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:

Sevil Dräxler Hadi Razzaghi

Prof. Dr. Holger Karl

Paderborn, December 4, 2018

Contents

List of Figures

List of Tables

1 Motivation

The rapid growth of mobile data services driven by mobile internet has led to a substantial challenge 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) has 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. NFVs require 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 implementation of a MANO framework in the market.

End-to-end network service delivery with high availability, scalability and low latency requires chaining of the Network Functions 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 Virtual Network Functions (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, forwarding graph of VNFs. Each of the VNFs contain its own descriptors (VNFD) and 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.

First our project aims at tackling this 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 to allow for the interaction between different MANO frameworks, expose the network service instantiation interfaces of the underlying MANO frameworks and retrieve 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 provision

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, we discuss the intended goals of the project and list some use cases.

2.1 Goals

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

2.1.1 Service Descriptor Translator (SDT)

In this work package we implement a translator engine for a Network Service Descriptor (NSD). The Network Function Virtualization Orchestrator (NFVO) is the main orchestration unit of a MANO framework that manages the lifecycle of a network service (NS). One of the main functions of a NFVO is registering a network service in the NS catalog by onboarding the NSD. In a scenario where a network service needs to be deployed and registered among two different MANO frameworks having their own NSD schema, our translator engine would help translate the NSD schema from one MANO framework to the other. We will have mechanisms to extract the NSD schema from the NS catalog of the two different MANO frameworks and pass it to SDT engine which will consume the NSD schemas and translate one NSD schema to the other MANO framework's NSD schema. REST interfaces will be made available for other services to interface with the SDT.

2.1.2 Service Descriptor Splitter (SDS)

In production environment, a Network Service will be deployed over a vast geographical region and will span 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 we implement a Service Descriptor Splitter (SDS) which will be splitting the NSD of NS. SDS will be taking NSD as an input which will have all the information elements which can be extracted to generate separate NSDs. In our approach, the Service Graph is extracted from the input NSD and is split into subgraphs which will result in a separate set of elements such as Virtual Network Functions, Virtual Links, Virtual Network Function Forwarding Graphs etc. We will

be evaluating different network graph splitting algorithms to split the Service graph. We will take these sets and generate NSDs according to the domain specific orchestrators.

Figure 2.1: This figure visualizes the structure of SDS work package.

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 4.1.2 MANO framework has adaptors for OpenStack **??** and Kubernetes**??**. 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, we are planning to investigate MANO scalability challenges by answering research questions such as the ones listed below but not limited to.

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

2.1.4 Integration Of Work Packages

In this work package we integrate the translator engine, splitter engine and the MANO adaptor. This integration will augment the funtionality of the MANO adaptor in terms of addressing the scalability challenges between different MANO frameworks. It also extends the capability of both the splitter engine and translator engine 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.

The project group will be initially divided into 3 subgroups. Each subgroup works on a single work package. However, update on work in progress and knowledge transfer will continuously take place between the subgroups. In the final development phase, the entire team will work towards integrating the individual modules of the work packages into a single software suite.

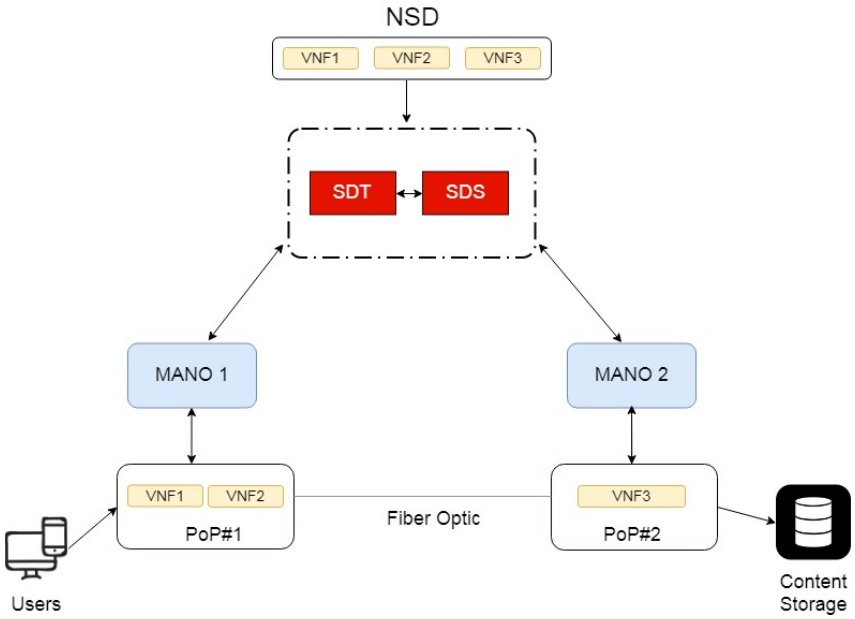

Figure 2.2: This figure visualizes the structure of first and second modules.

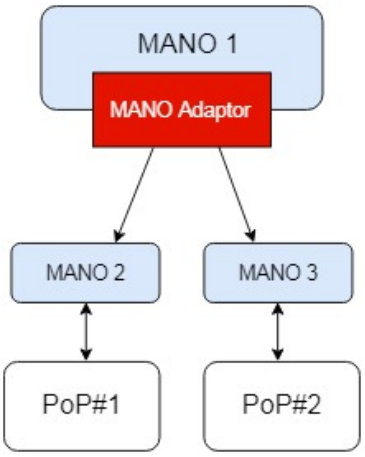

Figure 2.3: This figure visualizes the structure of third module.

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 [**?**] and another operator uses OSM framework [**?**]. These frameworks have different NSD schemata(refer 4.1.2 and 4.1.5). NSD schemata contains 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 to 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 interoperability 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.

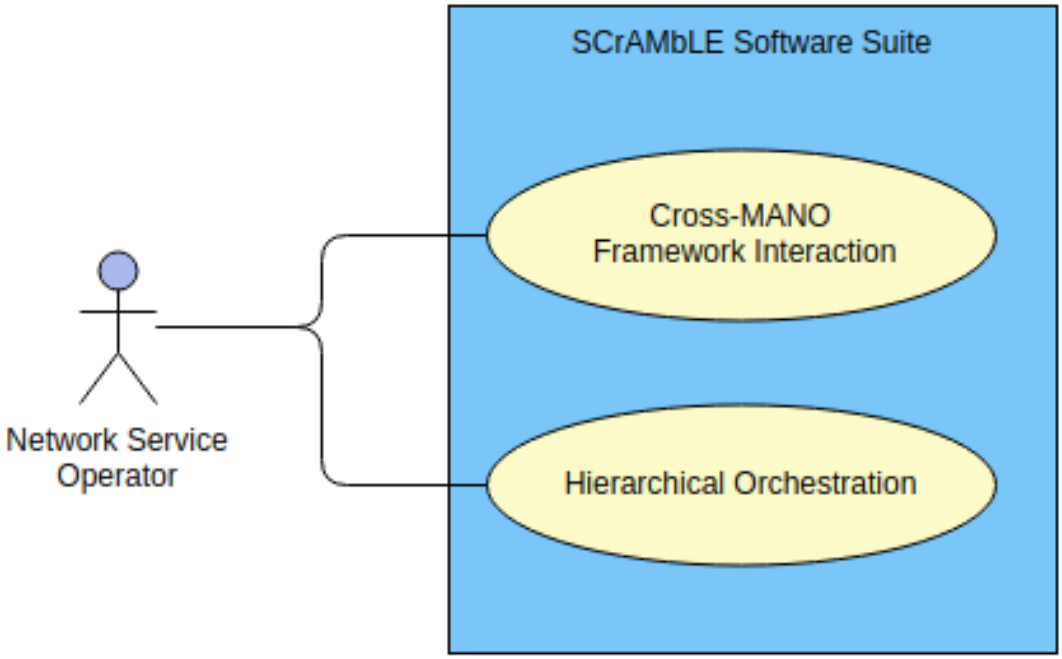

Figure 2.4: Use Case Diagram

3

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, 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 has to be met at 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 work packages should be available for translation or splitting of the NSDs.

4. ***Flexibility:*** This term defines the ability to accommodate the changing requirements. Here in this project, the system needs to accommodate the MANO framework and its instances.

5. ***Performance:*** This term summarizes the possible aspects of the system. In this project, throughput and latency of all the network functions involved is considered as a measurement of efficiency.

4 Technologies

4.1 MANO Frameworks

In this section, we have listed a few MANO frameworks and their NSD schemata (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.

4.1.1 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[**?**]. Following table lists the important parameters of NSD[**?**].

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 4.1: Open Baton NSD parameters

4.1.2 SONATA

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

- 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 [**?**].

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 4.2: 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 4.3: SONATA: Network Functions Section

4.1.3 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 [**?**]. Following table lists some of the parameters of the descriptors [**?**].

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 4.4: TeNOR: Network Service Descriptor

4.1.4 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[**?**]. Following are some high-level sections of the blueprint which describes the network services that are installed and configured [**?**].

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 4.5: Cloudify: Node Template

4.1.5 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 [**?**]. Table 4.6 lists the parameters of NSD schema. [**?**]

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 4.6: OSM: Network Service Descriptor

4.2 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.

4.2.1 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.

4.2.2 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.

4.2.3 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

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

and act 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.

5 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 5.1. The fundamental aspect of a service deployment is the NSD, in the section 5.2 the trends and options of NSDs and research papers that try to mitigate the interoperability challanges between different MANO frameworks are discussed. Section 5.3 will be a brief account on 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.

5.1 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 [**?**]. ETSI NFV [**?**] 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 [**?**]. A SFC defines an ordered set of network Service Functions (SFs) for delivery of end-to-end services. Reference [**?**] 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 [**?**].

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 [**?**].

5.2 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 netwrok 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. [**?**] emphasize on NSD, required to allow the different components to interoperate 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. We aim at building a mechanism to translate the NSDs in order to facilitate the interoperability between different MANO frameworks.

5.3 Scalability and Hierarchical Orchestration

MANO framework face 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 [**?**]. 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. [**?**] 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 [**?**]. The authors here address the hierarchical challenges with a distributed approach to calculate 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

6 Project Time Plan

In this section, the time plan for the project is discussed, which 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 over 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 6.1: List of all tasks in the time plan

The following list further defines the goals of each task:

1. **Project organization:** To establish communication between team members to understand each other, in terms of their skills set and expertise. Also, decide upon tools for project management, task management, version control and a platform for all further communications. As an outcome, we have decided to use Gitlab for project, task management and for 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 is 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:** To combine all the subjects presented by each team member and make a basic sketch of what to achieve, how to achieve and by when to achieve it. 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 pros and cons of each technology and to decide upon technologies to be used in the project.

5. **Designing architecture:** The aim here is to discuss, design and document the base for implementation. We have divided the project group into sub-groups who will decide on technologies and methods for implementation. The divided sub-groups is as follows.

WP1	WP2	WP3
Arkajit	Sanket	Ashwin
Vivek	Harshitha	Bhargavi
Suheel		Deeksha

Table 6.2: 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 working independently on their respective work packages and discuss the updates with team on weekly basis. Integrate and test the end product by benchmarking and by defining various scenarios where the product can be tested and to produce 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 following stages.

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

7. **Presentation:** After implementing the system, it is presented to the supervisors. This is achieved by presenting in document or pictorially the end results obtained by the produced developed. Present the learnings, challenges and benefits from the end product produced.

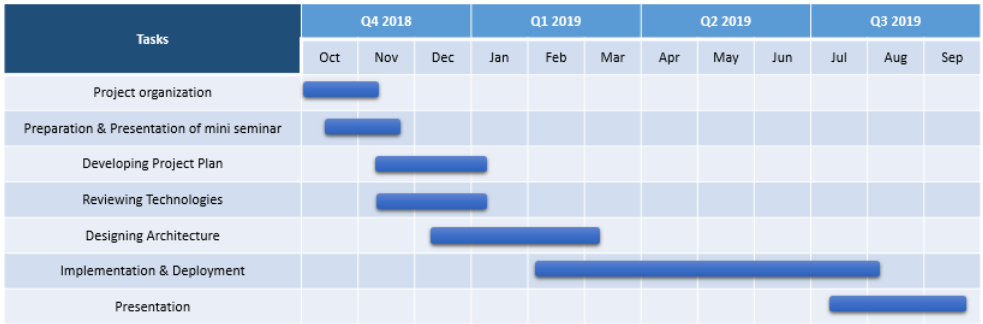

Figure 6.1: A gantt diagram visualizing the time plan.

The sequence of the 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 deadlines and to foresee further responsibilities and milestones to be achieved.

7 Conclusion

As discussed previously, there are many challenges to overcome and milestones to be achieved. At the end, the goal is to build an end product, i.e. OSM adaptor, NSD Splitter and NSD translator but an in-depth knowledge on how to achieve it is needed. The next step is to use this document as a base and start reviewing all the technologies and decide upon which ones to use. Design and documention of the architecture which is the foundation to implement the project will begin. Also in order to be efficient, The project group is divided into sub-groups, to work independently on various topics. Every week the progress of each sub-group is reviewed toshare knowledge and findings to rest of the team. Ultimately, collaborate all the results of the sub-groups to produce a stable end product.

Bibliography