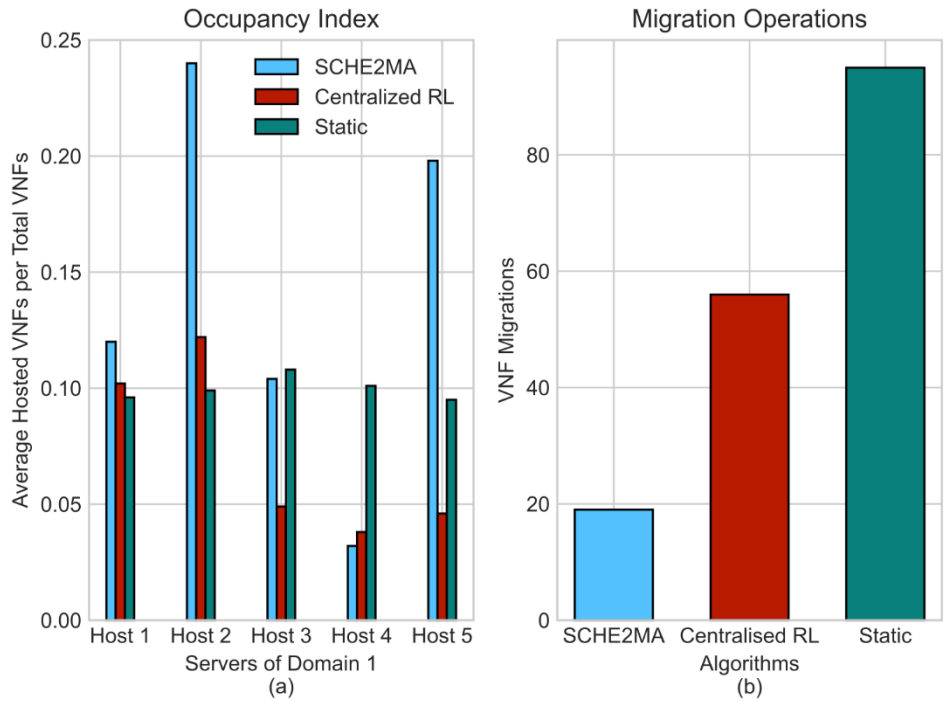


Figure 7. (a) VNF occupancy index represents the average number of hosted VNFs per total number of VNFs for 100 iterations of simulated traffic with 500 users. (b) Average migration operations of 100 iterations of simulated traffic with 50 SFCs and 125 VNFs.

Finally, Figure 9a shows the average number of hosted VNFs divided by the number of total VNFs to indicate the occupancy of the hosts of the first domain. We can see that SCHE2MA gravitated towards consolidating the SFC VNFs to reduce the number of hops to the end-user. Additionally, Figure 9b illustrates the total number of migrations of the local agent originally depicted in Fig. 5a that was applying an identical placement for a sustained period to avoid inter-domain SFC re-configurations and additional data transmissions that lead to higher energy consumption and latency.

4.2 Energy efficiency at RAN and Edge

4.2.1 ENERGY-EFFICIENT STATISTICAL FL-BASED DECENTRALIZED AES

To ensure energy efficiency at e.g., the edge, resource analysis for network slicing can leverage advanced federated learning techniques to build decentralized analytic engines. In this regard, Figure 8 depicts a decentralized architecture consisting of K slice/node levels MS/AEs with a B5G/6G tailored CU-DU functional split per slice.} Each CU $\mathrm{k}\,(\mathrm{k} = 1, \ldots, K)$ runs as a VNF on top of commodity hardware located at the edge cloud and performs slice-level RAN KPI data collection via a local MS as well as implements AI-enabled slice resource analytics through a local AE. To strengthen the analysis, the AE instances of each slice participate in an FL task by sharing their local models with the end-to-end AE that plays the role of an aggregation server.

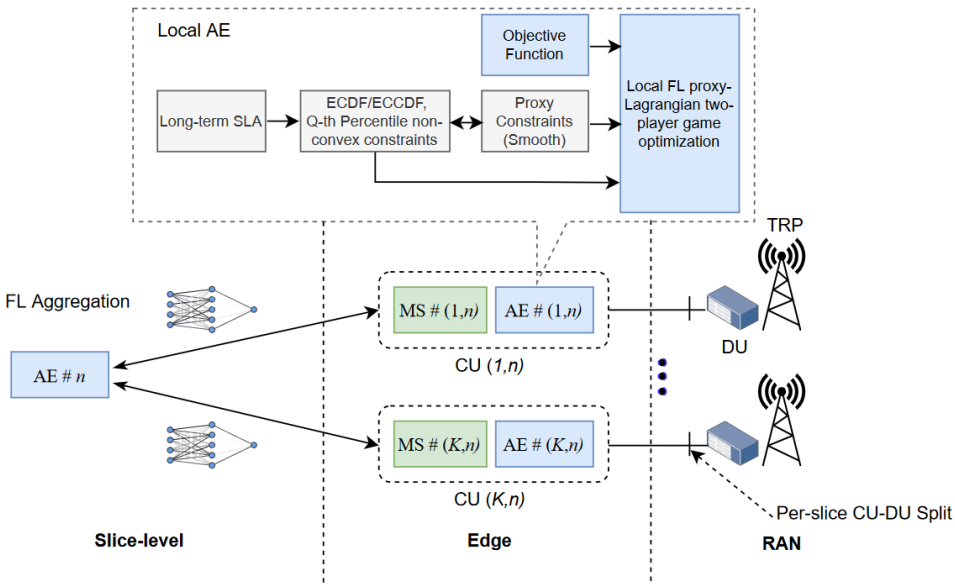


Figure 8. Decentralized architecture.

A typical SLA between slice *n* tenant and the infrastructure provider would state that any assigned resource to the **tenant should not exceed a range $[\alpha_n, \beta_n]$ with a probability higher than an agreed threshold γ_n**. This translates into learning the AE local resource prediction model under empirical cumulative density function (ECDF) and complementary ECDF (ECCDF) constraints that reflect the upper and lower bounds violation rates. The operator and slice tenant may also agree that, e.g., the *Q*-th percentile of a specific resource, i.e., the value below which *Q*% of the samples of this resource are distributed, must be lower than π_n, to ensure isolation. In both SLAs, the FL local task must capture the challenging long-term statistical behaviour of the target KPIs, especially that the learning is performed over offline datasets, which makes the constraints also data-dependent. Therefore, we call this novel AI scheme Statistical Federated Learning (StFL).

Unfortunately, the ECDF/ECCDF statistical measures are defined as an average sum of indicator functions that are non-convex and non-differentiable. On the other hand, the *Q*-th percentile is also non-smooth. Instead of optimizing the local FL problem with respect to their convexified surrogates only as mostly done in the literature (using e.g., the convex-concave procedure), we jointly consider both the original and surrogates by formulating the local FL problem via the so-called proxy Lagrangian framework [15], where we jointly optimize over two Lagrangians. The first, $\mathcal{L}_1$, is containing the loss function (between the predicted KPI and the observation) and a smooth approximation of the statistical measures called proxy constraints. Specifically, the ECDF/ECCDF constraints might see their indicator functions replaced with smooth Logistic functions, and the *Q*-th percentile approximated by the so-called smoothed empirical percentile. The second Lagrangian, $\mathcal{L}_2$, is composed of the original non-smooth SLA constraints. While optimizing the first Lagrangian with respect to the FL model weights requires differentiating the smooth functions, to differentiate the

second Lagrangian with respect to Lagrange multipliers, we only need to evaluate the original ECDF/ECCDF or the Q -th percentile function. Finally, the joint optimization of the two Lagrangians turns out to be a non-zero-sum two-player game wherein the first player wishes to minimize $\mathcal{L}_1$ and the second player aims at maximizing $\mathcal{L}_2$. This process ends up reaching a nearly-optimal nearly-feasible solution to the original constrained problem. The obtained model weights are then sent back to the end-to-end AE to perform averaging and broadcast the obtained model to all the AEs to start a new round of training until reaching convergence.

To exemplify the StFL performance, we provide numerical results of the ECDF/CCCDF SLA case. In this regard, the considered local *K=200* MSs datasets are non-independent identically distributed (NIID). These datasets of size $\mathrm{D_{k,n}} = 1000$ are randomly sampled from encoded measurement data corresponding to a live LTE-advanced network of size $\mathrm{D} = 21417$ samples. It includes, as input features, the hourly traffics of the main over-the-top (OTT) applications, channel quality indicator (CQI) and MIMO full-rank usage, while the considered supervised output KPIs are the downlink physical resource blocks (PRBs) and the CPU load. Once the slices are defined, the traffic of the underlying OTTs is summed to yield the traffic per slice. To exemplify the general framework of StFL, three main slices are considered:

- **eMBB**: involves NetFlix, Youtube and Facebook Video,
- **Social Media**: includes Facebook, Facebook Messages, Whatsapp and Instagram,
- **Browsing**: encompasses Apple, HTTP and QUIC,

The proposed StFL enables to control the long-term statistical behaviour of the SLA compared to FedAvg baseline. Indeed, as depicted in Figure 9, the FedAvg empirical CPU usage CDF of, e.g., eMBB and Social Media slices are breaching the bounds with high probabilities of about 25% and 15%, respectively. This stems from the fact that the baseline FL model cannot learn statistical properties over an observation interval and operate only at the sample level. However, in the StFL case, the CPU loads achieve a trade-off between dynamic allocation and long-term statistical SLA. In this case, the eMBB and Social Media CPU loads are confined in the imposed bounds with a high probability of 99%.

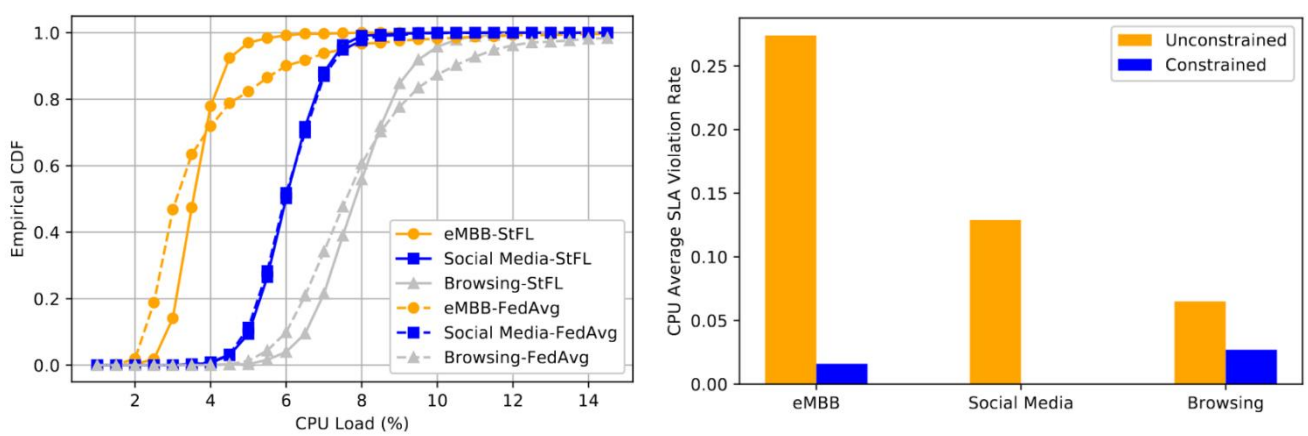


Figure 9. CPU CDF with $\alpha = [0,0,0]$, $\beta = [4,7,10]$ % and $\gamma = [0.01,0.01,0.01]$.

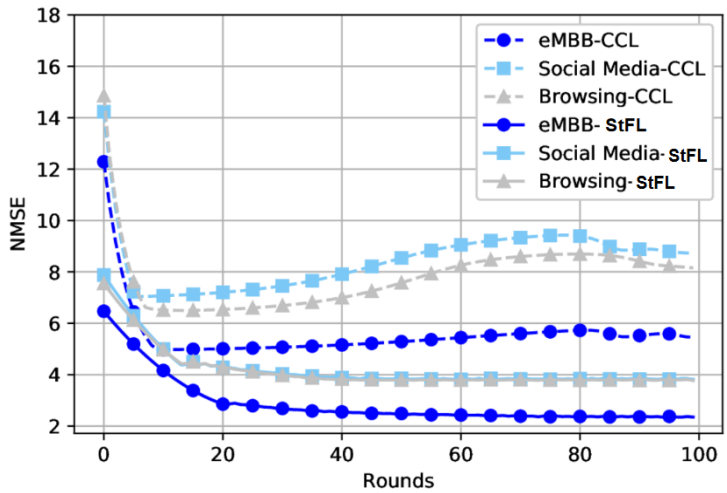


Figure 10. Convergence of the StFL vs the Constrained Centralized Learning (CCL) for SLA $\alpha = [0,0,0]$, $\beta = [15, 10,10]$ *% and* $\gamma = [0.01,0.01,0.01]$

On the other hand, based on the overhead analysis and datasets' sizes coded in 32 bits, the energy consumption is calculated by considering both the local computation energy at each CU [16] as well as the transmission energy over fiber optic transport links [17]. In this respect, Table 1 shows the overhead and energy consumption induced by both baselines fully centralized SLA-constrained deep learning (CCL) [18] and StFL where the samples have been coded in 32 bits. This means that a communication round in the federated setup is equivalent to 100 epochs over a batch in the centralized one. Starting from the convergence point of StFL, i.e., round 50 shown in Figure 10, more than 10 times overhead and energy consumption reductions are obtained at the expense of the short communication delay. Therefore, StFL turns out to be a more efficient scheme, especially when the transmission latency is comparable to the CCL processing delay, while also enabling to dramatically reduce CPU SLA violation rate compared to the FedAvg unconstrained algorithm [19] as showcased by Figure 9.

Table 1. Overhead and energy comparison

Rounds	50	60	70	80
Overhead CCL (KB)		18750		
Overhead StFL (KB)	1055	1266	1477	1688
Energy CCL (mJ)		118.3		
Energy StFL (mJ)	6.7	8	9.3	10.7
Energy Gain	$\times$**17.8**	$\times$**14.8**	$\times$**12.7**	$\times$**11.1**

4.2.2 STOCHASTIC FL-BASED POLICY FOR SCALABLE AE

To further reduce further the network data overhead, optimize the FL computation time and improve the underlying energy efficiency of the system, we can select only a subset of active AEs in each FL round. In this regard, we propose an SLA-driven stochastic AE selection policy. Upon the completion of the training at round *t*, each AE *(k,n)* evaluates the generalization of its FL model using a typical test dataset $\widetilde{\mathcal{D}_n}$ of size $\widetilde{D_n}$, which is common to all monitoring systems of slice *n* and calculates the so-called SLA violation rate as,

$$\nu_{k,n} = \frac{1}{\widetilde{D_n}} \sum_{i=1}^{\widetilde{D_n}} 1\left[\left(\widehat{y_{k,n}^{(i)}} < \alpha_n\right) \bigcup \left(\widehat{y_{k,n}^{(i)}} > \beta_n\right)\right].$$

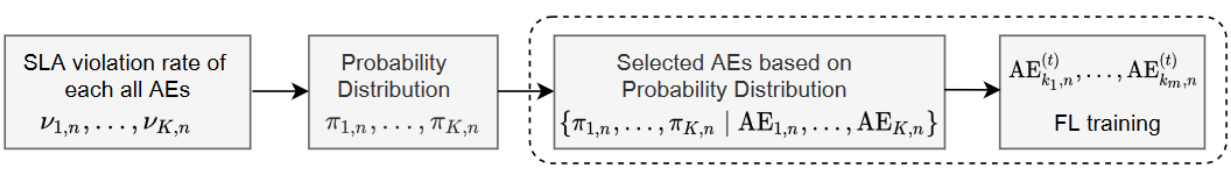


Figure 11. Proposed policy for AE selection

Next, at each FL round, all of the AEs send their SLA violation rates to the server which generates a probability distribution using the *softmin* function as,

$$\pi_{k,n} = \frac{\exp\{-\nu_{k,n}\}}{\sum_{p=1}^{K} \exp\{-\nu_{p,n}\}},$$

wherein AEs with low SLA violation are given a high probability of FL participation to drive the model convergence, but also AEs with high SLA violation may take part in the FL training with a low probability to guarantee the generalization that could stem from their datasets. Based on the probability distribution, only a subset of *m<K* AEs is drawn at each FL. Thus, the AEs would have stochastically participated in the FL task while avoiding the concurrent training at each round. And the model averaging at round *t* is performed as,

$$W_n^{(t+1)} = \sum_{k\in\{k_1,\ldots,k_m\}} \frac{D_{k,n}}{D_n} W_{k,n}^{(t)}.$$

Where D_n is the sum of datasets sizes over all the AEs of slice n. This proposed procedure is summarized in Algorithm 1.

Algorithm 1: SLA-Driven Stochastic Federated Learning Policy.

Input: K, m, η_λ, T, L. # See Table II
parallel for $k = 1, \ldots, K$ **do**
Calculate SLA based violation rate
AE (k, n) calculates $\nu_{k,n}$ according to 4 and reports it to the aggregation server
end parallel for
# Federated Learning
# Server generates probability distribution using Softmin function
for $k = 1, \ldots, K$ **do**
  $\pi_{k,n} = \frac{\exp\{-\nu_{k,n}\}}{\sum_{l=1}^{K} \exp\{-\nu_{l,n}\}}$, $k = 1, \ldots, K$
end
Server initializes $\mathbf{W}_n^{(0)}$ with initial training parameter
for $t = 0, \ldots, T-1$ **do**
  # Server selects the m AEs ID using np.random.choice
  $\mathrm{AE}_{k_1,n}^{(t)}, \ldots, \mathrm{AE}_{k_m,n}^{(t)} \sim \{\pi_{1,n}, \ldots, \pi_{K,n} \mid \mathrm{AE}_{1,n}, \ldots, \mathrm{AE}_{K,n}\}$
  Server broadcasts $\mathbf{W}^{(0)}$ to the m selected AEs
  parallel for $k \in \{k_1, \ldots, k_m\}$ **do**
  # Local epochs
  for $l = 0, \ldots, L-1$ **do**
    Solve the proxy-Lagrangian game between $\mathcal{L}_{\mathbf{W}_{k,n}^{(t)}}$ and $\mathcal{L}_\lambda$ and get $\mathbf{W}_{k,l}$
  end
  return $\mathbf{W}_{k,n}^{(t)} = \mathbf{W}_{k,L-1}$
  Each local AE k sends $\mathbf{W}_{k,n}^{(t)}$ to the aggregation server.
  end parallel for
  # FL Server Aggregation
  return $\mathbf{W}_n^{(t+1)} = \sum_{k\in\{k_1,\ldots,k_m\}} \frac{D_{k,n}}{D_n} \mathbf{W}_{k,n}^{(t)}$
  Broadcasts $\mathbf{W}_n^{(t+1)}$ to all K AEs.
end

4.3 DE energy-aware technique

Consider a C-RAN architecture presented in Figure 12. It is composed of *N* single-antenna small-cells (n=1, …,*N*) connected to a virtual baseband unit (i.e., CUs) pool that runs as a set of VNFs. A total number of *J* VNFs (j = 1, …, *J*) can be deployed on top of the C-RAN datacenter endowed with *I* active central processing units (CPUs), where each processor *i* (*I = 1, …, I*) has a computing capability of P_i million operations per time slot (MOPTS). At each time step *t*, *M* UEs (*m = 1, …, M*) can connect to the *N* small-cells according to the maximum received power criteria. Each UE *m* requests a slice and starts its activity, wherein the packet arrival to the CU VNF follows a Poisson distribution with mean rate $\lambda_m^{(t)}$. In this case, let $\Omega = \sum_{m=1}^{M} \lambda_m^{(t)}$. The mean arrival data rate of all UEs to the CU VNFs is Ω/j, where *j* is the number of active VNFs.

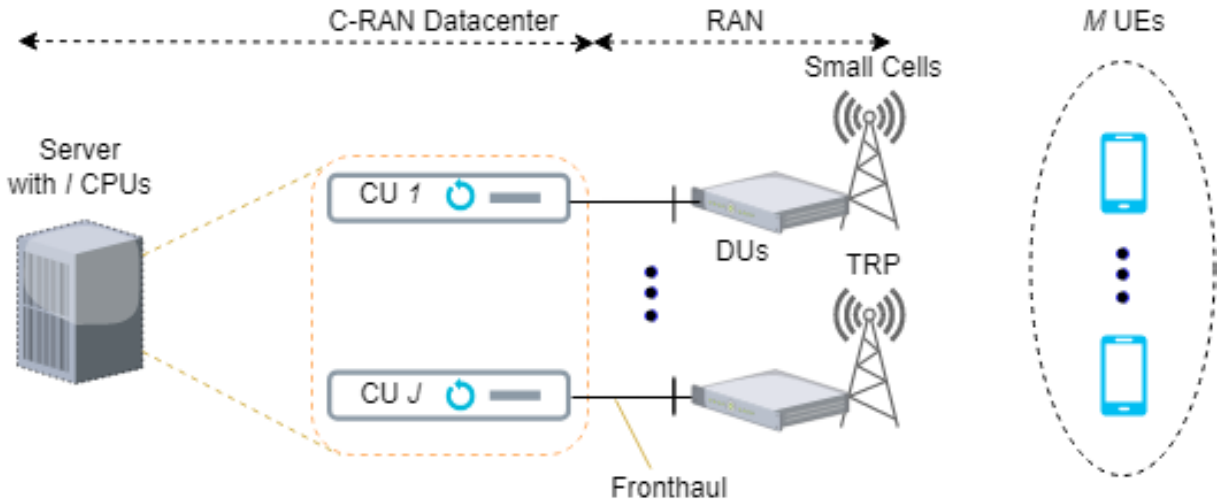


Figure 12. Distributed and hierarchical C-RAN architecture

NETWORK COSTS

We define overall network cost as all costs incurred at each time step *t* as follows:

$$\mathcal{N}_T^{(t)} = \frac{\omega_1^* \mathrm{K}_{\mathrm{Net}}^{(t)} + \omega_2^* \mathcal{L}_{\mathcal{N}et}^{(t)} + \omega_3^* \mathcal{E}_{\mathcal{N}et}^{(t)}}{\mathrm{M}},$$

where $K_{Net}^{(t)}$ denotes the computational cost, $\mathcal{L}_{\mathcal{N}et}^{(t)}$ is the latency and $\mathcal{E}_{\mathcal{N}et}^{(t)}$ represents the energy consumption. The weights ω_1^*, ω_2^* and ω_3^* are fixed weights and are determined based on the network preferences.

Computation cost $(K_{Net}^{(t)})$: The baseband processing procedure at a VNF consists of coding, Fast Fourier Transform (FFT) and modulation. Following the approach in [20], the corresponding computing resource is given by:

$$\mathrm{K}_{\mathrm{Net}}^{(t)} = \sum_{\mathrm{m}=1}^{\mathrm{M}} [\theta \log_2(1 + \delta_{\mathrm{m}})] + \mathrm{MK}_0,$$

where θ is an experimental parameter, δ_m denotes the signal-to-interference-plus-noise ratio (SINR) of UE *m* and K_0 includes computing resources for the FFT function that imposes a constant base processing load on the system.

Latency $\left(\mathcal{L}_{\mathcal{N}et}^{(t)}\right)$: We assume that VNFs have a FIFO queue. Let μ^* denote the mean service rate and $r_m = B_m \log(1 + \delta_m)$ the wireless transmission rate, where B_m represents the wireless transmission bandwidth for the m-th user. In this respect, we further suppose that cloud processing and wireless transmission queues follow an exponential distribution with a mean $\frac{1}{\mu^*}$ and $\frac{1}{r_m}$, respectively. According to queuing theory, the mean processing delay at time step *t* is $\mathcal{L}_{proc}^{(t)} = \frac{j}{j\mu^* - \Omega}$ and transmission latency in wireless transmission queue is $\mathcal{L}_{trans}^{(t)} = \frac{1}{r_m - \lambda_m^{(t)}}$. Hence, the latency in the system is given by:

$$\mathcal{L}_{\mathcal{N}et}^{(t)} = j\mathcal{L}_d^{(t)} + \sum_{\mathrm{m}=1}^{\mathrm{M}} \left[\frac{j}{j\mu^* - \Omega} + \frac{1}{r_m - \lambda_m^{(t)}}\right],$$

where $\mathcal{L}_d^{(t)}$ denotes latency for creating, booting up and loading new VNFs and *j* denotes the total number of active VNFs to be deployed. We suppose $\mathcal{L}_{\mathcal{N}et}^{(t)} < \eta_m^{(t)}$ where $\eta_m^{(t)}$ is a predefined maximum network delay for UEs which can be viewed as quality of service (QoS) requirement.

Energy ($\mathcal{E}_{\mathcal{N}et}^{(t)}$): The energy consumption incurred by the VNF instantiation, running processors and the wireless transmission power where $\mathcal{E}_{v}^{(t)} = \psi_j$ refers to energy consumption associated with the deployment of the jth VNF instance where ψ_j is a constant value. The energy consumed by the i-th processor (in Watts) is $\mathcal{E}_{p}^{(t)} = \sigma^* P_i^3$, where σ^* is a parameter determined by the processor structure. The wireless transmission power for UE m is given by $\mathcal{E}_{w}^{(t)} = \frac{1}{\rho}\left|\left|W_m\right|\right|_2^2$, where W_m is the precoding vector from all cells to UE m, and ρ denotes the efficiency of the power amplifier at the cells. Finally, we have that the energy consumption is expressed as:

$$\mathcal{E}_{\mathcal{N}et}^{(t)} = \sum_{i=1}^{I} \sigma^* P_i^3 + \sum_{j=1}^{J} \psi_j + \sum_{m=1}^{M} \frac{1}{\rho}\left|\left|W_m\right|\right|_2^2.$$

DRL-based resource allocation

The optimal resource allocation problem is formulated as a Markov Decision Process (MDP). The aim of the CU is to improve the average DRL (Deep Reinforcement Learning) return. Towards this end, **continuous** state and action spaces are defined, as well as the reward function. The MDP for a single agent is defined by a 5-tuple (S, A, P, γ, R), consisting of a set of states S (state space), a set of actions A (action space) and P denotes the state transition probability for state s and action a. In this problem, both state space and action space are continuous and **OpenAI Gym** has been considered for the comparison of different DRL algorithms.

State space. We use Box spaces as multidimensional continuous spaces with bounds. The state at time step t consists of:

- Number of new UEs which connect to the network and request services for each slice $\left(X^{(t)}\right)$
- Computing resources allocated to each VNF $\left(C^{(t)}\right)$
- Delay status with respect to latency cost for each slice $\left(\mathcal{L}^{(t)}\right)$
- Energy status with respect to energy cost for each slice $\left(\mathcal{E}^{(t)}\right)$
- Number of users being served in each slice $\left(m^{(t)}\right)$
- Number of VNF instantiations in each slice $\left(V^{(t)}\right)$

The network state space is given by $S^{(t)} = \{X^{(t)}, C^{(t)}, \mathcal{L}^{(t)}, \mathcal{E}^{(t)}, m^{(t)}, V^{(t)}\}$.

Action space. A vertical scaling action space is considered. This vertical scaling can be classified into scale up and scale down which , respectively. The CU selects continuous value action with respect to traffic fluctuation

and learn to decide to increase/decrease computing resources allocated to each VNF. Let us denote the vertical scaling of CPU resources as $\zeta_{CPU}^{(t)}$. The change of CPU resources at time slot t is given by:

$$\zeta_{CPU}^{(t)} \in \{ z \mid z \in \mathbb{R}, -K_{Net}^{(t)} \leq z \leq K_I^{(t)} - K_{Net}^{(t)} \}$$

It is worth noting that vertical scaling is limited by the free computational resources available in the physical server hosting the virtual machine.

Reward. The aim is to minimize the total network cost to let the DRL algorithm to increase the expected return. Having this aim in mind, the return is defined as follows:

$$R^{(t)} = \frac{1}{\mathcal{N}_T^{(t)}}$$

TWIN DELAYED DDPG algorithm

The Twin Delayed DDPG (TD3) algorithm is the following:

```
Initialize actor network φ and critic networks θ1, θ2
Initialize (copy parameters) target networks φ', θ'1, θ'2
Initialize replay buffer β
Import network slicing environment ('smartech–v0')
while t < max_timesteps do
    if t < start_timesteps then
        a = env.action_space.sample()
    else
        a ⟵ π_φ(s) + ε,   ε ~ N(0, σ)
    end
    next_state, reward, done, _ = env.step(a)
    store the new transition (s_t, a_t, r_t, s_{t+1}) into β
    if t ≥ start_timesteps then
        sample batch of transitions (s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})
        ã ⟵ π_φ'(s') + ε,   ε ~ clip(N(0, σ̃), −c, c)
        Q_t = r + γ * min(Q'_{t1}, Q'_{t2})
        L = l_MSE(Q_1, Q_t) + l_MSE(Q_2, Q_t)
        θ_f ⟵ argmin_{θ_f} N^{−1} Σ(L − Q_{θ_f(s,a)})^2
        if t%policy_freq == 0 then
            ∇_φ J(φ) = N^{−1} Σ [∇_a Q_{θ_1}(s, a)|_{a=π(φ)} ∇_φ π_φ(s)]
            θ'_f ⟵ τθ_f + (1 − τ)θ'_f
            φ' ⟵ τφ + (1 − τ)φ'
        end
    end
    if done then
        obs, done = env.reset(), False
    end
    t=t+1
end
```

Herein, we omit the details of the TD3 algorithm. Interested readers are referred to [21]. TD3 is based on the actor-critic paradigm. Next, we present some remarks of the proposed technique:

1) It considers **clipped double Q-learning with pair of critic networks**. We use two DNNs as two actor networks and denote them by ϕ for the actor network and ϕ' for the actor target. Two learnings happen simultaneously, namely, Q-learning and Policy learning, and they address approximation error, reduce the bias, and find the highest Q-value.
2) **Delayed policy updates and target networks**. The main idea is to update the policy network less frequently than the value network since we need to estimate the value with lower variance.
3) **Policy smoothing and noise regularisation**. When updating the critic, a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error, increasing the variance of the target. This induced variance can be reduced through regularization to be sure for the exploration of all possible continuous parameters. We add Gaussian noise to the next action a' to prevent two large actions played and disturb the state of the environment:

$$\tilde{a} \longleftarrow \pi_{\phi}{}'(s') + \epsilon, \quad \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$$

SIMULATION RESULTS

The implementation is written in Pytorch. We measure the performance on a customized Network slicing environment, interfaced through OpenAI Gym. In this environment, the mobile network operator (MNO) collects the free and unused resources from the tenants and when slices need more resources can receive new resources. It is done either periodically to avoid over-heading or based on requests of tenants. We consider a two-tenants scenario, i.e., two slices with different QoS requirements in terms of latency and CPU constraints. For each time step, the user's packets arrive at the network and the algorithm computes the computing requirements to allocate to the relevant VNF. We compare the performance of TD3 method against a fine-tuned version of the DDPG (Deep Deterministic Policy Gradient) [21] and TD3 as well as Soft Actor-Critic (SAC) [22] to keep all algorithms consistent.

As shown in Figure 15, the learning curve of TD3 outperforms the different baselines when reaching the convergence. Although the problem formulation is general, we take the constraints into consideration as penalties to lead the agent to the good results and this is the reason of negative values in the learning curves.

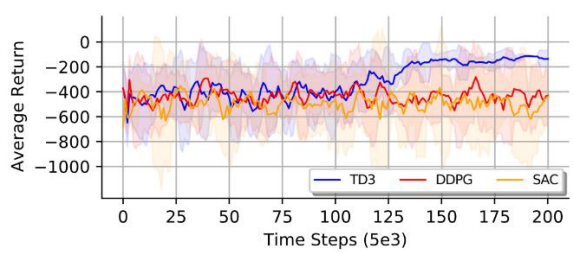
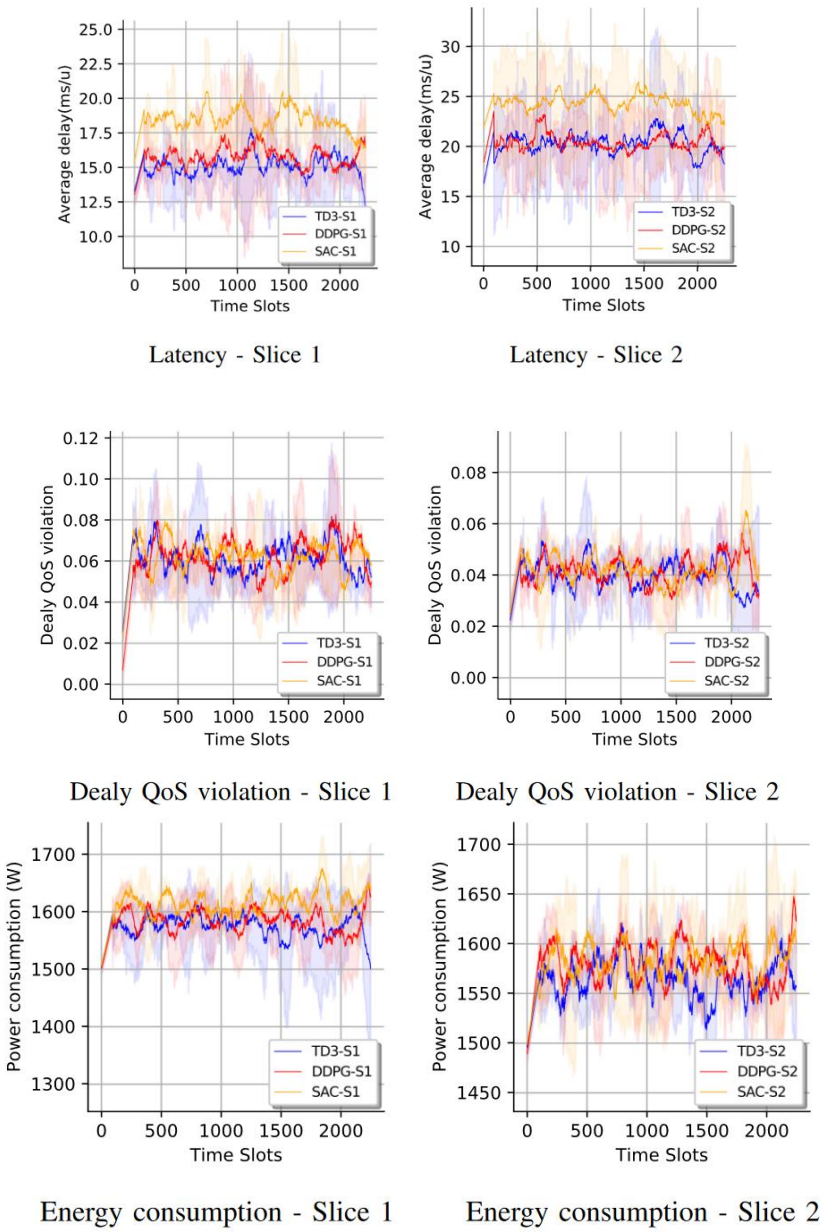


Figure 13. Learning curves of the gym NS environment for the different DRL methods.

Figure 16 presents the network performance and cost comparison between the proposed TD3 algorithm and other DRL baselines (SAC, DDPG). QoS requirement of slice 1 ($\eta_m^{(t)} = 20\,ms$), QoS requirement of slice 2 ($\eta_m^{(t)} = 40\,ms$)



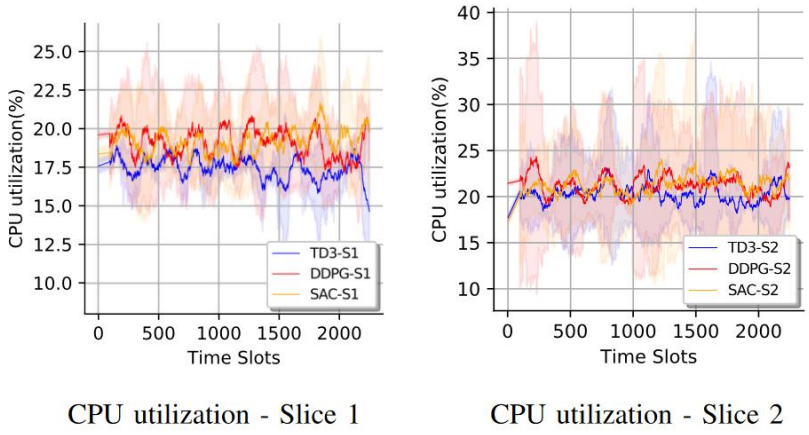


Figure 14. Network performance and costs. Comparison of DRL techniques (TD3, SAC, DDPG).

Latency: Our solution leads to less average delay per user compared to DDPG and SAC.

Delay QoS violation: The comparison between Delay QoS metric of TD3 and other DRL schemes.

Energy consumption: The performance of our scheme and other methods where the agent learns to satisfy another objective and minimize power consumption by decreasing VNFs instantiation and tuning wireless transmission power.

CPU utilization: TD3 deployment leads to more efficient usage of CPU compared to other methods.

5 Conclusions

This deliverable presents the final energy efficiency techniques that are proposed in MonB5G within the framework of Task 5.3. It starts with introducing the main energy management and orchestration architectural building blocks. Then, it presents the main energy-efficient algorithmic innovations. The first method which is shown herein is SCHE2MA. It enables a decentralized cross-domain energy efficiency. It operates with multiple agents, eliminating a centralized point of failure and is based on the auction mechanism. The performance of **SCHE2MA outperforms the two baselines, namely, classical centralized RL and static placement**.
The second solution in this deliverable is a novel statistical federate learning (StFL)-based analytic engine for slice-level KPI prediction, under strict SLA constraints. This scheme yields **more than x10 energy efficiency gain** compared to its centralized SLA-constrained deep learning counterpart while **achieving x20 lower SLA violation with respect to FedAvg**, which allows high scalability and sustainability for analysing a massive number of concurrent slices. As presented in Section 4.2.2, the communications overhead and the energy efficiency of this technique can be further improved by selecting the subset of AEs for the FL round based on their violation rate. Having this aim in mind, **a stochastic FL-based policy** is presented for scalable DE.

Finally, an advanced continuous DRL algorithm, called **twin delayed deep deterministic policy gradient (TD3)** is introduced in Section 4.3. It deals with a **multi-objective optimization problem** to make the CU learn how to re-configure the computing resources autonomously in C-RAN while minimizing latency, energy consumption, and VNF instantiation of each slice. A B5G network slicing environment is built using OpenAI Gym. The network performance and costs between TD3 is compared with other DRL benchmarks. As it is depicted, the proposed solution outperforms other DRL methods.

References

[1] GSMA, "5G-era Mobile Network Cost Evolution" [Online]. Aug. 28, 2019. Available: https://www.gsma.com/futurenetworks/wiki/5g-era-mobile-network-cost-evolution/.

[2] Analysis Mason, "The impact of 5G and next-generation networks on mobile opex spending", October 2018, Available: https://www.analysysmason.com/contentassets/dc6b0b42a98749d4a48588bc07842f34/analysys_mason_telecoms_opex_forecast_sample_may2022_rdns0.pdf.

[3] A. Laghrissi and T. Taleb, "A Survey on the Placement of Virtual Resources and Virtual Network Functions," in IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1409-1434, Second quarter 2019, doi: 10.1109/COMST.2018.2884835..

[4] R. Kołakowski, L. Tomaszewski and S. Kukliński, "Performance evaluation of the OSM orchestrator," 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2021, pp. 15-20, doi: 10.1109/NFV-SDN53031.2021.9665052..

[5] D2.4: Final release of the MonB5G zero touch slice management and orchestration architecture Available: https://monb5g.eu/deliverables/.

[6] S. Kukliński and L. Tomaszewski, "DASMO: A scalable approach to network slices management and orchestration," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1-6, doi: 10.1109/NOMS.2018.8406279.

[7] 3GPP, "Management of Network Slicing in Mobile Networks; Concepts Use Cases and Requirements," TS 28.530, v. 17.1.0, Apr. 2021.

[8] ETSI, "Zero-Touch Network and Service Management (ZSM); Closed-loop automation; Enablers," ETSI GS ZSM 009-1 V0.10.5, Jan 2021.

[9] D2.4: Final release of the MonB5G zero touch slice management and orchestration architecture Available: https://monb5g.eu/deliverables/.

[10] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590–3605, 2016.

[11] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," Computer Networks, vol. 93, pp. 492–505, 2015.

[12] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," IEEE Network, vol. 30, no. 3, pp. 81–87, 2016.

[13] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, “A deep reinforcement learning approach for vnf forwarding graph embedding,” IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1318–1331, 2019.

[14] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, “Virtual network embedding: A survey,” IEEE Communications Surveys Tutorials, vol. 15, no. 4, pp. 1888–1906, 2013..

[15] A. Cotter et al., "Two-player Games for Efficient Non-convex Constrained Optimization" [Online]. Available at: https://arxiv.org/abs/1804.06500.

[16] Y. Mao, J. Zhang, and K. B. Letaief, “Dynamic computation offloading for mobile-edge computing with energy harvesting devices", IEEE J. Sel. Areas Communications, vol. 34, no. 12, pp. 3590-3605, Dec. 2016.

[17] FS, Optical Transceivers Datasheets. [Online]. Available: https://community.fs.com/blog/understanding-the-tx-rx-optical-power-on-.

[18] H. Chergui and C. Verikoukis, “Offline SLA-Constrained Deep Learning for 5G Networks Reliable and Dynamic End-to-End Slicing," IEEE Journal on Selected Areas in Communications, vol. 38, no. 2, pp. 350-360, Feb. 2020..

[19] H.-B. McMahan et al., “Communication-Efficient Learning of Deep Networks for Decentralized Data", in 20th International Conference on Artificial Intelligence and Statistics (AISTATS' 2017).

[20] Y. Liao, "How much computing capability is enough to run a cloud radio access network?", IEEE Comm. Letters, vol. 21, no. 1, Jan. 2017.

[21] F. Rezazadeh, H. Chergui, L. Alonso, C. Verikoukis, "Continuous Multi-objective Zero-touch Netowork Slicing via Twin Delayed DDPG and OpenAI Gym", IEEE Globecom 2020.

[22] A. a. h. T. Lillicrap et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

[23] T. Haarnoja et al., “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with stochastic actor," Available: https://arxiv.org/abs/1801.01290.

[24] “ETSI ZSM 009-1, “Zero-Touch Network and Service Management (ZSM); Closed-loop Automation; Enablers”, V0.10.5 (2021-01)”.