Programmable and Flexible Management and Orchestration of Virtualized Network Functions

Hadi Razzaghi Kouchaksaraei, Sevil Dräxler, Manuel Peuster, Holger Karl
Computer Network Group
Paderborn University, Paderborn, Germany
{hadi.razzaghi, sevil.draexler, manuel.peuster, holger.karl}@uni-paderborn.de

***Abstract*—Supporting the vast variety of network services' management and orchestration requirements is one of the main challenges that Network Function Virtualization (NFV) is dealing with. While general management requirements such as Virtual Network Function (VNF) resource requirements can be specified by the service developers using service descriptors, specific management operations like VNF-specific configuration cannot be performed by these descriptors. On the other hand, it is inefficient and also very challenging for Management and Orchestration (MANO) frameworks to provide all specific-management operations for every individual network service and their constituent VNFs. To mitigate this issue, we propose the use of service-specific programs called Specific Managers (SMs) that can customize management and orchestration of network services and also extend the capability of MANO frameworks to support per-service management and orchestration. The results of our evaluation show that the higher flexibility and programmability enabled by SMs improve the performance of the service performance and also utilises the service provider resources more efficiently.**

I. Introduction

Managing the lifecycle of network services and orchestrating them are critical challenges on the way to Network Function Virtualization (NFV). These challenges come from the diversity of network services with a wide range of operational and deployment requirements that need to be supported by service operators.

It is crucial to have a sophisticated lifecycle management in NFV as it affects service performance and also resource consumption. Using a generic management process for managing all network services and ignoring their specific management requirements (e.g., service-specific placement) make the network services inefficient. Therefore, service operators need to manage services based on their specific requirements and that requires these operators to fully understand the specific requirements of all individual network services.

This understanding can be obtained from the service developers as they are the best source of knowledge when it comes to the requirements of network services. To this end, service descriptors are used to communicate knowledge from developers to operators. These descriptors are created by the service developers and shipped along with other service artefacts to service operators. Using these descriptors, the service operator's Management and Orchestration (MANO) framework can manage and orchestrate network services considering individual network service requirements.

978-1-5386-1478-5/18/$31.00 ©2018 IEEE

Although these descriptors simplify network service management, their rigidity limits programmability and flexibility of network service management and orchestration. Descriptors can be used by service developers to specify general requirements of services such as the resource requirements of a Virtualised Network Function (VNF) or the service graph of a network service. However, anything outside the predefined semantics of the descriptor cannot be realised; examples are service-specific management and orchestration operations like VNF-specific configuration or service placement with a specific optimisation goal.

In principle, it might be possible to have a very broad, comprehensive semantic of descriptors. However, supporting such descriptors is very challenging for the MANO framework. For example, it might be possible to extend the semantic of descriptors to specify service requirements like VNF-specific configuration, but a MANO framework cannot configure each VNF individually as each VNF might have a very specific configuration process that cannot be pre-implemented in the MANO framework. Therefore, the orchestration process also needs to be programmable, just like the network itself has become programmable.

To realise such programmability for the management and orchestration process, we propose the use of Specific Managers (SMs) inside an orchestration framework. SMs are management programs that are developed by service developers and are transmitted along with other service artefacts to the service operators. Using these programs, service developers can customize the management and orchestration of network services and also extend the capability of MANO frameworks to support complicated management scenarios. Architecturally, this evolves so-far monolithic orchestration platforms into a microservice-based system, reaping all the known benefits of this architecture pattern [1] over other concepts.

We implemented the SM concept in SONATA [2] (a platform for agile service development and orchestration) enabling the service developers to create, deploy, and run SM components. Moreover, using the network service chains placement as a test case, we showcased the benefits of programmable MANO frameworks.

The rest of this paper is organised as follows. First, we present related work in Section II. In Section III, we introduce the SM concept and explain how the SONATA MANO framework supports it. In Section IV, we discuss SM use cases.

Our evaluation results are depicted in Section V and finally, in Section VI, we highlight our conclusions.

II. Related Work

The European Telecommunication Standards Institute (ETSI) NFV group, for the first time, defined a reference architecture for management and orchestration of NFV services [3]. In this architecture, Element Management Systems (EMSs) are proposed to be used for providing specific management for one or multiple VNF(s). Using EMSs, the ETSI architecture allows service providers to program the management of VNFs. However, the ETSI architecture does not allow service developers to customise the management and orchestration of network services, which results in increasing the time to market of network services as the service provider needs to receive the service requirements from the developer, develop the specific EMS, and then deploy the service. It also lacks an approach for programming and customising the orchestration of networks services (e.g., service chain placement).

Following the definition of the ETSI architecture, various MANO frameworks have emerged for managing and orchestrating network services. In the following, we review some of these frameworks and discuss their capabilities to support per-VNF/service management and orchestration.

Tacker [4], the OpenStack NFV platform, has a policy-based approach for network service management and orchestration. For example, to scale and place network services, Tacker uses a monitoring component that triggers scaling based on strategies (e.g., CPU usage threshold) defined for each VNF or network service. On the other hand, each VNF/service has its specific policies (e.g., max/min number of instances for scaling out/in) specified in the VNF/service descriptor that describes how the scaling/placement should be performed. Although with this approach, the general VNF/service requirements can be considered, the pre-implemented lifecycle management modules (e.g., scaling and placement) are fixed and cannot be extended/programmed to support service-specific lifecycle management operations (e.g., VNF configuration during the runtime) of VNFs/services.

T-NOVA [5] is an NFV platform that provides a MANO framework to manage and orchestrate network services over virtualized infrastructures. T-NOVA's MANO framework employs a dedicated resource mapping module that is responsible for network service scaling and placement. This module contains some generic optimisation algorithms, which includes minimising mapping cost, maximising provider's revenue, and maximising the acceptance rate of NFV service requests. It is feasible for service operators to integrate new optimisation algorithms with different objectives into this module. However, service developers cannot influence the algorithms of the resource mapping module, and all network services are limited to the predefined resource mapping optimisation.

ONAP's [6] solution for providing per-VNF/service management is POLICY. POLICY is the ONAP subsystem that maintains, distributes, and applies a set of rules to ONAPs underlying control, orchestration, and management functions. Service developers can create a set of policies for each VNF/service during service creation. These rules are then injected into the MANO framework to customise the management of network services. Although ONAP's POLICY can support per-service management to some extent, its expressiveness is limited to the given policy language and does not allow a service developer to actually program the MANO framework. POLICY uses the YAML language to describe the policies, which does not provide any programming capability to extend the MANO framework for supporting complicated management scenarios.

OSM [7] employs TOSCA-based templates [8] for describing both service and VNF requirements. To perform per-VNF management, OSM uses Juju [9], which enables service developers to write scripts for configuring/reconfiguring their VNFs based on their specific needs. However, OSM misses a strategy for providing per-service orchestration on the service level, which makes customising orchestration tasks such as service placement or scaling rather challenging.

Cloudify [10] uses the blueprint, a TOSCA-based template, to describe service requirements. In the blueprint, lifecycle events are described in a YAML format, which is inflexible as in the other platforms. In Cloudify, developers can use workflows to customize the management of VNFs. Workflows are described in Python and use dedicated APIs to communicate with other components of the MANO framework. But workflows can only customise the management of VNFs and are unable to customize the orchestration of network services. Also, workflows have to be written in Python and cannot be implemented in any other programming language.

III. Specific Manager

The rigidity of conventional MANO frameworks comes from their monolithic architecture. Monolithic MANO frameworks are tedious to scale and extend as adding new features requires the whole framework to be reconfigured and tested. These issues can be mitigated using the microservice architecture, which breaks down monolithic frameworks into microservices. The microservice architecture allows MANO framework tasks to be implemented as a set of loosely-coupled functional blocks (microservices). This increases the flexibility of MANO frameworks since adding new features can be done by simply adding and integrating new microservices, eliminating costly and time-consuming testing and reconfiguring an entire framework. Having such a flexible architecture enables service-specific lifecycle management microservices that can be injected into the MANO framework to replace the generic and pre-implemented lifecycle management microservices of the MANO framework.

Specific Managers (SMs) are the realisation of this idea. SMs are microservices that can be injected into MANO frameworks to complement its ability for supporting specific requirements of network services as they can implement arbitrary decision logic. There are two types of SMs – (i) Function-Specific Managers (FSMs) and (ii) Service-Specific Managers (SSMs) – explained as follows.

- Function-Specific Managers (FSMs) are microservices that are developed to provide specific requirements of individual VNFs. FSMs extend a MANO framework's VNF-specific lifecycle management (see use cases in Section IV).
- Service-Specific Managers (SSMs) are service-specific microservices that provide specific orchestration requirements of network services. SSMs improve a MANO framework's programmability for orchestrating network services as a whole (see use cases in Section IV).

A. Specific Managers in the ETSI NFV Reference Architecture

As also discussed in Section II, the functional blocks in the ETSI NFV reference architecture [3] closest to SMs are EMSs. EMSs work as an intermediate component between VNF managers and VNFs to provide VNF-specific management. Fig. 1 shows the extended version of the ETSI NFV reference architecture, which includes the SSM/FSM components and their relationship with other components. In this architecture, we replaced EMS components with FSMs, which can replace the generic VNF managers. On the higher level, we added SSM components that are specific to each service and communicate with the MANO orchestrator and also their underlying FSMs to provide service-specific orchestration.

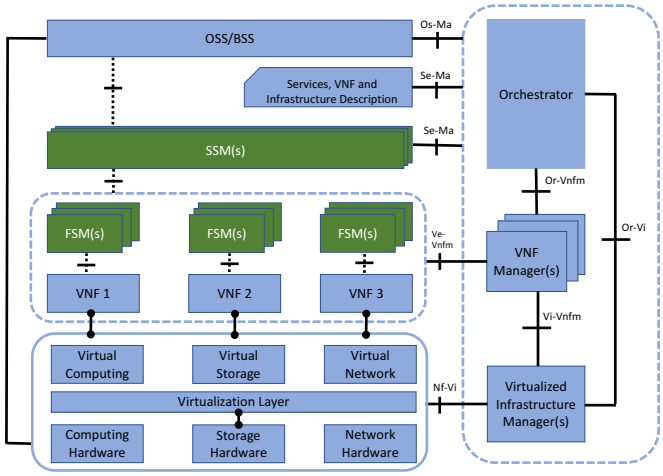


Fig. 1: Specific Managers in the ETSI NFV reference architecture

B. Specific Manager Support in SONATA

The SONATA service platform, which manages and orchestrates NFV services, is based on the microservice architecture in which the MANO tasks (service deployment, management, placement, etc.) are implemented in container-based microservices. These microservices communicate with each other through a message bus in the publish/subscribe fashion. To this end, APIs are defined for each task, which can be subscribed by microservices to send and receive requests and response messages, respectively. For example, the service placement API is subscribed by placement and lifecycle management microservices. The lifecycle management microservice publishes placement requests to this API, and, on the other side, the placement microservice, which is listening to this API, receives the request, reacts accordingly, and sends the response back to the same API to be received by lifecycle management microservice. This enables microservices to be loosely coupled, which provides a high level of flexibility.

In SONATA, SM components are also implemented as container-based microservices. SMs metadata such as ID, URI, and objectives are specified in the service and VNF descriptors, which is used by the SM lifecycle manager microservice (a SONATA microservice for managing SM containers) to onboard and instantiate SM containers and connect them to their corresponding APIs. For security reasons, SMs do not directly communicate over MANO's main message bus. Instead, they communicate with MANO microservices through their service-specific message bus, which is created during the SM's instantiation. A unique credential is assigned to each message bus that is given to all SMs belonging to the same services. This isolates SM messages belonging to a particular service from SM messages of other services, which prevents SMs from injecting malicious actions into the MANO framework or intrude on the management of other services. As it is shown in Fig. 2, SMs can also be deployed on the service developer's premise, which reduces the risk of running arbitrary code on machines that run crucial operational code. It also reduces the computational overhead that can be caused by deploying a large number of SMs on the premises of network service providers, at the price of increased latency.

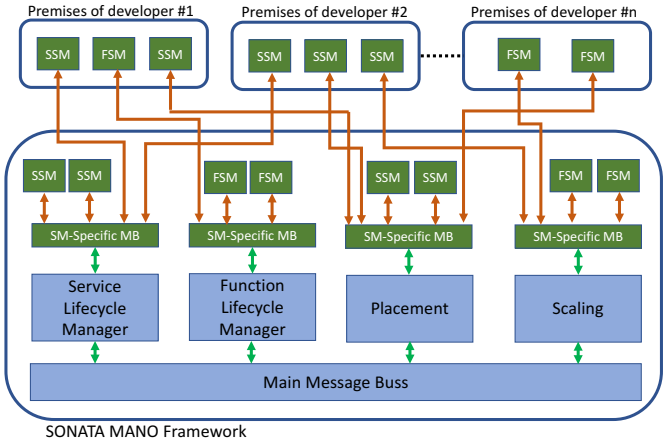


Fig. 2: Specific Managements in SONATA architecture

IV. Specific Management Use Cases

There are a variety of uses cases for both service and function SMs. In this section, we describe some of these use cases that are already validated by SONATA pilots [11].

A. Function-Specific Managers Use Cases

- Configuration FSM: The VNF developer can provide specific configuration operations to be triggered during VNF runtime. For example, start or stop some VNF

functionalities at a particular event (e.g., when it is required by the other VNF in the services chain or based on monitoring triggering) or run specific commands (e.g., installing rules in a virtualised firewall to drop some packets during the runtime) in the VNF.
- Monitoring FSM: Specify monitoring metrics of a particular VNF. It can communicate with the main monitoring component of the MANO framework, request particular monitoring data, and react accordingly.
- Scaling FSM: Defines how to scale a specific VNF. For example, a developer can specify how many instances of a particular VNF should be instantiated at a certain event during VNF runtime.

B. Service-Specific Management Use Cases

- Placement SSM: Maps functions to resources. For example, a developer can provide a sophisticated placement algorithm with optimisation goals that are not supported by the MANO framework (see Section V).
- Scaling SSM: Customise the scaling of network services. Scaling of a single VNF of a network service can impact the other VNFs involved in the same network service. To deal with consequences of scaling, a scaling SSM can be used. Scaling SSM can provide all the required service-specific orchestration functionalities caused by scaling one VNF (e.g., triggering the scaling of another VNF in the service chain).
- Service Chaining SSM: Another use case of SSMs is to take care of changes in the service graph during the service runtime, e.g., compensating for the failure of a VNF in a service chaining SSM.

V. Evaluation

To evaluate the benefits of programmability that SMs can bring to a MANO framework, we compare an SM-enabled MANO framework with others, using placing network service chains as test case. Conventional MANO frameworks employ pre-implemented placement algorithms that cannot be modified or extended to support specific requirements of individual network services. Therefore, all services – no matter what requirements they have – are placed in the same way, with the same optimisation algorithms and objectives. For example, latency-sensitive network services cannot be efficiently deployed using a MANO framework that does not support VNF placements minimizing total latencies.

To support this claim, we designed three MANO frameworks that place network services using different approaches. Utilizing these MANO frameworks, we deployed two services with contradicting optimisation goals on a network with 12 nodes and 42 edges based on the Abilene network from SNDlib [12]. Services are manually created in the form of a deployment request as explained in [13] and deployed in different network background loads. The network load is the percentage of network nodes' and links' capacity that is being used. We chose services based on the ETSI NFV use cases [14], including fixed access and mobile core networks. According to the requirements mentioned in [14], we defined the service's optimisation goals as follows.

- Fixed Access Network: minimising the number of used nodes as the primary objective and minimising total latencies as the secondary objective.
- Mobile Core Network: minimising total latencies as the primary objective and minimising the number of used nodes as the secondary objective.

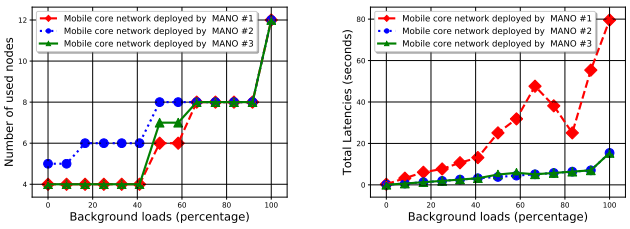
Having this scenario, we compared SM-enabled MANO frameworks with others as follows. Each test was performed five times with different service source/destination nodes, and the average results is shown in the figures.

A. Specific Managers vs. Single-Objective Placement

The placement approaches used in the first comparison for each MANO framework are as follows.

- MANO #1: minimises the number of used network nodes when placing services.
- MANO #2: minimises total latencies when placing services.
- MANO #3: minimises a service's desired objective (based on SM).

We evaluated the resulting service placements based on the two natural metrics, the number of used nodes to deploy the services and the total latency of the services. Fig. 3a and Fig. 3b show the number of used network nodes and total latencies that we obtained for each service, respectively. The results show that while MANO #1 can meet the primary optimisation requirement of the fixed access network the best in all network loads, it is inadequate for providing the secondary optimisation requirement. The same happens for the mobile core network service when MANO #2 is used for deployment. Considering the results obtained for both metrics, however, MANO #3 performs the best as services deployed by MANO #3 use a low number of network nodes plus having low latencies.



(a) Number of used nodes by the fixed access network service

(b) Total latency of the mobile core network service

Fig. 3: Results of service deployment using MANO frameworks with different placement approaches on different network background loads based on the first scenario

In high background load, the number of used nodes become almost the same for the fixed access network service deployed by all MANOs. This is to be expected as there are not many free nodes, and the placement algorithm does not have many

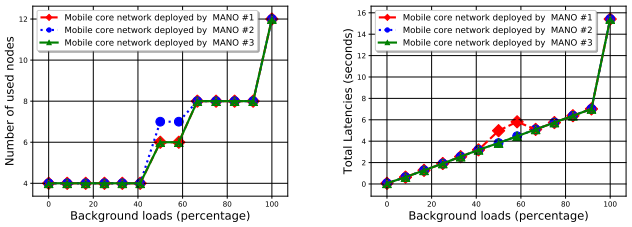
choices to place the services. On the other hand, for the mobile core network service, MANO #1 results in the worst performance even in high background load. This is because MANO #1 does not consider the service latency and just tries to decrease the number of nodes in all network loads. This results in choosing the nodes that generate higher latencies.

B. Specific Managers vs. Multiple-Objective Placement

In the second comparison, we changed the optimisation objectives used in MANO frameworks number one and two as follows, making them smarter and closer in behaviour to our SM-based MANO system.

- MANO #1: minimises the number of used network nodes when placing services, use total latency as a tiebreaker.
- MANO #2: minimises total latencies when placing services, use number of used nodes as a tiebreaker.
- MANO #3: minimises a service's desired objective (based on SM) – as above.

Like the first comparison, we have two metrics including the number of used network nodes to deploy services and the total latency of the services to evaluate the placement performed by different MANO frameworks. Comparing the results illustrated in Fig. 4a and 4b, we can see that the SM-enabled MANO framework is the only framework that gives the best results for both services in all background loads. Although MANO #1 performs the same as MANO #3 for deploying the fixed-access network service, it is the framework performing worst for deploying the mobile core network service. On the other hand, MANO #2 gives the worst results for deploying the fixed access network service. This is as expected, given MANO and service characteristics.



(a) Number of used nodes by the fixed access network service

(b) Total latency of the mobile core network service

Fig. 4: Results of service deployment using MANO frameworks with different placement approaches on different network background loads based on the second scenario

A significant difference between the results from different MANO frameworks can be seen at medium network loads. This is because, in low network load, the placement algorithm has plenty of options to accurately map the services on the resources. Therefore, a good result can be achieved even without a sophisticated optimisation algorithm. On the other hand, when the network is highly loaded, the placement algorithm has little manoeuvring space. Therefore, the results of any placement algorithm will be the same. The use of a service-specific placement becomes advantageous when the network operates at medium background load. But as this is indeed a practically relevant operational regime (networks should be reasonably utilised but not overloaded to be commercially feasible), our SM-based approach shines when it matters.

VI. Conclusion

In this paper, we have presented an innovative method to perform per-service management and orchestration for NFV services. The proposed method is to inject microservices that can perform service-specific management and orchestration into a MANO framework. These microservices are instantiated by MANO frameworks in conjunction with their corresponding services and replace the generic lifecycle management components of MANO frameworks.

Evaluating this method, we showed that SM-enabled MANO frameworks are advantageous over rigid MANO frameworks for both service users and providers as it improves service performance, better utilizes resources, and consequently reduces the cost of NFV services.

A possible caveat against such an orchestrator architecture could be a higher overhead in the orchestrator. We are currently working to show that indeed the opposite is the case: A microservice-based orchestrator should have much better opportunities for parallelisation, improving orchestrator response time while at the same time reducing CPU and memory utilization in the orchestration platform.

Acknowledgment

This work has been partially supported by the 5G-PICTURE project, funded by the European Commission under Grant number 762057 through the Horizon 2020 and 5G-PPP programs and the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly Computing (SFB 901)

References

[1] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, 2015.
[2] S. Dräxler et al., "SONATA: Service Programming and Orchestration for Virtualized Software Networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 973–978.
[3] ETSI NFV ISG, "Network Functions Virtualisation (NFV): Architectural Framework," ETSI, Group Specification ETSI GS NFV-MAN 001 V1.1.1, Oct. 2013.
[4] (2018) Tacker. [Online]. Available: https://wiki.openstack.org/wiki/Tacker
[5] T-NOVA, "Overall System Architecture and Interfaces," T-NOVA, Group Specification D2.22, Sept. 2015.
[6] (2018) ONAP. [Online]. Available: https://www.onap.org/
[7] OSM, "OSM RELEASE THREE A TECHNICAL OVERVIEW," ETSI, Group Specification, Oct. 2017.
[8] TOSCA, "Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0," 2013.
[9] (2018) JUJU. [Online]. Available: https://jujucharms.com/
[10] (2018) Cloudify. [Online]. Available: http://cloudify.co/
[11] SONATA, "Definition of the Pilots, infrastructure setup and maintenance report," SONATA, Group Specification D6.1, Oct. 2016.
[12] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *3rd International Network Optimization Conference (INOC)*, 2007.
[13] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions." in *In IEEE 3rd International Conference on Cloud Networking*. IEEE, 2014.
[14] ETSI NFV ISG, "Network Functions Virtualisation (NFV): Use Cases," ETSI, Group Specification ETSI GS NFV-MAN 001 V1.1.1, Oct. 2013.