Service Specific Management and Orchestration for a Content Delivery Network

Thomas Soenen*, Wouter Tavernier*, George Xilouris †, Stavros Kolometsos †, Felipe Vicens ‡,
Einar Meyerson Uriarte § and Shuaib Siddiqui §
*Ghent University - imec, † NSCR Demokritos, ‡ ATOS, § i2CAT Foundation Barcelona - Spain

***Abstract*—Any non-trivial network service requires service specific orchestration to meet its carrier-grade requirements regarding resiliency, availability, etc. How the network service components are mapped on the substrate, how VNFs get reconfigured after a monitored event or how they scale, only network service/function developers know how to execute such workflows to guarantee an optimal QoS. It is therefore of paramount importance that NFV Service Platforms allow developer specified input when performing such life cycle events, instead of defining generic workflows. Within the scope of the SONATA and 5GTANGO projects, a mechanism was designed that allows developers to create and execute Service and Function Specific Managers. These managers are processes, created by the developer, that define service or function specific orchestration behaviour. The SONATA Service Platform executes these managers to overwrite generic Service Platform behaviour, creating developer customised life cycle workflows. We will demonstrate the development, testing and operational execution of these managers by using a Content Delivery Network which requires specific placement and scaling behaviour.**

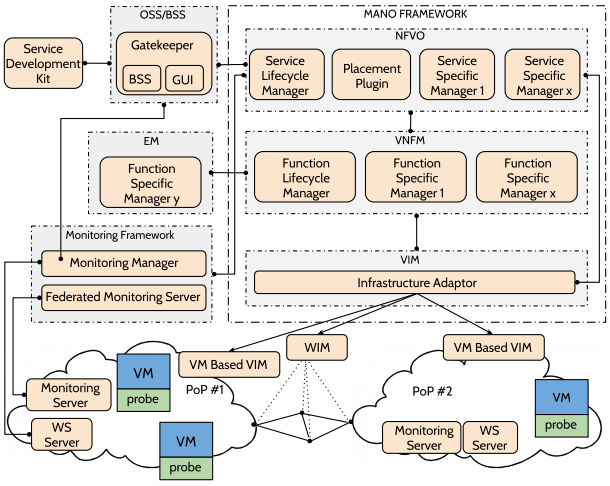


Fig. 1: SONATA ETSI conform Architecture

I. Introduction

Introducing Software Defined Networking and Network Function Virtualization to telecom networks aims, among other targets, to lower the cost of network and service operations. To achieve this, management and orchestration processes must be automated as much as possible. Network services should be mapped and instantiated on-demand, dynamically reconfigured in case of monitored events and terminated upon customer request, all without human intervention from the operator. Automating such management and orchestration workflows in a generic way will not suffice, as any non-trivial network service requires service specific orchestration behaviour for all such workflows to provide a good QoS. Deciding where in the infrastructure VNFs are deployed, when to reconfigure VNFs if their load increases or how to scale the service or VNF influences the QoS of the network service and can only be defined by the developer. As such, mechanisms are needed where automated workflows are combined with developer defined input describing how to orchestrate the network service or function.

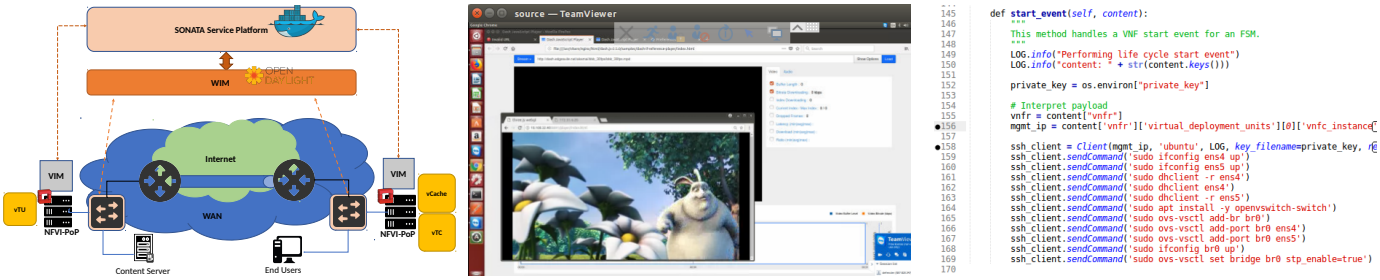
The SONATA [1] Service Platform is an NFV MANO Platform. It incorporates a modular orchestrator that bridges the gap between telecom business needs and its available resources. It automates end-to-end network service instantiation, (re-)configuration and termination workflows by managing and orchestrating the infrastructure. One of its main features is that it allows the inclusion of Service and Function Specific Managers by the service and function developers in their respective descriptors. These managers are processes that are packaged in Docker containers and consume the Service Platform MANO API. They contain service or function specific logic that can overwrite the generic orchestration behaviour of the Service Platform. This allows to highly customise the orchestration on a per service/function basis, allowing to optimise how they are managed at runtime and thus improve their performance and QoS, however defined.

This specific manager mechanism is the feature that we highlight in this demo paper. We will demonstrate how such Service and Function Specific Managers can be developed and tested and how much power and freedom they give the developer in customising the Service Platform by showcasing a Content Delivery Network that uses them. Both specific placement and scaling behaviour are injected into the Service Platform, increasing the QoS of the user. To the best of our knowledge, there are no other MANO platforms available that offer this degree of customisation flexibility.

II. SONATA Service Platform

The SONATA Service Platform[1], see Fig. 1, encapsulates an ETSI conform and modular MANO framework[2] [1]. Its plug-in based design make it flexible and agile, so interoperability with other components and customisation by the operator are easily achieved. All modules are developed as microservices

[1]https://github.com/sonata-nfv
[2]https://github.com/sonata-nfv/son-mano-framework



(a) Demo setup (b) CDN user (c) FSM code segment

Fig. 2: Prototype illustrations.

and deployed as individual Docker containers. Network service workflows such as instantiation, (re-)configuration and termination are implemented by the NFV Orchestrator (NFVO) through the Service Lifecycle Manager (SLM) microservice, while network function workflows are realised by the VNF Manager (VNFM) through the Function Lifecycle Manager (FLM). To allow network service and function developers to customise these MANO workflows, Service Specific Managers (SSM) and Function Specific Managers (FSM) can be created. These specific managers are processes that incorporate service or function specific orchestration behaviour. They are coded by the developer, which can use the SONATA Software Development Kit (SDK) CLI tools[3] to develop and test them, and packaged in Docker containers. They need to consume the correct MANO Framework API[4], which is facilitated over a RabbitMQ message bus, to be usable. SSMs overwrite generic NFVO behaviour with service specific instructions, while FSMs do the same for the VNFM. When the Service Platform needs to execute a network service life cycle event, it will first check whether one or more SSMs are associated with this service and workflow. If that is the case, the Service Platform will execute this SSM, i.e. Docker container, instead of executing the generic workflow. A similar behaviour can be expected for network function life cycle events and FSMs. SSMs and FSMs only customise the orchestration behaviour for the network service or VNF they are associated with.

For example, when an operator receives a new service request, the service instantiation workflow is executed to deploy it. The first step is calculating a resource mapping of the different network functions and virtual links of the service on the infrastructure substrate of the operator. Both the operator and the developer benefit from controlling the objective of this placement algorithm. With our SSM/FSM mechanism, the operator can decide whether itself or the developer gets this control. The developer can implement his or hers desired placement algorithm that optimises the QoS of the service and package it as a valid SSM by using the SDK CLI tools. When going through the service instantiation workflow, the Service Platform will check if a Placement SSM is associated with this workflow, execute the Docker container and provide it with the required input (i.e. infrastructure topology and descriptors) by sending a message on the message bus. Once it calculated the placement, the container responds with the mapping. For this service, the generic placement algorithm is not invoked.

More complex SSM/FSM structures can be created that provide developers with great Service Platform customisation capabilities. FSMs can configure VNFs. Such an FSM contains logic that can access a VNF instance (i.e. a VM or a container) and change its configuration, such as changing forwarding rules, configuring interfaces or changing the mode of a VNF. FSMs can configure VNFs differently, depending on the input that the FSM receives. This input can be defined by a Configuration SSM, which sends different inputs to each FSM, depending on the required configuration. Such a Configuration SSM receives all monitoring data that is collected by the SONATA Monitoring Framework related to that service. The Configuration SSM can now use this data to optimise the QoS of the network service, e.g. by adding new VNFs and extra interfaces to existing VNFs to scale a network service.

Since these SSMs and FSMs are executed within the Service Platform, they need to consume to correct Service Platform MANO APIs. To aid the developer with this, the SDK provides CLI tools that help developing and testing them standalone. The SSM and FSM docker containers are deployed locally and injected with various mocked payloads, and it is validated whether they respond correctly.

III. Demonstration Scenario

This demo will demonstrate three things:

- The ease of developing and testing SSMs and FSMs by using the SDK CLI tools.
- A Placement SSM overwriting the default placement functionality of the Service Platform.
- A combination of Configuration SSM and FSMs that dynamically scale a Content Delivery Network.

These demonstrations showcase the high degree of Service Platform orchestration customisation that is obtained by the developer by using SSMs and FSMs, and the ease of developing them. The network service that will be used for this demonstration is a Content Delivery Network (CDN). This network connects a user with a content server, and contains three VNF: i) a vCache VM image that caches content, ii) a vTC VM that classifies traffic and iii) a vTU VM that transcodes traffic. For the service to perform optimally, the

[3]https://github.com/sonata-nfv/son-cli

[4]https://github.com/sonata-nfv/son-mano-framework/wiki/ssm-fsm

vCache and the vTC should be located close to the user, while the vTU should be located close to the content server. The vTC analyses the video requests that are made by the user, and informs the vTU of this. If the vTU notices that some of the video requests require an encoding that is not provided by the content server, the vTU will transcode the video.

Our demo setup is as follows (see Fig. 2a). Our testbeds provide two clusters of servers, each representing one datacenter. OpenStack is used to manage the virtualised infrastructure of each datacenter (also referred to as Point of Presence or PoP). The two datacenters are connected in an software-defined WAN, that is controlled through OpenDaylight. One PoP is located in the access network of the content server, the other PoP in that of the users. The SONATA Service Platform, which runs on a dedicated server, orchestrates both PoPs and the underlying network. In this Service Platform, the service packages[5] required for the demo are already available. The SONATA SDK is installed on the same dedicated server. The VM images of the three VNFs are available in the databases of each PoP. Required for this demonstration:

- A power socket
- A secondary monitor
- wired/wireless Internet to connect to the testbeds.

A. Scenario 1: SSM and FSM developing and testing

In order to ease the process of developing SSMs and FSMs, and assuring that they are without errors, we developed an SDK CLI tool *son-sm*[6]. This tool is the subject of this first demo scenario. For this scenario, the vCache VM is instantiated in one of the PoPs. We will (all tasks are performed using *son-sm*):

- create a new FSM template
- add some code to the FSM that configures the vCache (Fig. 2c shows an FSM code snippet)
- generate mocked FSM input based on the characteristics of the deployed vCache
- locally execute the FSM
- access the vCache to check if configuration succeed
- iterate the previous two steps to debug and test the FSM
- package and publish the FSM as Docker container and add it to the descriptor

Hereby, we demonstrate the ease of developing and testing SSMs and FSMs conform the SONATA MANO APIs.

B. Scenario 2: Service Specific Placement

Optimally, the vCache is located on the PoP close to the users to serve them quicker and to relax the core. Requests originate close to the users, so locating the vTC close to them allows for a quicker request analysis. Transcoding traffic near the content server prevents individual transcoding at each PoP, therefore the vTU should be located on the PoP close to the content server. Among other, the Placement SSM receives the following information from the MANO Framework: the location of i) each datacenter, ii) the destination (i.e. the content server) and iii) the source (i.e. the user). the CDN Placement SSM will use this info to determine which datacenter is closest to the content server, and which to the user, and map the VNFs on them accordingly. For this scenario, we will:

- instantiate a complete CDN with a Placement SSM (implemented as described above) through the SONATA Service Platform
- instantiate a CDN without a Placement SSM, that therefore will use the generic placement functionality
- show that the resulting placement is different by accessing the OpenStack dashboards of each PoP
- show that the sub optimal generic placement results in a lower quality of the streamed video

Hereby, we demonstrate how the SSM/FSM mechanism allows the developer to optimise the performance of a service by overwriting the placement functionality.

C. Scenario 3: Dynamic and Custom Scaling

Here, we show the extended capabilities of the SSM/FSM mechanism. We start from a CDN in minimal configuration, i.e. without the vTU, to save resources. Once some transcoding requests are noticed by the vTC, the service will scale up and dynamically add the vTU to the chain. This is realised as follows. The developer defines transcoding requests as a monitoring metric for this service. Once such requests are monitored, the Configuration SSM is informed. This Configuration SSM, as it is created by the developer, instructs the Service Platform to deploy the vTU on the PoP close to the content server and to include it in the chain. It also instructs the vTC and vCache to reconfigure accordingly through there respective Configuration FSMs. During the demo, we will:

- demonstrate how a custom monitoring metric and threshold can be defined by the developer,
- show that crossing this threshold instantiates the vTU and includes it in the service chain and
- show that the vTC starts using the vTU and the received encoding on the user side (accessible through TeamViewer as shown on Fig. 2b) has changed.

Hereby, we showcase the ease by which developers can set up advanced structures that customise the Service Platform's orchestration so that the service scales as desired. It also shows that it can reconfigure chains and add VNFs to services at runtime, triggered by custom monitoring metrics and without the need for human intervention by the operator.

[5]https://github.com/sonata-nfv/son-vcdn-pilot

[6]https://github.com/sonata-nfv/son-sm

ACKNOWLEDGEMENT

Funded by the European Commission H2020 5G-PPP projects SONATA (671517) and 5GTANGO (761493).

REFERENCES

[1] H. Karl *et al.*, "Devops for network function virtualisation: an architectural approach," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1206–1215, 2016.