
Department of Computer Science
Computer Networks Research Group

# Project Plan


Management of ServiCes Across MultipLE clouds

## Authors:

Arkajit Dhar
Ashwin Prasad Shivarpatna Venkatesh
Bhargavi Mohan
Deeksha Mysore Ramesh
Harshitha Pandithanahalli Somashekaraiah
Sanket Kumar Gupta
Suheel Shrirangapura Nazeersab
Vivek Jaganath

## Supervisors:

Prof. Dr. Holger Karl | Sevil Dräxler | Hadi Razzaghi Kouchaksaraei

Paderborn, December 6, 2018

# Contents

# List of Figures

# List of Tables

# 1 Motivation

The rapid growth of mobile data services driven by mobile internet has led to substantial challenges of high availability, low latency, high bit rate and performances in networks. The recent development of Network Function Virtualization (NFV) and Software-Defined Networks (SDN) have emerged as a key enabler for 5G networks. There has been a paradigm shift in networking with the recent developments in network virtualization technology. NFV involves decoupling of the hardware components from the software components of a network function. NFV requires a central management and orchestrations (MANO) framework in order to fully deliver end-to-end services of an application. There are multiple open source as well as industrial implementations of MANO framework in the market.

End-to-end network service delivery with high availability, scalability, and low latency requires chaining of the Virtual Network Functions (VNFs) across different Internet Service Providers (ISPs) which in turn have their own MANO frameworks. In order to seamlessly create a network service by utilizing the VNF within each of the MANO frameworks, the need of interoperability among different MANO frameworks is of utmost importance.

## 1.1 Problem Description

European Telecommunications Standards Institute (ETSI) defines the reference architecture for a MANO framework. Each network service is composed of multiple network functions virtually linked and orchestrated by a MANO framework. The network services require a descriptor which contains the details of all the VNFs, virtual links and forwarding graph of VNFs. Each of the VNFs contains it's own Virtual Network Function Descriptor (VNFD) and the information about the number of virtual machines it requires. Different MANO frameworks have their own descriptor schemata pertaining to the standard defined by ETSI. This framework-specific Network Service Descriptor (NSD) hinders the orchestration and management of VNFs between different MANO frameworks.

Firstly, the project aims at tackling the abov- mentioned problem with the implementation of a translator and splitter engines, which would help translate the NSD and divide the VNFs from different MANO frameworks, thus creating a framework-independent network service chain. Secondly, the project aims at the implementation of an Orchestrator level adaptor, that allows interaction between different MANO frameworks, exposes the network service instantiation interfaces of the underlying MANO frameworks and retrieves monitoring information about the network service status. The adaptor will mainly address MANO scalability challenges and

perform state management. Lastly, the project aims at integrating these individual modules in order to provide for an end-to-end network service delivery, spread between different MANO frameworks, with high availability, scalability, and low latency.

# 2 Goals and Use Cases

In this chapter, the intended goals of the project and some use cases are being discussed.

## 2.1 Goals

The goal of the project is to develop a software suite which facilitates interoperability between MANO frameworks, thereby enabling the management of a network service across multi-vendor environments. To achieve this, the software suite is divided into 3 individual work packages (WPs), which are initially developed in parallel and are merged later. In the following sections, the individual goals of these work packages are explained in detail.

### 2.1.1 Service Descriptor Translator (SDT)

In this work package, a translator engine for a NSD is implemented. The Network Function Virtualization Orchestrator (NFVO) is the main orchestration unit of a MANO framework that manages the lifecycle of a network service. One of the main functions of a NFVO is registering a network service, by onboarding, it's NSD to the network service catalog. In a scenario, where a network service is to be deployed and registered among two different MANO frameworks, having their own NSD schema respectively, our SDT engine would help in translating a NSD schema from one MANO framework to the other framework. To accomplish this, the NSD schema from the network service catalog of the two different MANO frameworks needs to be extracted and passed on to SDT engine. SDT engine consumes the NSD schemas and translates one NSD schema to the other MANO framework's NSD schema. REST interfaces are made available for other services to interface with the SDT.

### 2.1.2 Service Descriptor Splitter (SDS)

In the production environment, a network service is deployed over a vast geographical region spanning multiple domains. In such scenarios, there is a need to split a network service into smaller network services and deploy it over multiple domains. In this work package, a Service Descriptor Splitter (SDS) is implemented which splits the NSD of a network service. SDS takes NSD as an input that contains all the information elements which can be extracted to generate separate NSDs. In the proposed approach, the Service Graph is extracted from the input NSD and is split into subgraphs that result in a separate set of elements such as VNFs, virtual links, forwarding graphs of VNFs etc. Different network graph splitting algorithms are

then evaluated to split the service graph. These sets are considered to generate NSDs according to the domain-specific orchestrators.

### 2.1.3 MANO Adaptor (MA)

A key component of a MANO framework is Virtualized Interface Manager (VIM), which helps in assigning the hardware resources to virtual resources. The MANO framework uses specific adaptors to communicate with their respective VIMs. For instance, Sonata (5.1.2) MANO framework has adaptors for OpenStack (5.2.1) and Kubernetes (5.2.3). However, when there is a spike in the service request beyond the capabilities of a single MANO instance, multiple MANO instances should be instantiated to balance the load. The goal of this work package is to build an adaptor that facilitates interaction between different instances of MANO frameworks, thereby exposing the underlying MANO framework's interfaces and enabling monitoring of the underlying service states.

Apart from the implementation of MA, the plan to investigate MANO scalability challenges (See 3.3.3)

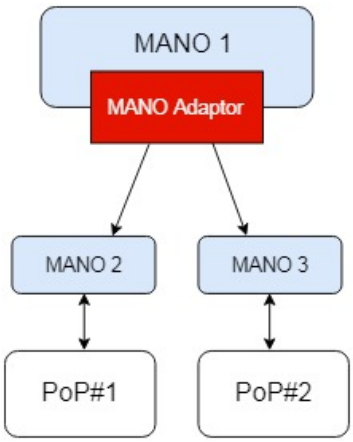


Figure 2.1: This figure visualizes the structure of MA.

### 2.1.4 Integration Of Work Packages

In this work package, the SDT, SDS, and MA are integrated. This integration will augment the functionality of the MA in terms of addressing the scalability challenges between instances of different MANO frameworks. It also extends the capabilities of both the SDS and SDT to split a NSD and then deploy a part of it to a different MANO framework. All these individual modules are planned to be implemented as microservices and integrated keeping their individual autonomy intact.

## 2.2 Use Cases

### 2.2.1 Cross-MANO Framework Interaction

The MANO frameworks used by every network service provider varies from one another. NSD translation enables the deployment of network services that is in accordance with the intended framework.

For instance: Consider two Network Service Operators using different MANO frameworks. One of them uses Sonata framework [DKP+17] and another operator uses OSM framework

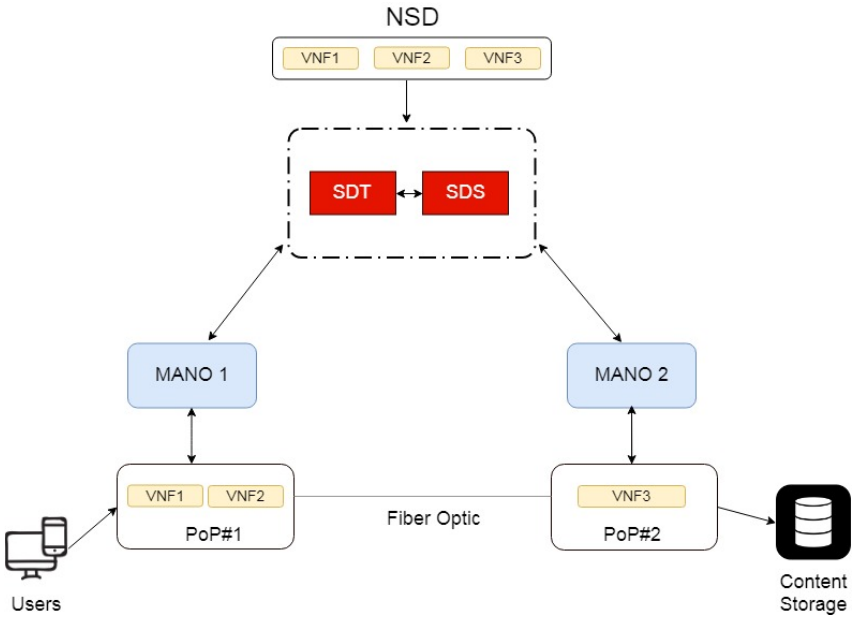


Figure 2.2: This figure visualizes the structure of SDT and SDS.

[Ers13]. These frameworks have different NSD schemata(refer 5.1.2 and 5.1.5). NSD schemata contain VNFs, virtual links, and VNF forwarding graphs and also describes the deployment of a network service. By using a translator, these NSD schemata can be translated into a framework-specific schema. With this, operators can deploy and manage network services across different MANO implementations.

### 2.2.2 Hierarchical Orchestration

By using the MANO adapter, dynamic instantiation of multiple MANO instances and inter-operability between different MANO frameworks can be achieved. The operator will be able to handle the resources in an efficient manner, as one MANO framework can manage a limited number of service requests, operators can explore options to include additional MANO instances under the existing MANO instance to mitigate the traffic load on a single instance. The resources can be provisioned based on the number of requests. This helps the operator in extending their profitability.

**Actors** : The Network Service Providers who would use features of SCrAMbLE.

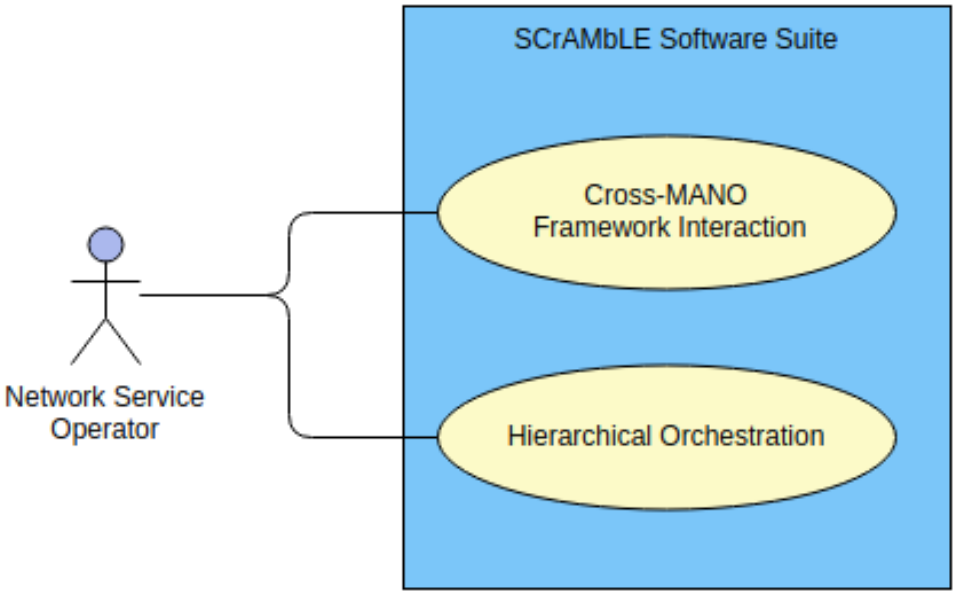


Figure 2.3: Use Case Diagram

# 3 Work Package Tasks

In this chapter, the high level tasks in each WP is discussed briefly. A reference architecture diagram is also given for each WP, this is meant to be seen as an early proposal and it is subjected to change in future documents.

## 3.1 WP1: Service Descriptor Translator

### 3.1.1 Requirements definitions and architecture design

The development of the SDT engine follows a microservice architecture. The architecture inludes a message broker with a publish/subcribe model. The NSDs(yaml files) corresponding to the communicating MANO are published to the broker. Our SDT engine, subscribed to the broker, consumes the yaml files and translate one MANO NSD to the other template and publish them back to the broker. These translated NSDs could be accessed by any other services through a REST API.

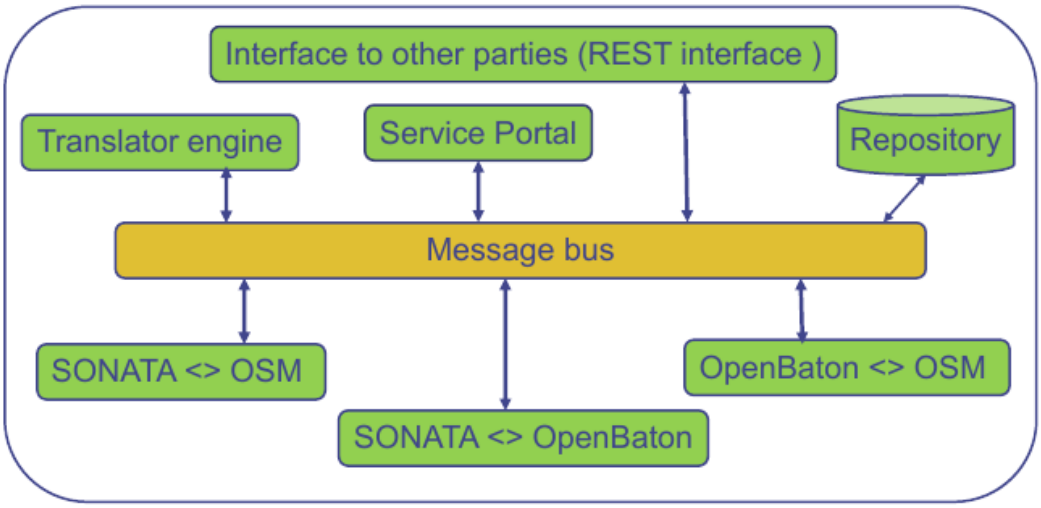


Figure 3.1: Reference architecture for SDT [PDHK]

### 3.1.2 Prototype implementation of SDT components

After the concrete architecture is planned, a working prototype of the SDT engine including all the requisite components is modeled. This prototype is built as a standalone autonomous microservice.

### 3.1.3 Proof of concept demonstration

The functionality of the SDT engine will be demonstrated on virtual machines containing atleast two different MANO frameworks installed and deployed.

## 3.2 WP2: Service Descriptor Splitter

### 3.2.1 Requirements definition and architecture design

A publish/subscribe message broker model will be once again followed for the SDS engine implementation. The NSD(yaml file) is published to the message broker. A service graph splitter and a NSD splitter consumes the NSD from the broker and splits the NSD into two. The two seperate NSDs are published back to the broker from where other services can consume those through REST API.

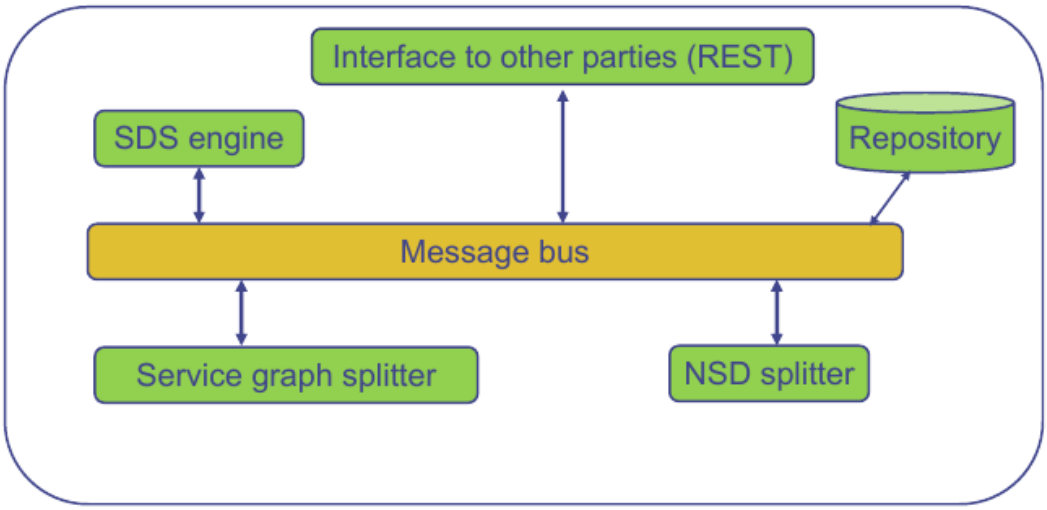


Figure 3.2: Reference architecture for SDS [PDHK]

### 3.2.2 Investigation of service graph partitioning algorithms and libraries

The service graph needs to be split optimally based on the loads and resource allocation on each VNF. To accomplish this a graph partitioning algorithm such as *Mixed Integer Program for Partitioning Graphs* **??** will be used. However the exact choice of algorithm suitable for the project will be decided going forward.

### 3.2.3 Prototype implementation of SDS components

After the concrete architecture is planned, a working prototype of the SDS engine including all the requisite internal components is coded and modeled. This prototype is also built as a standalone autonomous microservice.

### 3.2.4 Proof of concept demonstration

The functionality of the SDS engine will be demonstrated on virtual machines containing atleast two different MANO frameworks installed and deployed.

## 3.3 WP3: MANO Adaptor

### 3.3.1 Requirements definition and architecture design

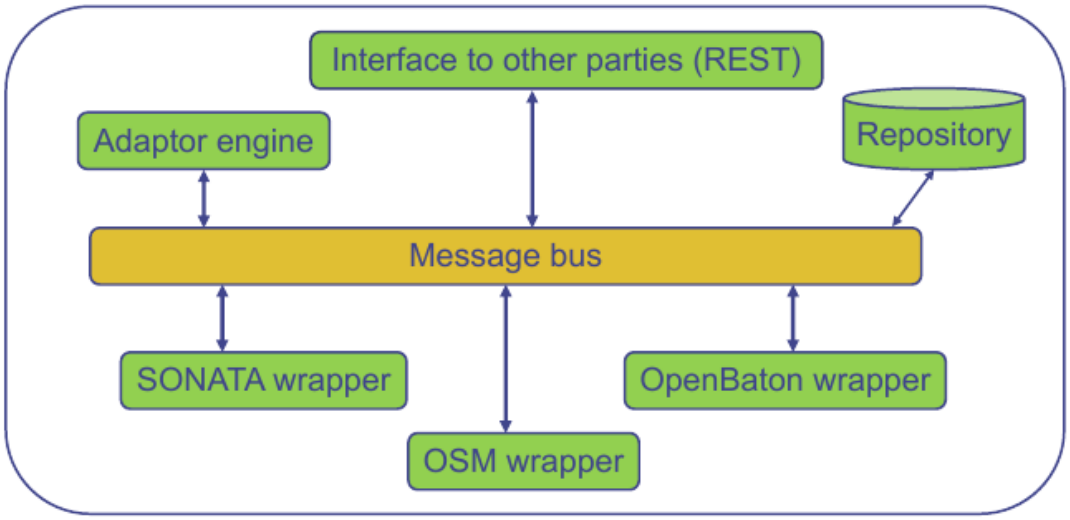


Figure 3.3: Reference architecture for MA [PDHK]

### 3.3.2 Prototype implementation of adaptor components

As a starting point of the prototype, the plan is to choose one of the MANO frameworks listed in the section **??** and implement the wrapper for the interfaces provided by it, with focus on modularity and portability. Thereby, making it easy to add support for interfaces provided by other MANO frameworks. Further, we will add the southbound interfaces required to communicate with the MA.

### 3.3.3 Investigation of MANO scalability challenges

The plan is to investigate MANO scalability challenges by answering research questions such as the ones listed below but not limited to. (Figure 3.4)

- What is the optimal number of MANO instances in a system?
- What is the optimal hierarchical level in a system?

### 3.3.4 Proof of concept demonstration

MA is demonstrated by having two/three virtual machines with MANO frameworks installed, a scenario where a huge number of service requests on one of the MANOs is created. In order to balance the load, the MANO experiencing the service request spike splits the load with another

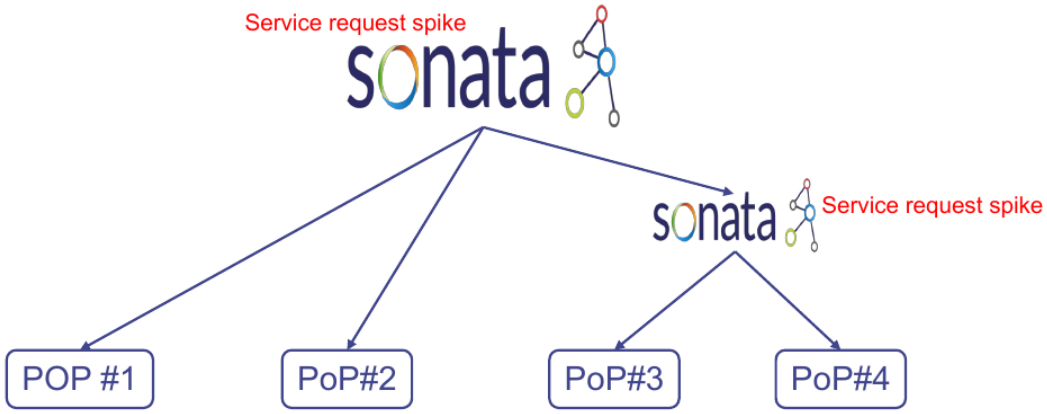


Figure 3.4: MANO scaling scenario [PDHK]

MANO instance. The capability of the MANO to communicate between other MANO instances with the help of MA is thus demonstrated.

# 4

# Non-Functional Requirements

The important non-functional requirements for the work packages are defined here. Below is the overview of some requirements and description of their relevance in this project.

1. ***Scalability:*** One of the main requirements of any cloud-based networking, is its ability to adapt its throughput to varying load. Here, the load of particular network functions, load on the translator and splitter is also considered as an add-on to the overall load of the system. One of the main work packages mentioned in this project is the scalability support for the MANO framework. The adapter plays a major role in scaling up the defined system.

2. ***Reliability:*** The requirement of reliability aims at the ability of the system to provide continuous correct service. It is the continuation of the service in compliance with the service specification. It is one of the weak requirements that have to be met in any case. In this project, this can also be viewed as providing service for the required duration and then terminating.

3. ***Availability:*** This term can be paraphrased as *readiness for correct service*, i.e., delivery of service in compliance with the service specification. Here, the proposed software suite should be available for translation or splitting of the NSDs.

4. ***Flexibility:*** Flexibility abstracts the ability to change the system according to changing requirements. Here in this project, the software suite needs to accommodate the MANO framework and its instances.

5. ***Performance:*** This term summarizes many possible aspects of the system. In this project, the software suite and its components - the translator, splitter, and an adaptor have to be efficient.

# 5 Technologies

## 5.1 MANO Frameworks

In this section, a few MANO frameworks and their NSD schemata are listed(list of a few parameters and their description). Among several frameworks, the plan is to select a few and set them up locally, deploy network services and support them in the project. This list is subjected to future additions or removals.

### 5.1.1 OpenBaton

Open Baton is an open source implementation of ETSI MANO specification. It's main objective is to develop an extensible and customizable framework capable of orchestrating network services from different domains[Ove]. Following table lists the important parameters of NSD[Docb].

| **Parameter** | **Description** |
|---|---|
| Name | The name of the network service |
| Vendor | The vendor or provider of the network service |
| Version | The version of the network service (can be any string) |
| vnfd | The list of VNFs composing the network service (see Virtual Network Function Descriptor for more details) |
| Vld | The list of Virtual Links that are referenced by the VNF Descriptors in order to define network connectivity |
| Vnf_dependency | The list of dependencies between VNFs |

Table 5.1: Open Baton NSD parameters

### 5.1.2 SONATA

SONATA is one of the NSO research projects which targets two crucial technological challenges in the foreseeable future of 5G networks as follows[SON].

- Flexible programmability
- Deployment optimization of software networks for complex services /applications.

SONATA complies with the ETSI NFV-MANO reference architecture. Following tables lists the important parameters of NSD [Docd].

**Network Service Descriptor Section**

| Parameter | Description |
|---|---|
| Vendor | Identifies the network service uniquely across all network service vendors. |
| Name | Name of the network service without its version. |
| Version | Names the version of the NSD. |
| Author | It's an optional parameter. It describes the author of NSD |
| Description | It's an optional parameter, provides an arbitrary description of the network service. |

Table 5.2: SONATA: Network Service Descriptor Section

**Network Functions Section**

| Parameter | Description |
|---|---|
| network_functions | Contains all the VNFs that are handled by the network service. |
| Vnf_id | Represents a unique identifier within the scope of the NSD |
| Vnf_vendor | As part of the primary key, the vendor parameter identifies the VNF Descriptor |
| Vnf_name | As part of the primary key, the name parameter identifies the VNF Descriptor |
| Vnf_version | As part of the primary key, the version parameter identifies the VNF Descriptor |
| Vnf_description | It's an optional parameter, a human-readable description of the VNF |

Table 5.3: SONATA: Network Functions Section

### 5.1.3 TeNOR

Developed by T-NOVA project, it's the NFV Orchestrator platform which is not only responsible for the entire lifecycle service management of NFV but also optimizing the networking and IT resources usage. Network service and VNF descriptors follow the TeNORs data model specifications that are a derived and an extended version of the ETSI NSD and VNF Descriptor data model [dSPR+18]. Following table lists some of the parameters of the descriptors [Doce].

| Parameter | Description |
|---|---|
| Id | Unique ID of the network service |
| Name | Name of the network service |
| Vendor | Identifies the network service uniquely across all network service vendors |
| Version | Names the version of the NSD |
| Manifest_file_md5 | Exposes MD5 check-sums of the meta-data that the manifest file contains |
| vnfds | Array of VNF Descriptors |

Table 5.4: TeNOR: Network Service Descriptor

### 5.1.4 Cloudify

Cloudify is an open source cloud orchestration framework mainly focused on optimization of NFV Orchestration and management. It provides a TOSCA based blueprint which facilitates end-to-end lifecycle of NFV Orchestration. Cloudify follows MANO reference architecture but not entirely compliant to it[dSPR+18]. Following are some high-level sections of the blueprint which describes the network services that are installed and configured [Doca].

| Parameter | Description |
|---|---|
| type | The node-type of this node template |
| properties | The properties of the node template, matching its node type properties schema |
| instances | Instances configuration(deprecated replaced with capabilities and scalable) |
| interfaces | Used for mapping plugins to interface operation, or for specifying inputs for already-mapped node type operations |
| relationships | Used for specifying the relationships that this node template has with other node templates |
| capabilities | Used for specifying the node template capabilities. (Supported since: cloudify_dsl_1_3) Only the scalable capability is supported |

Table 5.5: Cloudify: Node Template

### 5.1.5 Open Source MANO(OSM)

OSM is an open source management and orchestration stack in compliant with ETSI NFV information models. OSM architecture splits between resource orchestrators and service orchestrators [dSPR+18]. Table 5.6 lists the parameters of NSD schema. [Docc]

| Parameter | Description |
|---|---|
| id | Identifier for the NSD |
| name | NSD name |
| Short-name | Short name which appears on the UI |
| vendor | Vendor of the NSD |
| logo | File path for the vendor specific logo. For example, icons/mylogo.png. The logo should be a part of the network service. |
| description | Description of the NSD |
| version | Version of the NSD |

Table 5.6: OSM: Network Service Descriptor

## 5.2 Virtualized Infrastructure Manager (VIM)

VIM is one of the three functional blocks specified in the Network Functions Virtualization Management and Orchestration (NFV-MANO) architecture. VIM is responsible for controlling and managing the NFV Infrastructure (NFVI), by provisioning and optimizing the allocation of physical resources to the virtual resources in the NFVI. Performance and error monitoring is also a key role of the VIM. Popular VIMs are discussed in the following sections.

### 5.2.1 OpenStack

OpenStack[1] is a community-driven open source cloud resource management platform. Compute, storage, and networking resources in a data center can be provisioned using Application Program Interfaces (APIs) or web dashboard provided by OpenStack component, for instance, NOVA is a component which can provide access to compute resources, such as virtual machines and containers. Network management is enabled by NEUTRON component which handles the creation and management of a virtual networking infrastructure like switches and routers. SWIFT component provides a storage system. By making use of many such components, OpenStack can deliver complex services by utilizing an underlying pool of resources.

### 5.2.2 Amazon Web Services (AWS)

AWS[2] is a cloud computing platform, It offers (1) Infrastructure as a Service (IaaS) – provides resources that can be utilized for custom applications (2) Platform as a Service (PaaS) – provides services such as database and email which can be used as individual components for building complex applications (3) Software as a Service (SaaS) – provides user applications with customization and administrative capabilities that are ready to use. AWS is specially preferred by small companies for the flexibility and ease of use of it's cloud infrastructure.

### 5.2.3 Kubernetes

Kubernetes[3] (K8s) is an open-source platform for automation and management of containerized services, it manages computing, networking, and storage infrastructure. Kubernetes was initially developed by Google and now under Cloud Native Computing Foundation. Kubernetes Architecture consists of (1) Master server components – it is the control plane of the cluster and act

[1] https://www.openstack.org/
[2] https://aws.amazon.com/
[3] https://kubernetes.io/

as the gateway for administrators (2) Node Server Components – servers which are performing work by using containers that are called nodes, they communicate with the master component for instructions to run the workload assigned to them. Kubernetes provides comprehensive APIs which are used to communicate between the components and with the external user.

# 6 Related Work

In this chapter, the relevant research efforts that can be used to achieve the goals are discussed. Firstly, the standards and specifications for orchestration and management of NFV in the section are discussed 6.1. The fundamental aspect of a service deployment is the NSD, in the section 6.2 the trends and options of NSDs and research papers that try to mitigate the interoperability challenges between different MANO frameworks are discussed. Section 6.3 will be a brief account of the MANO scalability problem.

As this is the initial project proposal, the state-of-the-art could change progressively and the approach will be updated accordingly.

## 6.1 Standards and Specifications

SDN decouples network control from forwarding with programmable ability. With the decoupling and programmability, SDN brings many benefits such as efficient configuration, improved performance, higher flexibility [XWF+15]. ETSI NFV [NFV2] architecture virtualizes network functions and enables dynamic and flexible selection of service functions. In ETSI NFV architecture, Network Function Forwarding Graph (VNF-FG), which consists of multiple network functions, is defined to describe network service. Internet Engineering Task Force (IETF) Service Function Chaining (SFC) working group also proposes the SFC architecture in RFC 7665 [HP15]. An SFC defines an ordered set of network Service Functions (SFs) for delivery of end-to-end services. Reference [QE16] designs a protocol named Network Service Header (NSH) to decouple the service from topology. An intelligent control plane is proposed to construct service function chains but does not consider the multi-domain situation [B+16].

ETSI NFV designs a basic frame for NFV-MANO. It defines VIM, VNF Manager (VNFM) and NFV Orchestrator (NFVO) for management and orchestration of Network Functions Virtualization Infrastructure (NFVI),VNF and network services [ETS14].

## 6.2 Network Service Description and Interoperability

The description of the network service plays an important role in integration and interoperability of different MANO frameworks. According to ETSI, network service is the “composition of network functions and defined by its functional and behavioral specification.” Following this

approach, a network service can be defined as a set of VNFs and/or Physical Network Functions (PNFs), with virtual links (VLs) interconnecting them and one or more virtualized network function forwarding graphs(VNFFGs) describing the topology of the network service.

Garay et al. [GMUJ] emphasize on NSD, required to allow the different components to inter-operate by comparing the NSD templates by OpenStack (HOT [1]) and OASIS (TOSCA[2]). A strawman model is proposed in the paper to address the upcoming interoperating challenges. The aim is to build a mechanism to translate the NSDs in order to facilitate the interoperability between different MANO frameworks.

## 6.3 Scalability and Hierarchical Orchestration

MANO framework faces significant scalability challenges in large-scale deployments. The amount of infrastructure a single instance of MANO framework can manage is limited. Network service scaling with NFV is discussed in paper [AHOLA+18]. It also shows different procedures that the Network Function Virtualization Orchestrator (NFVO) may trigger to scale a network service according to ETSI specifications and how NFVO might automate them. Abu-Lebdeh et al. [ALNGT] explores the effects of placement of MANO on the system performance, scalability and conclude by suggesting hierarchical orchestration architecture to optimize them. They formally define the scalability problem as an integer linear programming and propose a two-step placement algorithm. A horizontal-based multi-domain orchestration framework for Md-SFC(Multi-domain Service Function Chain) in SDN/NFV-enabled satellite and terrestrial networks is proposed in [LZF+]. The authors here address the hierarchical challenges with a distributed approach to calculate the shortest end-to-end inter-domain path.

The main intention is to answer the MANO scalability challenges, by exploring the optimal number of MANO deployments in a system and optimal hierarchical level. Also, how to manage the state of such a system dynamically.

[1] Heat orchestration template (HOT) specification:
http://docs.openstack.org/heat/rocky/template_guide/hot_spec.html

[2] Topology and Orchestration Specification for Cloud Applications (2013):
http://docs:oasis-open:org/tosca/TOSCA/v1:0/os/TOSCA-v1.0-os.html

# 7 Project Time Plan

In this section, the time plan for the project is discussed. This provides an overview of different tasks and what the team achieves for the course of one year. The intention of this timeline is to provide a rough idea regarding the tasks and their duration. The actual plan will be decided and defined as the project proceeds further. For better understanding, the main tasks of the project are divided as shown in the table below.

| Sr.No. | Tasks | Start | End |
|---|---|---|---|
| 1 | Project organization | 11.10.2018 | 10.11.2018 |
| 2 | Preparation and Presentation of mini seminar | 11.10.2018 | 19.11.2018 |
| 3 | Developing Project Plan | 20.11.2018 | 06.12.2018 |
| 4 | Reviewing Technologies | 20.11.2018 | 20.12.2018 |
| 5 | Designing Architecture | 06.12.2018 | 31.01.2019 |
| 6 | Implementation and Deployment | 07.02.2019 | 30.08.2019 |
| 7 | Presentation | 01.09.2019 | 20.09.2019 |

Table 7.1: List of all tasks in the time plan

The following list further defines the goals of each task:

1. **Project organization:** Establish communication between team members and understand each other, in terms of their skill-set and expertise. Also, decide upon tools for project management, task management, version control and a platform for all further communications. As an outcome, Github is used for issue tracking, task management and version control, Texstudio for documentation and Slack for internal communication.

2. **Preparation and Presentation of mini-seminar:** To get an overview of various technologies and subjects that are required for the project. Select one of the subjects, research the subject in depth and present the concepts that are relevant to the project, to the team.

3. **Developing project plan:** Consolidate all the seminar topics presented by each team member and make a basic sketch of what to achieve, how to achieve and by when to achieve. For the project group, the document is a valuable reference to get an overview of the problem, related technologies, and required sub-tasks.

4. **Reviewing technologies:** To list all the technologies that are relevant to the project,

review the pros and cons of each technology and to decide upon technologies to be used in the project.

5. **Designing architecture:** The aim is to discuss, design and document the base for implementation. The project group has been divided into sub-groups and the sub-groups will decide on technologies and methods/approaches for implementation. The sub-groups are as follows.

| WP1 | WP2 | WP3 |
|---|---|---|
| Arkajit | Sanket | Ashwin |
| Vivek | Harshitha | Bhargavi |
| Suheel | | Deeksha |

Table 7.2: Details of sub-groups

This phase is one of the most crucial phases in the project group, as it is a core foundation to the project.

6. **Implementation and Deployment:** Each sub-group works independently on the respective work packages and shares the updates with the team on weekly basis. Integration and testing the end product by benchmarking and defining various test cases to be able to deliver a stable product by the end of the implementation phase. This requires maximum efforts and time, as various versions/prototypes will be developed until a satisfying and stable end product is obtained. Implementation can be done in the following stages.

   - **Initial Development**: Sub-groups working independently on respective work packages.
   - **Unit Test**: Conduct testing on each of the packages developed.
   - **Integration**: Integrate the work of all the sub-groups.
   - **Final Testing**: Use bench-marking and test the final product.

7. **Presentation:** After implementing the software suite, it is presented to the supervisors. This is achieved by presenting the results obtained in the form of a report or pictorial representation. Also, the team presents the challenges, insights, and benefits inferred from the end product that is built.

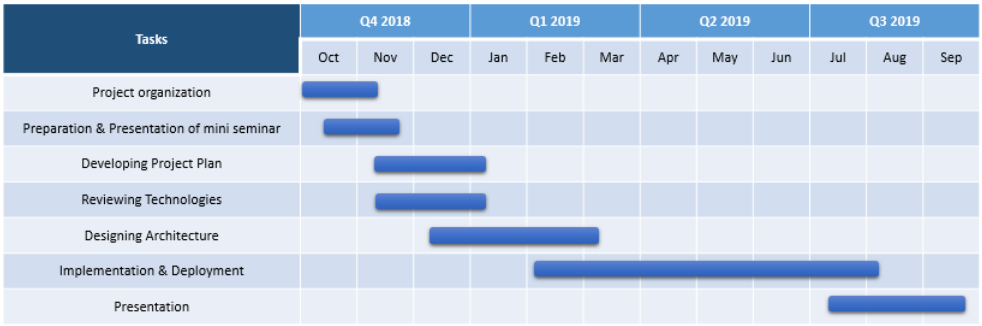


Figure 7.1: A gantt diagram visualizing the time plan.

The sequence of tasks is visualized in the above diagram. Though the timeline mentioned is fixed and can change during the course of various phases of the project, this is to constantly remind the team about the deadlines and to foresee further responsibilities and milestones to be achieved.

# 8 Conclusion

As discussed previously, there are many challenges to overcome and milestones to be achieved. In the end, the goal is to build an end product incorporating a MANO adaptor, NSD Splitter, and NSD translator. The next step is to use this document as a base and start reviewing all the technologies and decide upon which ones to incorporate. Design and documentation of the architecture which is the foundation to implement the project will begin. The sub-groups will start their work independently on the intended modules, Every week the progress of each sub-group is reviewed to share knowledge and findings with the rest of the team. Ultimately, we collaborate the results of the sub-groups to create a stable end product.

# Bibliography


[AHOLA+18] Oscar Adamuz-Hinojosa, Jose Ordonez-Lucena, Pablo Ameigeiras, Juan J Ramos-Munoz, Diego Lopez, and Jesus Folgueira. Automated network service scaling in nfv: Concepts, mechanisms and scaling workflow. *arXiv preprint arXiv:1804.09089*, 2018.

[ALNGT] Mohammad Abu-Lebdeh, Diala Naboulsi, Roch Glitho, and Constant Wette Tchouati. NFV orchestrator placement for geo-distributed systems.

[B+16] Mohamed Boucadair et al. Service function chaining (sfc) control plane components & requirements. *draft-ietf-sfc-control-plane-06 (work in progress)*, 2016.

[DKP+17] Sevil Dräxler, Holger Karl, Manuel Peuster, Hadi Razzaghi Kouchaksaraei, Michael Bredel, Johannes Lessmann, Thomas Soenen, Wouter Tavernier, Sharon Mendel-Brin, and George Xilouris. Sonata: Service programming and orchestration for virtualized software networks. In *Communications Workshops (ICC Workshops), 2017 IEEE International Conference on*, pages 973–978. IEEE, 2017.

[Doca] Cloudify Documentation. *Cloudify Schema Documentation.*

[Docb] Open Baton Documentation. *Open Baton Schema Documentation.*

[Docc] OSM Documentation. *OSM Schema Documentation.*

[Docd] SONATA Documentation. *SONATA Schema Documentation.*

[Doce] TeNor Documentation. *TeNor Schema Documentation.*

[dSPR+18] Nathan F Saraiva de Sousa, Danny A Lachos Perez, Raphael V Rosa, Mateus AS Santos, and Christian Esteve Rothenberg. Network service orchestration: A survey. *arXiv preprint arXiv:1803.06596*, 2018.

[Ers13] Mehmet Ersue. Etsi nfv management and orchestration-an overview. In *Proc. of 88th IETF meeting*, 2013.

[ETS14] NFVISG ETSI. Gs nfv-man 001 v1. 1.1 network function virtualisation (nfv); management and orchestration, 2014.

[GMUJ] Jokin Garay, Jon Matias, Juanjo Unzilla, and Eduardo Jacob. Service description in the NFV revolution: Trends, challenges and a way forward. 54(3):68–74.

[HP15] Joel Halpern and Carlos Pignataro. Service function chaining (sfc) architecture. Technical report, 2015.

[LZF+] Guanglei Li, Huachun Zhou, Bohao Feng, Guanwen Li, and Qi Xu. Horizontal-based orchestration for multi-domain SFC in SDN/NFV-enabled satellite/terrestrial networks. 15(5):77–91.

[NFV2] GS NFV. Network functions virtualisation (nfv); architectural framework. *NFV ISG*, 2.

[Ove] Open Baton Overview. *Open Baton Overview*.

[PDHK] Sevil Dräxler Prof. Dr. Holger Karl, Hadi Razzaghi Kouchaksaraei. Work packages descriptions.

[QE16] Paul Quinn and Uri Elzur. Network service header. *Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-nsh-10*, 2016.

[SON] About SONATA. *SONATA*.

[XWF+15] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51, 2015.