Miquel Puig Mena, Apostolos Papageorgiou, Leonardo Ochoa-Aday, Shuaib Siddiqui, Gabriele Baldoni. *Enhancing the Performance of 5G Slicing Operations via Multi-Tier Orchestration*. 23rd Conference on Innovations in Clouds, Internet and Networks (ICIN 2020).

Enhancing the performance of 5G slicing operations via multi-tier orchestration

Miquel Puig Mena, Apostolos Papageorgiou,
Leonardo Ochoa-Aday, Shuaib Siddiqui
Software Networks group
i2CAT Foundation
Barcelona, Spain
apostolos.papageorgiou@i2cat.net

Gabriele Baldoni
Advanced Technology Office
ADLINK Technology Inc.
Paris, France
gabriele.baldoni@adlinktech.com

***Abstract*—End-to-end 5G network slicing that spans across diverse access and core networks, as well as cloud and edge infrastructure, promises to satisfy on-demand the requirements of different vertical applications in a fast and cost-efficient manner. However, in order to achieve this, slice management systems need to combine different types of orchestrators. The combined usage of such orchestrators can be performed in many ways and related studies of standardization bodies have led to various open issues. This paper presents a solution based on a multi-tier orchestrator which glues NFV, MEC, and Cloud-native orchestrators using API abstraction layers and inter-orchestrator coordination workflows, while revisiting some standardized directives for NFV-MEC integration. The multi-tier orchestrator is evaluated against standard approaches that are based on the aforementioned directives, showing performance enhancements of up to 13x in terms of CPU load of certain orchestrator hosts in scenarios where up to 15 services are instantiated concurrently in an integrated NFV-MEC environment.**

***Index Terms*—NFV, MEC, slicing, 5G, orchestration**

I. Introduction

The term *network slicing* was introduced around a decade ago, but the concrete specifications of its diverse ingredients are being still strongly shaped within the context of the evolving 5G technologies and standards. The common denominator of related 3GPP [1], NGMN [2], and other related standards, as well as prominent research studies [3], is that slices are isolated and individually manageable and configurable sets of network infrastructure resources together with the (physical or virtual) network functions of standard network architectures that are deployed on them. The usage of diversified slices shall help to satisfy requirements of different tenants or vertical applications with reduced costs [4].

In order to make sure that network slices can be created quickly and on-demand to serve these diverse requirements, the aforementioned slicing standards and solutions are very tightly related to network softwarization, especially NFV (Network Function Virtualization), because it enables a faster and more dynamic setup and configuration of the network services involved in the slices. Therefore, NFV orchestrators (NFVO), comprising the central intelligence of NFV solutions, will certainly play an important role in a slice management system. However, end-to-end slices span across various tiers, e.g., access networks of different technologies, core networks, Data Centers (DC) in the Cloud and at the edge, intra-DC networking elements, and more. Moreover, different groups of resources on any of those tiers might be virtualized and managed based on different technologies (e.g., containers vs VMs). This will mean that end-to-end slice management systems will not only have to use different orchestrators, but also different *types* of orchestrators.

For example, an NFVO (as per the ETSI NFV specs) and a MEC (Muli-access Edge Computing) Application Orchestrator (MEAO) are likely to be required in many end-to-end slice management systems. Further, the trend in favour of Cloud-native systems [5] will mean that Cloud-native orchestrators (e.g., orchestrators based on the *kubernetes* platform that are relevant for the slicing/NFV lifecycle but use models and workflows that are not strictly compatible with ETSI NFV or ETSI MEC) will also need to be integrated. Although many attempts for specifying the combined and coordinated usage of NFVO and MEAO have been performed during the last years (cf. [6]), they were not performed in the scope of end-to-end slicing, while the potential co-existence of Cloud-native orchestrators was also out of scope of such studies.

This paper suggests a solution in which a multi-tier orchestration layer with an inter-orchestrator coordination intelligence mediates the triggering of NFV, MEC, and Cloud-native orchestrators within the slice management lifecycle. This involves harmonization of the descriptors and models used by the different orchestrator types, as well as consideration of slicing-related workflows which can be resource-consuming if applied at scale. In this regard, section II explores the background and related work in terms of NFV-MEC integration and other multi-orchestrator solutions, section III presents our solution along with a revision of related open issues from ETSI specifications, section IV evaluates the multi-tier-orchestration approach in terms of CPU load of the involved orchestrator hosts, while section V concludes and discusses future directions. The results show that our approach can achieve up to 13 times lower CPU load of the MEAO host compared to cases in which the MEAO needs to act as a master in the integrated NFV-MEC environement.

978-1-7281-5127-4/20/$31.00 ©2020 IEEE

II. Background and Related Work

Works related to the coordination of NFV and MEC orchestrators are the most relevant and important in our context, because they look into coordinating different *types* of orchestrators, and thus two standardized orchestrators which are also used in our solution. However, other solutions that present layered orchestrators are briefly explored as well.

A. NFV-MEC integration

The flagship publication on the topic of NFV-MEC integration is ETSI's study on the *Deployment of Mobile Edge Computing in an NFV environment* [6]. This work describes a reference architecture for achieving such a deployment, including and inter-relating all the involved NFV interfaces, all the involved MEC interfaces, as well as three additional interfaces that are meant to facilitate this integration. One of these additional interfaces (called Mv1) is interconnecting the NFVO with the MEAO and it is the only interface that is used to enable a coordinated usage of the two orchestrators. In its details, the study mainly describes the various implications that the MEC deployment in an NFV environment has for the descriptors and the lifecycle management of VNFs (Virtual Network Functions) and MEC applications. More details will be understood when comparing our approach to the solutions proposed in [6] in Section III.

In the work of [7], we have revisited the 14 open issues identified by [6] and suggested how they were handled in a platform that is focused on enhancing the edge capabilities of an NFV-based platform for 5G neutral hosts. That work already introduced the concept of a multi-layer orchestrator on top of the NFVO and MEAO. However, the current paper extends that work beyond high-level architectural enhancements, providing a detailed description, analysis, and evaluation related mainly to four of these fourteen issues (cf. III-B).

Various other works consider NFV-MEC integration from different perspectives. For example, [8] describes how to combine MEC and NFV functional blocks in order to deploy CPU-intensive services in a way that enhances application-specific metrics such as transcoding efficiency. Further, [9] suggests to define proximity zones for the MEC servers within an NFV environment in order to reduce latency. Finally, [10] and [11] investigate in the context of merged MEC-NFV environments the topics of 5G core components management and VNF placement, respectively. However, none of these works revisit any aspect related to the NFVO-MEAO coordination or the Mv1 interface, and thus they all implicitly or explicitly endorse the solution of [6] for that part.

B. End-to-end orchestration

In contexts other than NFV-MEC integration, layered network service orchestrators usually appear with name "End-to-end orchestrator". For example, the Cloud CO (Central Office) reference architecture of the BroadbandForum [12] includes an End-to-end Service Orchestrator (E2E SO), which interconnects Cloud CO domain orchestrators, which are practically NFVOs that are extended or customized in a way that fits the Cloud CO case. Therefore, this case does not present any issues for harmonizing diverse orchestrator workflows and concepts, since the coordinated orchestrators are of the same type and technology. Similarly, the end-to-end network service orchestration mechanisms described in [13] is applied on top of per-domain NFVOs, not addressing any challenges that would arise by orchestrator heterogeneity due to the involvement of edge or Cloud-native orchestrators.

III. Multi-tier Orchestration

This section presents the architecture, the most important workflows, algorithmic aspects, and some implementation notes about our solution for performing slicing based on a multi-tier orchestration layer. Further, it revisits some of the ETSI-identified open issues for NFV-MEC integration, for which our solution follows a different approach or provides new insights.

A. Architecture and Way of Operation

As shown in Fig. 1, the Multi-tier Orchestrator (MTO) lies between an end-to-end Slice Manager and the NFV/SDN and Cloud-edge orchestration domains, and it mediates the triggering of high-level actions such as onboarding, instantiation, and monitoring of (network) services and platform rules and configurations.

The MTO includes an *Abstraction API*, which is used to trigger the required API invocation chains on the different orchestrators when a high-level action is performed. However, it is not a typical, plain abstraction layer that simply maps and translates high-level API calls to low level invocations. It contains a layer of intelligence (*Forwarding and coordination*

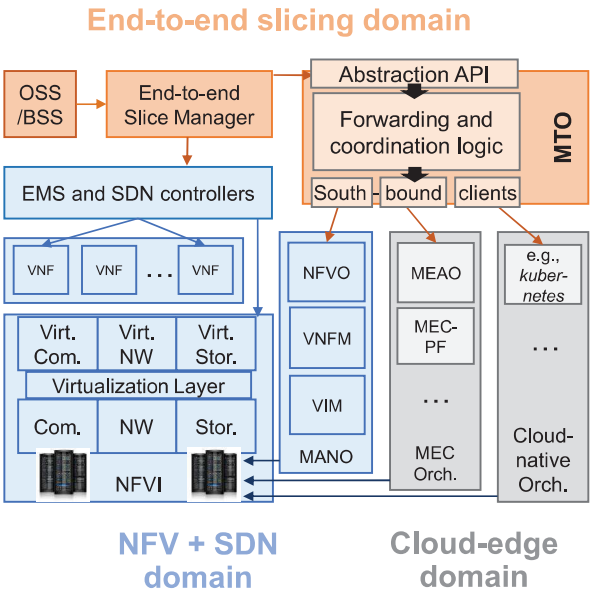


Fig. 1. High-level architecture of multi-tier orchestration solution

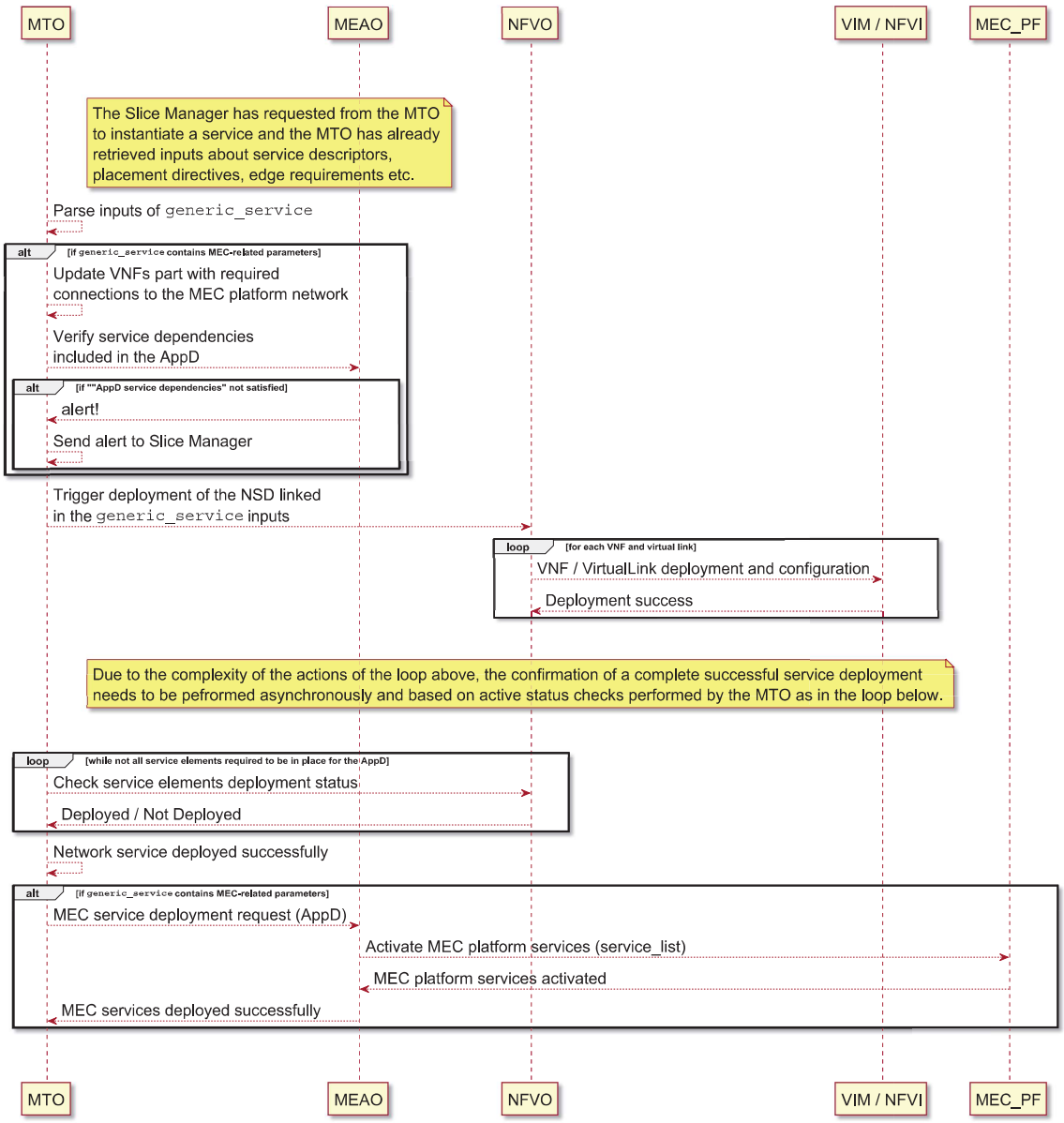


Fig. 2. MTO workflow for NFV and MEC orchestrator coordination

logic), which implements new inter-orchestrator workflows based on the high-level inputs. Finally, a series of *Southbound clients* is used in order to connect to NFV, MEC, and Cloud-native orchestrators. The NFV MANO (Management and Orchestration) consists of the NFVO, the VNFM (Virtual Network Function Manager), and the VIM (Virtual Infrastructure Manager) and deploys VNFs (Virtual Network Functions) on the virtualized compute, network (NW), and storage resources of the NFVI (Network Function Virtualization Infrastructure), as specified by the ETSI NFV architecture [14]. Similarly, the MEC Orchestration involves the MEC Application Orchestrator (MEAO) and the MEC platform (MEC-PF) (see also [6]), while both the MEC Orchestrator and the Cloud-native Orchestrators can also act upon the NFVI resources.

An important workflow that is implemented by the Forwarding and coordination logic is related to the coordinated triggering of NFV and MEC orchestrators and is shown in Fig. 2. For instantiating a cross-domain service for a certain

slice, the */generic_service* endpoint of the Abstraction API of the MTO is triggered with inputs that are related to different domains, including links to standardized descriptors that are used by the different orchestrators, e.g., NFV Network Service Descriptors (NSD) and MEC Application Descriptors (AppD). If MEC-related inputs are present, then the MTO adds to the VNFs of the NSD all the required connections towards the MEC platform and checks with the MEAO for further dependencies of the VNFs on services of the MEC platform. In the next steps, it first deploys the VNFs and then sets the MEC platform services and rules, by triggering the respective orchestrators as shown in the diagram, reporting any incompatibilities back to the Slice Manager. Similar workflows apply to cases that involve other orchestrator types or other kinds of actions (e.g., service status monitoring).

B. Comparison with ETSI NFV-MEC integration standards

With regard to the NFVO-MEAO interaction and coordination, the ETSI study about the deployment of MEC in an NFV environment [6] suggests for the case of service and application onboarding that either of the two orchestrators should act as a *master*, thus mediating all calls to the other orchestrator. For the rest of the service and application lifecycle operations, it suggests that the MEAO should use the NFVO-MEAO interface (Mv1) to trigger the subset of the NFVO functionalities that it requires in order to accompany the deployment of its (mobile edge) applications, which is practically also a kind of *master-based* approach. Of the 14 open issues identified in the study, various are directly or indirectly dependent on this approach. In the following we revisit the ETSI-identified open issues that are tightly related to the NFVO-MEAO coordination method and we discuss what our MTO-based solution implies compared to the possible master-based solutions:

- *Communication between MEAO and NFVO via Mv1* (ETSI-identified issue #3): In the master-based solution, the master (i.e., normally the MEAO) not only needs to integrate the logic for becoming a client of the other orchestrator, but in many cases it also needs to maintain state across calls and to keep being synchronized about the execution steps of diverse actions. For service instantiation, for example, this could imply that the master keeps monitoring the status of the deployment of the VNFs and synchronizes its own orchestration tasks (e.g., edge service activation, traffic redirection) accordingly. In the MTO-based solution, the hideous task of synchronizing such actions is taken outside both orchestrators in order to avoid interfering with the actual tasks of the orchestrator. The sequence of Fig. 2 is actually a high-level example of such a case.
- *VNF Package vs. MEC application package* (ETSI-identified issue #7): In the master-based solution, it is suggested that the VNF package (i.e., the VNF descriptor and potentially the VNF image) are extended in order to include MEC-specific files. It is noted that care is required in order to prevent that these extensions interfere with the NFV lifecycle. In the MTO-based solution, VNF and ME application packages are left intact, requiring an external catalogue that inter-relates NS/VNF descriptors with AppD's. This implies a tradeoff of harder synchronization of the used repositories for a cleaner separation of concerns and lower complexity.
- *VNF package onboarding* (ETSI-identified issue #8): In the master-based solution, both orchestrators are involved for most cases of VNF package onboarding. In the MTO-based solution, only the external catalogue and the responsible orchestrator are triggered.
- *NFV construct that corresponds to Mobile Edge Host* (ETSI-identified issue #11): In the master-based solution, it is suggested that NFVI constructs such as *NFVI-PoP* (Point of Presence) and *Zone* be used in order to represent mobile edge hosts, based on definitions, assumptions, and mappings that are yet to be defined. In the MTO-based solution, higher-level elements such as the Slice Manager and the MTO (cf. Fig. 1) are using abstractions of the VIM in order to group and model compute resources, with the mobile edge hosts being a type of them. For example, our implementation uses the OpenStack availability zone concept for this purpose, In terms of NFVO-MEAO coordination, this simply avoid that any aspect of this mapping needs to be known and handled between the two orchestrators.

C. Testbed Implementation

The Slice Manager and the MTO have been implemented from scratch, while the rest of the core components of the architecture presented in Fig. 1 have been implemented using open-source technologies. Technical information for the latter is provided below:

- *NFVO*: Open Source MANO (OSM release 6.0.1) has been deployed as the orchestration component in our experimental prototype. This platform includes also the VNFM and provides end-to-end network service orchestration by following the ETSI MANO standards. Additionally, the used release has been enhanced with the connectors that are required to seamlessly deploy services on an edge infrastructure managed by a platform such as fog05 [15] (more information follows).
- *VIM*: For the VIM, we selected the open-source Eclipse project fog05. This can act as an edge VIM, which is not only supported by the OSM orchestrator, but also provides provisioning and management of compute, storage and networking resources in decentralized infrastructures. Moreover, fog05 is a lightweight platform that enabled us to instantiate NSDs based on *lxc* containers, which minimizes the footprint in our datacenters.
- *MEAO and MEC_PF*: These two elements have been implemented on the same server instance using plug-ins on top of the fog05 software, as described in [7]. The MEAO orchestrator receives the service dependencies included in the AppD from the MTO and it subsequently activates the corresponding services on the MEC_PF following the

TABLE I
RESOURCES ASSIGNED TO DEPLOYED ORCHESTRATORS.

Component	CPU (cores)	RAM (GB)	Disk (GB)
NFVO	2 x 2.19 GHz	8	40
MEAO	1 x 2.19 GHz	1	10

ETSI MEC specifications. Further references to fog05 in this paper will mainly refer to its MEAO functionality (and not its VIM functionality).

- *Monitoring System*: To retrieve runtime information from the deployed components, a monitoring component has been added to our experimental testbed. To that end, we used Zabbix [16] (release 4.4) as monitoring engine and Grafana [17] (version 6.4.1) for visualizing the resource utilization of the collected metrics.

The resources allocated to each orchestrator of our experimental testbed are summarized in Table I. For the NFVO, we have assigned the resources as recommended in the official documentation of the release SIX [18] of this platform. On the other hand, we have assigned less resources to the MEAO orchestrator, since fog05 is often meant to be deployed directly within in edge infrastructures, which are often expected to be more limited in terms of computing, memory, and storage. The MTO, where applicable, was deployed on a separate host, as it is expected to reside next to (or inside) the OSS/BSS, without expecting any significant overhead, as it is an extremely lightweight Cloud-hosted proxy.

IV. EVALUATION

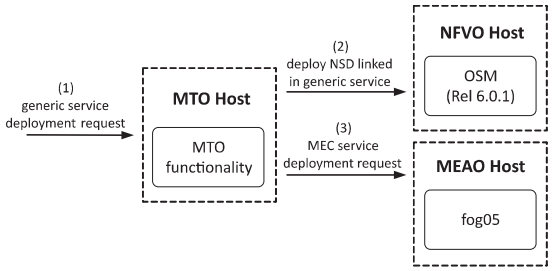
To validate the proposed solution we have implemented the system described in the section III using a layered API. Using our implementation along with the third-party software mentioned in subsection III-C, we could take measurements not only for our MTO-based solution, but also for scenarios that simulate the cases in which no MTO exists, but the NFVO or the MEAO act as a master, as implied in the ETSI study.

A. Scope and setup

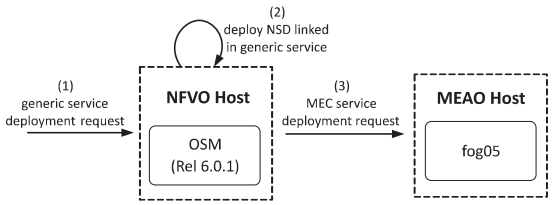
This subsection describes the metrics, the variables, and other settings that were applied in order to perform a comparison of the different MEC-NFV orchestration approaches.

Compared Approaches: The same experiments have been performed for three different approaches. The first one represents our solution while the two others represent two possible cases implied in the related ETSI study [6]. More concretely, the compared approaches will be referred to as:

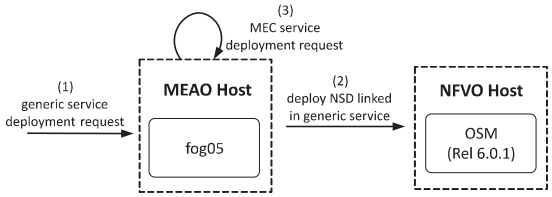
- *MTO-based*: This solution includes our Multi-Tier Orchestrator (MTO) as described in Section III, namely it receives high-level requests for deploying (generic) services that require both NFV- and MEC-layer orchestration, and implements the workflow of Fig. 2 while using OSM and fog05 for the NFVO and MEAO logic, respectively.
- *NFVO-mastered*: In this approach, the NFVO receives the high-level requests and handles the NFV-related service instantiation while triggering the MEAO for applying the MEC-related parts of the descriptor.
- *MEAO-mastered*: In this approach, the MEAO receives the high-level requests and triggers the NFVO for handling the NFV-related service instantiation while executing its tasks based the MEC-related parts of the descriptor.



(a) MTO-based scenario



(b) Scenario 2: NFVO-mastered scenario



(c) Scenario 3: MEAO-mastered scenario

Fig. 3. Network topology scenarios for the three compared approaches.

Fig. 3 represents the very high-level architecture and flow of actions that take place in each of these three scenarios.

Metric: The measured metric is the *Maximum CPU load* of the Virtual Machines (VM) that host the NFVO and the MEAO during the time that it takes to instantiate a certain number of services. The CPU load of the VMs is quite stable during most of this time, but we focus on the peaks, because it is them that can lead to overloads and outages.

Inputs and variables: The main setting that has been varied was the *number of services* that were given to the system concurrently in order to be instantiated. This number has been varied between 1 and 15, because this range was sufficient for demonstrating significant CPU loads of the CPU hosts. With regard to the more concrete inputs, our experiments actually use copies of exactly the same service, which includes a simple NSD along with a simple associated AppD, which are bundled together. As shown in Fig. 5, the NSD consisted of two sample VNFs, which had no real telecom functionality but included

Fig. 4. Example details of resource utilization in the NFVO host during evaluation of the MEAO-mastered approach.

one virtual interface each, which was used as a connection point, while the connection points of the two VNFs were connected with a virtual link. On the other hand, the sample AppD included only identification infromation (name, version, description etc.) and a few directives, e.g. for supporting Radio Network Information Services (RNIS) and publish-subscribe (PubSub) MQTT transport based on oath2 authentication (cf. Snippet 1).

Controlled variables: The *host resources* were fixed to the values of Table I (refer to subsection III-C for justification), while the *additional load* of the hosts during the experiments was as close to zero as a VM can get in its idle state.

Measurement method and details: The used monitoring environment (Zabbix and Grafana) produced detailed logs, some of which were used to produce our selected evaluation results (i.e., those related to the Maximum CPU load for the different approaches). Further aspects that were measured by the environment (but not discussed in detail in this paper) can be understood by looking at Fig. 4. This is an example output of our monitoring environment, in which the CPU usage over time (across all service instantiations of a scenario), along with the incoming and outgoing traffic, are visualized for the NFVO host in the MEAO-mastered approach with 7 service instantiations.

Snippet 1. Sample AppD used for all service instantiations of the evaluation

```
{
  "appDId": "example-meapp1",
  "appName": "example_mec_application",
  "appProvider": "ADLINK",
  "appDVersion": "1.0",
  "appSoftVersion": "",
  "mecVersion": ["1"],
  "appDescription": "Simple_MEC_Application",
  "appServiceRequired": [],
  "appServiceOptional": [],
  "appServiceProduced": [
    {
      "id": "0",
      "serName": "Wifi_RNIS",
      "serCategory": {
      "href": "http://rnis.in/the/catalog",
      "id": "RNIS",
      "name": "RNIS_Services",
      "version": "1.0"
      },
      "version": "0.1",
      "transportSupported": [
         {
           "id": "0",
           "transport": {
             "type": "MB_PUBSUB",
             "protocol": "MQTT",
             "version": "5.0",
             "security": {
               "oauth2_info": "None",
               "grant-types": ["
                   OAUTH2_IMPLICIT_GRANT"],
               "token-endpoint": "None"
             }
           },
         "serializers": ["JSON"]
         }
      ]
    }
  ],
  "appFeatureRequired": [],
  "appFeatureOptional": [],
  "transportDependencies": [],
  "appTrafficRule": [],
  "appDNSRule": [],
  "appLatency": {
    "timeUnit": 10,
    "latency": "ms"
  }
}
```

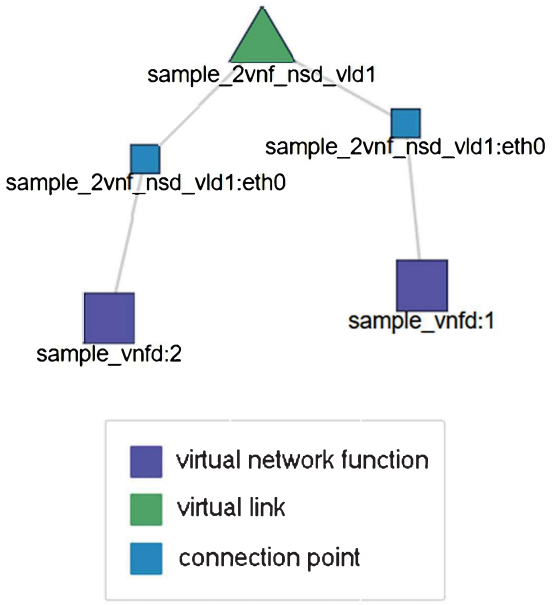


Fig. 5. Visualization of the NSD used in the experiments.

B. Maximum CPU Load

This subsection presents and assesses the experimental results related to the maximum CPU load on the NFVO and MEAO hosts involved in the different approaches.

Fig. 6 shows the maximum CPU load for each host in the MTO-based scenario when the number of concurrent service instantiations increases from one to fifteen. Here we can corroborate that the NFVO host exceeds the MEAO in terms of the maximum CPU utilized despite the fact that significantly more resources had been given to it (see Table I). The reason for this is that the NFVO trigger a bigger number of internal elements and procedures in order to fulfill its tasks and hence the host of the OSM has a bigger footprint. Another aspect that is notable about Fig. 6 is that the CPU load of the MEAO host does not increase in tandem with the number of service instantiation requests. Thus, it may be inferred that the number of resources assigned to the MEAO server was sufficient to effectively handle the service dependencies of each AppD requested throughout the execution of this scenario.

Fig. 7 shows the results when the NFVO is acting as the master of service deployment, which means that it receives the high-level requests and handles the NFV-related service instantiation by itself. On one hand, as expected, the maximum CPU load in the NFVO host is higher than in the MTO-based scenario. Specifically, the observed differences are up to 10% or more. Therefore, this is an indication that for a higher number of requests, the CPU load of the NFVO host in the NFVO-mastered scenario is going to reach critical levels sooner than in the MTO-based scenario. On the other hand, the CPU load of the MEAO host maintained stable values

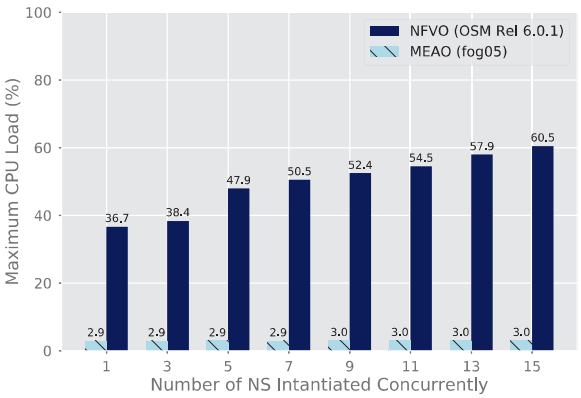


Fig. 6. Maximum CPU load in the MTO-based scenario.

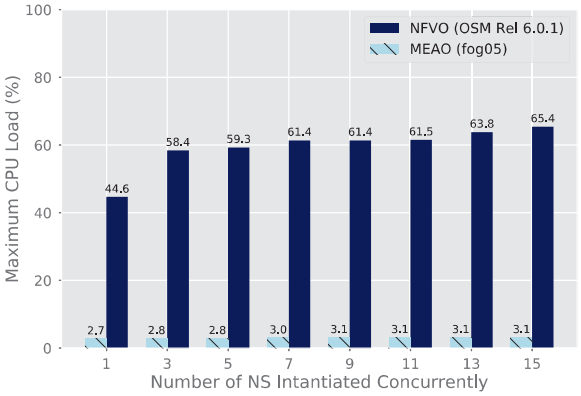


Fig. 7. Maximum CPU load in the NFVO-mastered scenario.

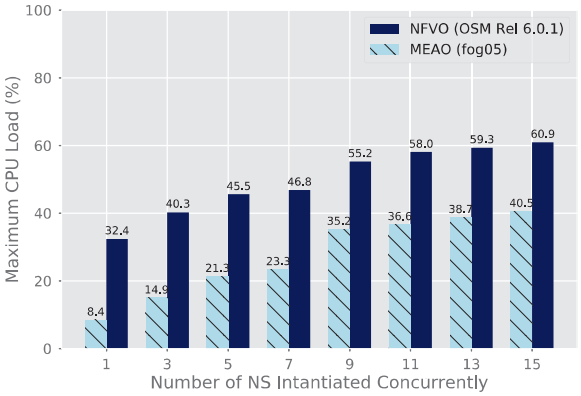


Fig. 8. Maximum CPU load in the MEAO-mastered scenario.

since this orchestrator is performing the same functions as in the MTO-based scenario when a generic service is instantiated (i.e. just handling MEC-related parts of the descriptors).

The CPU loads for the case where the MEAO includes the master functionality are plotted in Fig. 8. These results show that the maximum CPU load of the MEAO host increased between approximately 3 and 13 times compared to the previous two scenarios. The maximum difference attained was of up to 37.4% when the MEAO host receives 15 concurrent service instantiation requests.

Furthermore, Fig. 8 makes visible how, in the MEAO-mastered scenario, the maximum CPU load measured for the MEAO host continually grows as the number of instantiated generic services increases. This behaviour is significantly different compared to the other scenarios, where the CPU load of the MEAO host remained almost negligible across all requests. This very issue might lead very fast to bottlenecks for the MEAO-mastered scenario, which is the main option of the related ETSI study, especially when the MEAO is hosted in edge infrastructure with limited resources.

Although this novel functionality can be implemented on the same host of the NFVO or MEAO, we do believe however that deploying the Multi-Tier Orchestrator on a separate server will improve the scalability of the orchestrators. Moreover, it will also offload the CPU consumption in the orchestrator hosts, a characteristic that is crucial in practical applications.

V. Conclusion

This paper has investigated the issue of integrating multiple service orchestrators that belong to different layers, namely they provide service orchestration functionalities that are different but inter-related. The most common such case, namely the integration of NFV and MEC orchestrators, has been discussed. In this context, a solution based on a multi-tier orchestration component, which coordinates the basic action flow of the two involved orchestrators, has been presented and compared to solutions implied in a relevant ETSI study. The latter solutions are based on one of the existing orchestrators, typically the MEAO, playing the role of the master.

Some advantages of the proposed solution are related to better programmability and separation of concerns, since none of the existing orchestrators needs to be extended with complex knowledge about other tiers. However, the gains of the multi-tier orchetrator can be also related to performance. With this regard, our evaluation has demonstarted how the CPU load of the MEAO host can grow dangerously in current solutions in which the MEAO acts as a master, as well as how this is alleviated by deploying our multi-tier orchestrator on top of the MEAO and the NFVO. The CPU loads using our solution were up to 13 times lower than in the MEAO-mastered approach when instantiating 15 services concurrently in our testbed. This can grow even bigger if more service instantiations are performed or if the MEAO host has less resources, which is a possible scenario in edge environments.

Future enhancements of the multi-tier orchestrator solution shall add the capability to trigger more types of orchestrators, e.g., Cloud-native. This requires the revision of the service instantiation workflow which was presented in this paper, as well as architectural extensions. Further, experiments with more complex services and MEC platforms could reveal the exact circumstances under which our approach should be preferred compared to master-based approaches.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 761508 (5GCity project) and the Spanish national project 5GCity (TEC2016-76795-C6-2-R).

References

[1] 3GPP, "Study on Management and Orchestration of Network Slicing for Next Generation Network," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 28.801, Jan. 2018, version 15.1.0, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3091.

[2] NGMN, "Description of Network Slicing Concept," Next Generation Mobile Networks alliance, Tech. Rep., Jan. 2016, https://www.ngmn.org/fileadmin/user_upload/160113_Network_Slicing_v1_0.pdf.

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[4] A. Galis and K. Makhijani, "Network Slicing Landscape: A Holistic Architectural Approach, Orchestration and Management with Applicability in Mobile and Fixed Networks and Clouds," in *IEEE Network Softwarization (NetSoft)*, Jun. 2018.

[5] S. Sharma, R. Miller, and A. Francini, "A cloud-native approach to 5g network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 120–127, 2017.

[6] ETSI, "Mobile Edge Computing (MEC) - Deployment of Mobile Edge Computing in an NFV Environment, ETSI GR MEC 017 V1.1.1," 2018. [Online]. Available: www.etsi.org/deliver/etsi_gr/MEC/001_099/017/01.01.01_60/gr_MEC017v010101p.pdf

[7] G. Baldoni et al, "Edge Computing Enhancements in an NFV-based Ecosystem for 5G Neutral Hosts," in *5GNetApp Workshop at the IEEE Conference on NFV and SDN*, Nov. 2018, pp. 1–5.

[8] G. Cattaneo, F. Giust, C. Meani, D. Munaretto, and P. Paglierani, "Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case," *Computers*, vol. 7, no. 4, p. 55, 2018.

[9] M. C. Filippou, D. Sabella, and V. Riccobene, "Flexible mec service consumption through edge host zoning in 5g networks," *arXiv preprint arXiv:1903.01794*, 2019.

[10] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "Cds-mec: Nfv/sdn-based application management for mec in 5g systems," *Computer Networks*, vol. 135, pp. 96–107, 2018.

[11] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[12] BBF, "TR-384: Cloud Central Office Reference Architectural Framework," The BroadbandForum, Technical Report (TR), Jan. 2018, issue 1, https://www.broadband-forum.org/download/TR-384.pdf.

[13] R. Muñoz, R. Vilalta, R. Casellas, R. Martínez, F. Vicens, J. Martrat, V. López, and D. López, "Hierarchical and recursive nfv service platform for end-to-end network service orchestration across multiple nfvi domains," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 2018, pp. 1–5.

[14] ETSI, "Network Functions Virtualisation (NFV) - Terminology for Main Concepts in NFV, GS ETSI NFV 003 v1.4.1," 2018. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.04.01_60/gs_NFV003v010401p.pdf

[15] A. Corsaro and G. Baldoni, "fog5: Unifying the computing, networking and storage fabrics end-to-end," in *2018 3rd Cloudification of the Internet of Things (CIoT)*, July 2018, pp. 1–8.

[16] Zabbix . Zabbix monitoring manuals, available at: https://www.zabbix.com/manuals. Accessed on 23/10/2019.

[17] Grafana Labs. Grafana documentation, available at: https://grafana.com/docs/. Accessed on 23/10/2019.

[18] Open Source MANO release six. (2019) Available at: https://osm.etsi.org/wikipub/index.php/OSM_Release_SIX. Accessed on 23/10/2019.