MonB5G: AI/ML-Capable Distributed Orchestration and Management Framework for Network Slices

Sławomir Kukliński*†, Robert Kołakowski*†, Lechosław Tomaszewski*, Luis Sanabria-Russo‡, Christos Verikoukis‡, Cao-Thanh Phan§, Lanfranco Zanzi¶, Francesco Devoti¶, Adlen Ksentini‖, Christos Tselios**, George Tsolis**, Hatim Chergui‡

*Orange Polska, †Warsaw University of Technology, ‡CTTC, §B-COM, ¶NEC Laboratories Europe, ‖Eurecom, **Citrix Systems Inc.

***Abstract*—Enabled by the network slicing paradigm, a wide set of vertical services are expected to populate the future mobile ecosystem while efficiently coexisting over a shared infrastructure. However, the intrinsic diversity of vertical services, together with the heterogeneity of mobile infrastructure resources, poses severe management challenges that demand deep architectural innovations to seamlessly support enhanced orchestration mechanisms based on automation, flexibility, and programmability. In this paper, we present a novel management and orchestration platform for network slices devised by the H2020 MonB5G project. The presented concept addresses the scalability of network slicing management and orchestration by using a distributed and programmable management architecture that is AI-driven. AI-enabled management operations are adopted at different levels of the management hierarchy. The proposed architecture is a significant step towards self-managed network slices.**

***Index Terms*—5G, 6G, network slicing, AI, ML, ZSM, management, orchestration**

I. Introduction

The increasing number of deployed 5G radio access nodes, needed to satisfy ubiquitous coverage and capacity requirements, the overall 5G network complexity and growing end-user demand for reliable low-latency and high bitrate, already exacerbate the network management complexity with respect to previous mobile network generations. The human-centric management and the use of existing management solutions have become unfeasible, especially in standalone 5G architecture that uses network slicing. The network slicing paradigm is essential in 5G, as it supports concurrent provisioning of diverse vertical services over shared physical infrastructure [1]. In fact, each slice instance may be treated as an independent network, which further aggravates the management scalability and brings the complexity of management to an entirely new level. 3GPP [2] adopts the ETSI NFV MANO framework [3] for the management and orchestration of network slices. The 3GPP approach uses a centralized Operations Support System (OSS) and Business Support System (BSS) for all functional, slice and service layers, raising some questions about its scalability – currently, it has been not validated yet by large scale deployments. Moreover, the orchestration of multi-domain slices is not supported by 3GPP, and a new approach has to be developed for such cases.

This paper describes a framework for distributed orchestration and management of network slices that has been developed by the MonB5G [4] project. The framework proposes distributed implementation of management and orchestration functions to mitigate the problems mentioned above. The use of Artificial Intelligence (AI) and Machine Learning (ML) at various levels of the MonB5G management hierarchy contributes to the minimization of interactions between different entities of the framework. The embedded intelligence enables using KPIs for monitoring and intent-based reconfigurations at different management levels. In order to improve the scalability, each slice runtime management is implemented as a part of the slice and orchestrators are focused on resources. As a result, the concept is a step towards self-managed slices composed of self-managed functions, further extended by the creation of a multi-domain slice as a composition of self-managed single domain slices.

The paper is structured as follows. Section II describes the related work in the field. Section III details the MonB5G concept and describes its main architectural building blocks. Section IV discusses the implementation options of the concept. Finally, Section V concludes the paper.

II. Related work

Management and orchestration of network slices attracted tremendous efforts from both research communities and standardization bodies. Several projects funded under the EU 5G Private Public Program (5G-PPP) worked on network slices management and – more recently – on leveraging AI/ML algorithms to automate the management. 5G!Pagoda [5] was among the first EU projects to address the challenges of managing and orchestrating a high number of parallel network slices, proposing a reference architecture featuring a scalable management plane that distributes some management functions inside the network slice (namely, In-Slice Management, ISM), without application of AI/ML, however. 5G-MoNArch [6] brought three enabling innovations: *i*) cross-domain management allowing the coordination across slices and domains; *ii*) experiment-driven optimization enabling the design of highly performing algorithms; *iii*) cloud-enabled protocols increasing the flexibility of orchestration of Virtual Network Functions (VNFs). 5G-MoNArch also explored the usage of AI/ML for network slices management, adopting the ETSI ENI approach [7]. The solution, however, due to high centralization, may not scale well with the vast number of network slices. 5G-CLARITY [8] developed an AI-based network management system that provides an intent-based interface for network configuration. SELFNET [9] and its follower SLICENET [10] target self-organizing network management mechanisms jointly leveraging the NFV/SDN paradigms and AI/ML technologies. The three aforegoing projects share the same weakness, which is still high centralization of management and poor scalability when dealing with dense slice scenarios. 5G-ZORRO [11], and 5G-Ensure [12] focus on zero-touch management in a multi-stakeholder scenario, emphasizing security and trust. On the other hand, standardization groups, such as the ETSI Zero touch network & Service Management (ZSM) [13] and the Experiential Networked Intelligence (ENI) [7], have been working on using AI and ML to realize an agile, fully automated management and orchestration of network resources. ETSI ZSM has already issued a reference architecture, but it cannot be easily adapted to the management of network slices. ETSI ENI presents a centralized framework that aims to standardize different methods and policies to use AI and ML to manage networks. O-RAN specifications [14] aim to introduce AI and ML into the management of RAN resources and to improve network performances using a continuous collection of RAN-related monitoring data. In this case, the slicing support is still under discussion. Management and orchestration of network slices are also of concern of the 3GPP Release 17. Network slicing is supported at the levels of network function, slice subnet, slice instance and communication service management. The current 3GPP approach enables single domain orchestration only (single OSS/BSS and MANO orchestrator). The management architecture is complementary to the NFV MANO stack [3]. Significant effort is also put by the open-source communities, particularly ONAP [15] and OSM [16]. ONAP is one of the main solutions considered capable of addressing the rising need for automation in the management and resource orchestration processes. Still, its architecture is highly centralized and complex. OSM, since the Release FIVE, introduces the support of network slices. In the Release EIGHT, OSM moves a step forward by allowing the real-time gathering of metrics and alerts designed to manage NFV Network Service assurance in large production deployments. Moreover, both ONAP and OSM are already working on AI/ML for autonomous management.

III. MonB5G architecture description

The MonB5G architecture aims to provide a new framework for scalable and automated management and orchestration of network slices on a massive scale. It adopts the management system decomposition proposed by ITU-T [17] and uses the MAPE (Monitor-Analyze-Plan-Execute) paradigm [18] as the basis for AI-driven operations. The proposed approach has the following key features:

1) Strong separation of concerns: In MonB5G, OSS/BSS of each orchestration domain is focused on the Life Cycle Management (LCM) of slices and resource management but is agnostic to slices. Each single- or cross-domain slice can be seen as a "service" with its management platform (called embedded management or ISM [19]), which is separated from the domain(s) OSS/BSSes. ISM is a part of the slice template, and it is responsible for runtime slice management that includes fault, configuration, accounting, performance, and security management (FCAPS) of a slice. That approach provides benefits like isolation of management planes of slices (not provided by ETSI NFV MANO or 3GPP). Moreover, in the proposed approach, the deployment of a slice requires marginal modification of OSS/BSS to support its management. In the case of multi-domain slices, we have proposed to add a unique, inter-domain ISM component to the end-to-end slice template. It interacts with domain-level ISMs to achieve the end-to-end management of the slice.

2) Distribution of AI-driven management operations: The management operations are AI-driven and pursue different goals. The hierarchical embedded management concerns node, slice, orchestration domain and end-to-end slice levels. Distributed AI allows local management information processing, thus reducing the management data exchange. The application of AI also enables the use of intent-based interfaces.

3) Hierarchical end-to-end slice orchestration: One master orchestrator and multiple domain-level ones are used. It implies the use of domain-specific slice templates. The use of multiple orchestrators contributes to the orchestration scalability.

4) ISM capability of orchestration: ISM of each slice (i.e. slice OSS/BSS) may act as a service orchestrator, i.e. it may use the *Os-Ma-nfvo*-like interface [3] to request slice template modifications (the action is no longer executed by the domain-level OSS/BSS).

5) Scalable and programmable slice management: Since ISM is part of a slice and implemented as a set of VNFs, the resource scaling mechanism can contribute to its performance. Moreover, all the FCAPS functionalities (VNFs responsible for slice management) can be dynamically deployed/updated within the slice during its lifetime using the orchestration capabilities of ISM.

6) Enhanced security of slices: ISM provides isolation of the management spaces of different slices, thus enhancing slices' security. It also limits information exchange between OSS/BSS instances of each domain.

7) Support for Management as a Service (MaaS): MonB5G allows the creation of a "management slice" that can be used for runtime management of multiple slice instances of the same template. In such a case, a new business player, called Slice Management Provider, can be involved in slice management.

8) Programmable infrastructure management: The infrastructure management system can use the MonB5G mechanisms to deploy its services in a similar way in which slices are deployed. The framework is aware of the energy cost of infrastructure resources and may act for its optimization.

The mentioned features are inline with the key ETSI ZSM requirements. The proposed approach uses AI mechanisms at different levels of the management hierarchy. At each level, there exists a Monitoring System (MS), one or more Analytic Engines (AEs) and Decision Engines (DEs). MS interacts with entities and processes their monitoring information (aggregates, filters, etc.); AEs analyse the data coming from MS to find anomalies; finally, DE implements an algorithm that proposes reconfiguration. It is assumed that AEs and DEs are AI-driven.

The MAPE/AI-based FCAPS management is implemented hierarchically, featuring different control loops of different scopes, goals, and timescales, at the following levels:

- **Central OSS/BSS level:** MAPE functions are responsible for end-to-end slice management and orchestration in multi-domain environment – global actions, e.g. cross-slice and cross-domain optimizations.
- **Orchestration Domain level:** MAPE functions are responsible for domain-level (e.g. cloud infrastructure, RAN, etc.) FCAPS and domain resources management (slice admission control, allocation of resources to slices).
- **Slice level:** MAPE functions are responsible for each slice management (FCAPS) and are embedded in slice template (i.e. ISM) – it also concerns multi-domain slices.
- **Node (VNF/PNF/CNF) level:** MAPE functions are responsible for node-level management (can be seen as modified, intelligent Element Manager, cf. MANO [3]). They can be used for node plug-and-play functionality. Their local loops contribute the most to the reduction of the monitoring exchange.

The hierarchy of MAPE introduces fast local control loops and increasingly slower ones for wider-scope administrative domains (e.g. slice-level, tenant-level, etc.). Such time scale separation and hierarchization contribute to the overall stability of the system. All the mentioned MAPE-based subsystems cooperate to achieve an overall goal.

All subsystems use the mentioned MS/AE/DE triplets for specific, control loop-based optimization, in most cases driven by AI algorithms. Intent-based interfaces are proposed for inter-subsystem exchange. The intent-based interfaces exchange high-level information, typically translated into multiple, low-level operations by an AI-driven engine. Due to the use of such interfaces, the information exchange between the management system components is minimized. The system stability is monitored, and some decisions to improve it are taken, if necessary.

The MonB5G architecture is composed of static and dynamic components (cf. the forthcoming sections), which provide support for operations related to fault management (self-healing), self-configuration, performance optimization (including energy-saving) and security of slices. In the presented concept, the term “slice” is used not only as referring to “network slices” *per se*, but also to any other set of interconnected virtual functions serving a specific goal, e.g. providing a management functionality using MaaS.

A. Static components of the architecture

The business actors and static components of the architecture are presented in Fig. 1 where slices are omitted for clarity. The components and their main functionalities are the following:

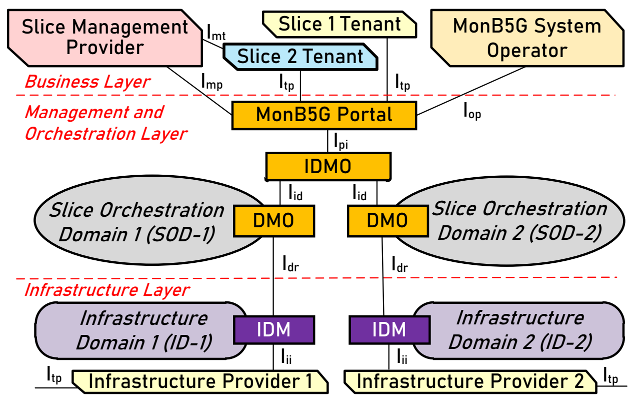
Slice Management Provider
I_{mt}
Slice 1 Tenant
Slice 2 Tenant
MonB5G System Operator
Business Layer
I_{mp}
I_{tp}
I_{tp}
Management and Orchestration Layer
MonB5G Portal
I_{op}
I_{pi}
IDMO
I_{id}
I_{id}
Slice Orchestration Domain 1 (SOD-1)
DMO
DMO
Slice Orchestration Domain 2 (SOD-2)
I_{dr}
I_{dr}
Infrastructure Layer
Infrastructure Domain 1 (ID-1)
IDM
IDM
Infrastructure Domain 2 (ID-2)
I_{ii}
I_{ii}
I_{tp}
Infrastructure Provider 1
Infrastructure Provider 2
I_{tp}

Fig. 1. Static components of and business actors of MonB5G architecture (none slice deployed)

1) MonB5G Portal: The portal is used by Slice Tenants, Slice Management Providers and Infrastructure Providers to request operations related to slice LCM, i.e. slice deployment, slice modification and slice termination. Slice Management Providers may use MaaS to manage multiple instances of slices based on the same template (as the slice runtime management is slice-specific). The Infrastructure Providers are allowed to ask for orchestration of infrastructure-oriented management functions similarly as Slice Tenants request slices. For both cases, the I_{tp} interface is used (typically a web-based interface). The MonB5G operator also interacts with the system via the MonB5G portal using I_{op} management interface. The MonB5G Portal exposes the MonB5G framework capabilities (available slice templates, etc.) and partakes in negotiations related to the business dimension of the contract. In order to make the negotiations, it interacts with IDMO and this interaction, via the I_{pi} interface, is afterwards used for LCM of negotiated slices.

2) Inter-Domain Manager and Orchestrator (IDMO): This entity plays a crucial role in slice preparation and deployment phases by negotiating deployment policy with slice requester (Slice Tenants, Slice Management Providers or Infrastructure Providers). IDMO interacts with DMOs (see further) via I_{id} to deploy the end-to-end-slices, based on the information obtained from DMOs. It is responsible for the modification of the end-to-end slice template before its deployment according to the negotiated contract. The modified slice template includes slice stitching mechanisms for obtaining the end-to-end slice and its proper modification management plane (correlation of events and KPIs from different domains that are used for slice deployments). It can be seen as an end-to-end orchestrator (umbrella orchestrator [20]). If there are multiple infrastructure owners, IDMO may decide how to split the end-to-end slice template into single-domain templates before end-to-end slice deployment. The split may be shaped by various factors, e.g. price, performance or energy efficiency. For that purpose, IDMO is aware of all infrastructure domains involved in the system and the status of their resources. IDMO also keeps a permanent Accounting Database, as ISM is not a permanent one. IDMO interacts with the Infrastructure Provider via

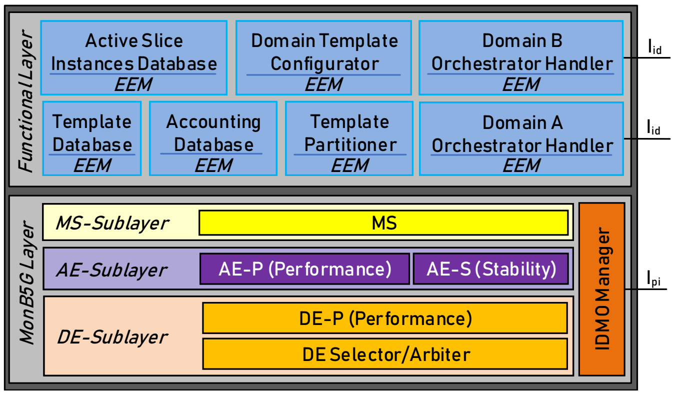

Fig. 2. Main blocks of IDMO (an example)

DMO. The IDMO structure is decomposed into Functional and MonB5G Layers (cf. Fig. 2 for IDMO internal structure details). The MonB5G Layer (management of IDMO) is AI-driven, uses MS/AE/DE triplets and other components of the management architecture. The approach is described in details later.

3) Domain Manager and Orchestrator (DMO): DMO of a Slice Orchestration Domain (SOD) is responsible for the orchestration of slices in its domain and the management of resources of this domain. In the NFV MANO-orchestrated domain, DMO can be seen as a combination of resource-oriented OSS/BSS and MANO orchestrator. In other technological domains, other orchestrators can be used. It has to be noted that IDMO does not interact directly with the orchestrator but with the OSS/BSS of each SOD. Therefore, the IDM-IDMO interface can be defined in a similar way for different orchestration technologies. DMO is focused on SOD operations concerning resources (resource allocation to slices, slice LCM, resources FCAPS) and is agnostic to slices, i.e. it is not involved in slice runtime management, including initial slice configuration. Therefore, DMO deals only with the software dimension of slices (LCM, resource scaling), whereas the runtime management is handled by the management components embedded in slices. Similarly to IDMO, all DMO operations are AI-driven. Therefore, the internal structure of DMO is also composed of Functional and MonB5G layers. Operations are mostly related to slice admission, LCM and resource sharing. The operations, as well as the exchange of infrastructure-related data (e.g. about energy consumption), are performed via I_{dr} interface.

4) Infrastructure Domain Manager (IDM): The framework enables programmable infrastructure management for which a separate entity – IDM – is responsible. It has an interface to the Infrastructure Provider, who can use the MonB5G portal, asking for the deployment of additional infrastructure management functions, called IOMFs (see further). The functions are orchestrated similarly to slices, and LCM requests are sent by the Infrastructure Provider via the MonB5G Portal.

B. Dynamic components of the architecture

The dynamic components of the architecture are slices that are also involved in the overall management of the MonB5G system. For that purpose, the slice runtime management, in most cases, is a part of the slice (i.e. included in a slice template). Such an approach leads to self-managed slices and reduces information exchange between slices and external management components of the architecture. However, for backward compatibility and enabling MaaS, it can deploy the management part of the slice independently. Moreover, MaaS can handle multiple slice instances based on the same template. The different options of deployed slices are presented in Fig. 3. Option A concerns the deployment of the self-managed multi-domain slice, Option B shows the deployment of slices that use shared functions and MaaS, and Option C shows the infrastructure management-oriented slice deployment.

In all the mentioned cases, the MonB5G slice templates have been decomposed into Slice MonB5G Layer (SML) and Slice Functional Layer (SFL). SFL, composed of virtual functions, provides slice "core" functionality. SML provides its management as an implementation of the ISM concept. A generic structure of a self-managed, single domain slice (SML/SFL) is shown in Fig. 4, which shows more SML details.

The SFL part may use shared functions, called Domain Shared Functions (DSFs), which are available in SOD (so-called shared VNFs). These functions can be implemented as PNF/VNF or CNF and may be used by all or some slices. The use of DSFs reduces the deployed slices' footprint, improving that way also slices deployment time. DSFs are grouped (i.e. form a slice) for their easy management by DMO and can be seen as an implementation of PaaS. SFL part of slice template consists of AI-driven Element Managers, called Embedded Element Managers (EEMs, composed of parts for monitoring (MS-F), anomaly detection (AE-F), decision making (DE-F) and actuation (Actuator Function, ACT-F). Legacy Element Managers (EMs) can also be used [3]. Moreover, both EM and EEM contain Management Function (MAN-F) for the external management of the component. Each EEM is involved in some local closed-loop management decisions, reducing the outside exchange of monitoring data. The ACT component is responsible for converting high-level DE output (e.g. the action space of a deep reinforcement learning algorithm) into a set of low-level primitives stemming from API. EEMs and EMs are the links between the SFL and SML parts of a slice.

The SML part is internally split into MS, AE and DE sublayers. It is assumed that MS provides generic, reusable monitoring with certain granularities, whereas AEs and DEs cooperate to achieve a specified data-driven proactive decision. Specifically, AE analyses MS data received for a specific purpose (e.g. fault detection, performance prediction or security attack identification). The analysed data are then fed to the related DEs (as a state-space) to take appropriate decisions that are afterwards converted by ACT to elementary actions. As in the management systems, there are multiple goals to be optimized; therefore, multiple AEs and DEs can be a part of SML. To resolve unavoidable conflicts between DEs, the DE Selector/Arbiter component is a part of the DE sublayer. Moreover, stability is observed to avoid a chaotic behaviour

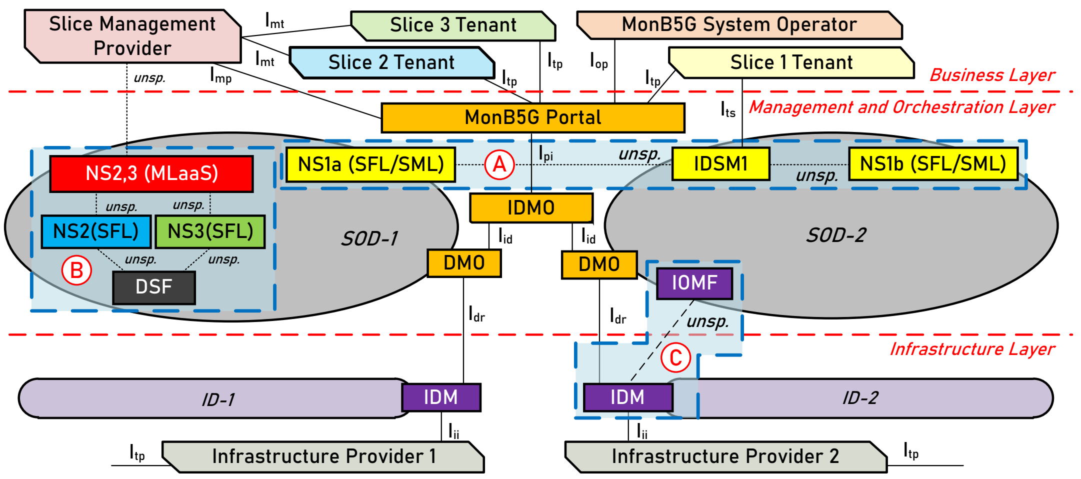

Fig. 3. MonB5G system with different options of slice deployment: A – multi-domain slice; B – slices that use MaaS and shared functions, C – orchestrated management functions of the Infrastructure Provider

of the system or the ping-pong effect. SML DEs may decide about SML and SFL functionality updates. They may change SFL configuration parameters or send new function (VNF) orchestration requests (Network Service Update according to ETSI NFV framework) using a direct connection between each SML and DMO. SML may also ask for a resource allocation update. Such resource scaling can be proactive, based on monitored slice users' QoE. This is in contrast to the NFV MANO orchestrator's reactive resource scaling only. SML provides direct, intent-based management to the Slice Tenant that provides a perfect way of slices management isolation. The MonB5G Slice Manager component of SML is responsible for the interaction with DMO. It allows for SML configuration that enables completely manual management of a slice if necessary (overriding AI-based management).

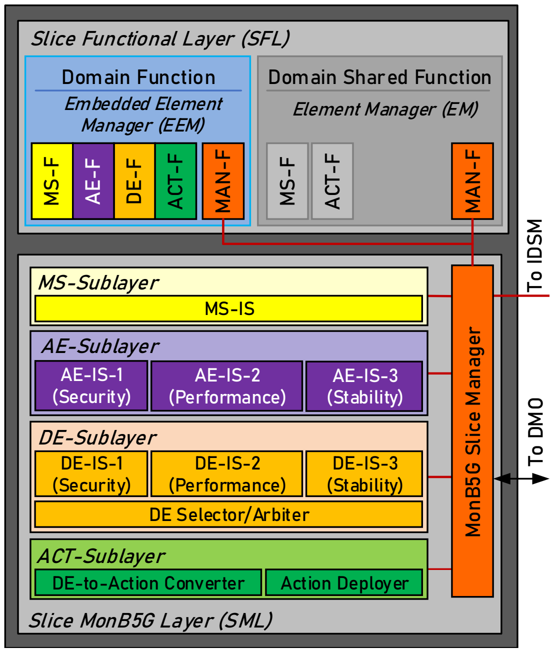

Fig. 4. Generic structure of the ML/FL slice

When a slice spans multiple SODs, the Inter-Domain Slice Manager (IDSM) entity is responsible for the end-to-end slice management. It interacts with SMLs of all domain slices composing the end-to-end slice. IDSM is a part of the end-to-end slice template. In some cases, it can be generated automatically by IDMO (if IDMO is responsible for slice template split between multiple SODs). When IDSM is in use, it provides the management interface to the Slice Tenant and is responsible for calculating slice KPIs. Fig. 3 option A shows a multi-domain slice in which IDSM is deployed in one of SODs.

The addition of SML to SFL undoubtedly increases slice footprint, also implying longer deployment time. To address this issue, MonB5G proposes using MaaS/PaaS paradigm already defined by ETSI [21]. In this case, SML is an independent slice capable of managing multiple SFL instances of the same template (since SML is slice-specific, it cannot be used for generic slice management). Such a split requires implementing additional functions in SML related to the creation of secure partitions (SMLs) for the managed SFLs and dynamic adaptation in case of deployment of a new SFL or termination of the existing one. The MaaS platform (called MonB5G Layer as a Service, MLaaS) can be operated by a business entity named Slice Management Provider. LCM of MLaaS is done via I_{mp} interface. This case for a single SOD is marked as Option B in Fig. 3 where SFLs of the MLaaS-managed slices also use DSF services for the reduction of the SFL footprint.

IDM functionalities supporting the management of infrastructure can be dynamically enhanced by the use of additional management services, which are orchestrated by the framework. Such additional management functionalities, called Infrastructure Orchestrated Management Functions (IOMFs), can be orchestrated by DMO on the request of the Infrastructure Provider (via the MonB5G Portal). The functions cooperate with IDM to achieve its goal (predictions of resource consumption, etc.). This option is marked in Fig. 3 as C.

IV. Implementation Remarks

The vision of the MonB5G framework is a high-level one, thus enabling different implementation approaches. The implementation should reuse existing, standardized and open source components and interfaces. The implementation remarks are following:

1) MonB5G Portal: It exposes MonB5G management services to tenants, providers, and operators via Web endpoints, translates requests to orchestration procedures (of e.g. code, resources, etc.) and uses standardized interfaces and open-source tools, such as Ansible [22].

2) IDMO: Its operations leverage a distributed messaging platform, e.g. Apache Kafka [23], to enable interactions among entities of multiple domains. It may also use Kubernetes federation [24] for provisioning or updating SMLs configuration across domains.

3) DMO: It can incorporate existing interfaces and tools for resource orchestration in its corresponding SDO (e.g. ETSI OSM [16]) and extend them, e.g. to enable energy efficiency, PaaS provisioning, etc. The OSS/BSS part of DMO is an AI-driven management platform focused on resources. It interacts with IDMO, slices and infrastructure via a distributed messaging platform.

4) IDM: It is the OSS/BSS (preferably AI-driven) of the infrastructure. It supports the orchestration of IDM management functions (i.e. IOMFs) at the request of Infrastructure Providers.

5) SML and SFL: SML's MS/AE/DE triplets can be reused for different slice types. SML exposes an intent-based interface for KPIs enforcement and visualization (Grafana, Prometheus, etc.). SFL is composed of the tenant's VNF and corresponding EEM. SML VNFs support PaaS itself (e.g. Kubernetes cluster), which effectively acts as a runtime environment for MonB5G management components.

6) MLaaS and DSF: Both are shared among groups of SFLs. MLaaS relies on PaaS [21] services (i.e. Kubernetes) to ensure secure isolation of management functions to each SFLs as well as LCM and operation (e.g. scaling, updates). MLaaS and DSF resources are orchestrated alongside with SFLs.

V. Summary and Conclusions

In this paper, we have presented the MonB5G framework for distributed orchestration and management of network slices. To improve network slices' management and orchestration scalability, the proposed architecture allows for local processing of management data at different hierarchical levels, providing a flexible decision-making framework to deal with runtime and life cycle management of a massive number of slices. The distributed approach enables fast reaction to events, minimizing the management information exchange between components in comparison to centralized approaches, consequently increasing the overall network scalability. The paper provides only a high-level, introductory description of the concepts for which the implementation effort is following the ongoing remarks of Section IV. It has to be noted that the architecture is agnostic to AI algorithms and management procedures. Therefore, their definition is out of the scope of this paper.

Acknowledgment

This work has been supported by the EU H2020 project MonB5G (Grant Agreement No. 871780). The views and opinions are those of the authors and do not reflect the official position of their companies.

References

[1] NGMN Alliance, "NGMN 5G White Paper", Jan. 2015.

[2] 3GPP, "Management of network slicing in mobile networks; Concepts, use cases and requirements", 3GPP TS 28.530, ver. 17.1.0, Apr. 2021.

[3] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI GS NFV-MAN 001 V1.1.1, Dec. 2014.

[4] MonB5G project, "Distributed management of Network Slices in beyond 5G". [Online]. Available: https://www.monb5g.eu [Accessed: Jul. 22, 2021].

[5] H2020 EU/JP, "5G!Pagoda: Federating Japanese and European 5G Testbeds to Explore Relevant Standards and Align Views on 5G Mobile Network Infrastructure Supporting Dynamic Creation and Management of Network Slices for Different Mobile Services". [Online]. Available: https://5g-pagoda.aalto.fi/ [Accessed: Jul. 22, 2021].

[6] 5G-PPP, "5G-MoNArch: 5G Mobile Network Architecture for diverse services, use cases, and applications in 5G and beyond". [Online]. Available: https://5g-monarch.eu/ [Accessed: Jul. 22, 2021].

[7] ETSI, "Experiential Networked Intelligence (ENI)". [Online]. Available: https://www.etsi.org/committee-activity/eni [Accessed: Jul. 22, 2021].

[8] 5G-PPP, "Beyond 5G Multi-Tenant Private Networks Integrating Cellular, Wi-Fi, and LiFi, Powered by Artificial Intelligence and Intent Based Policy". [Online]. Available: https://www.5gclarity.com [Accessed: Jul. 22, 2021].

[9] 5G-PPP, "SELFNET: A framework for self-organized network management in virtualized and software defined networks". [Online]. Available: https://selfnet-5g.eu/ [Accessed: Jul. 22, 2021].

[10] 5G-PPP, "SLICENET: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks". [Online]. Available: https://slicenet.eu/ [Accessed: Jul. 22, 2021].

[11] 5G-PPP, "Zero-touch service, network and security management in multi-stakeholder environments". [Online]. Available: https://www.5gzorro.eu [Accessed: Jul. 22, 2021].

[12] 5G-PPP, "5G Enablers for network and System security and resilience". [Online]. Available: https://www.5gensure.eu/ [Accessed: Jul. 22, 2021].

[13] ETSI Zero touch network & Service Management (ZSM). [Online]. Available: https://www.etsi.org/committee/zsm [Accessed: Jul. 22, 2021].

[14] O-RAN Working Group 2: AI/ML workflow description and requirements, Tech. Rep. O-RAN.WG2.AIML-v01.01.

[15] The Linux Foundation, "Open Network Automation Platform". [Online]. Available: https://www.onap.org/ [Accessed: Jul. 22, 2021].

[16] ETSI, "Open Source MANO". [Online]. Available: https://osm.etsi.org/ [Accessed: Jul. 22, 2021].

[17] ITU-T, "Overview of TMN Recommendations", ITU-T M.3000 (02/00), Feb. 2000.

[18] IBM, "Autonomic Computing White Paper: An architectural blueprint for autonomic computing", 3rd edition, 2006.

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

[20] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options", ETSI GS NFV-IFA 009, V1.1.1, Jul. 2016.

[21] ETSI, "Report on the Enhancements of the NFV architecture towards Cloud-native and PaaS", ETSI GR NFV-IFA 029, V3.3.1, Nov. 2019.

[22] Red Hat Ansible. [Online]. Available: https://www.ansible.com [Accessed: Jul. 22, 2021].

[23] Apache Kafka. [Online]. Available: https://kafka.apache.org [Accessed: Jul. 22, 2021].

[24] Kubefed. [Online]. Available: https://github.com/kubernetes-sigs/kubefed [Accessed: Jul. 22, 2021].