Optimal Virtual Network Function Placement Algorithm in Network Functions Virtualization Infrastructure

Submitted by:

Bhavesh Nathwani (bpn232)

Madhav Prabhu (mcp573)

Contents

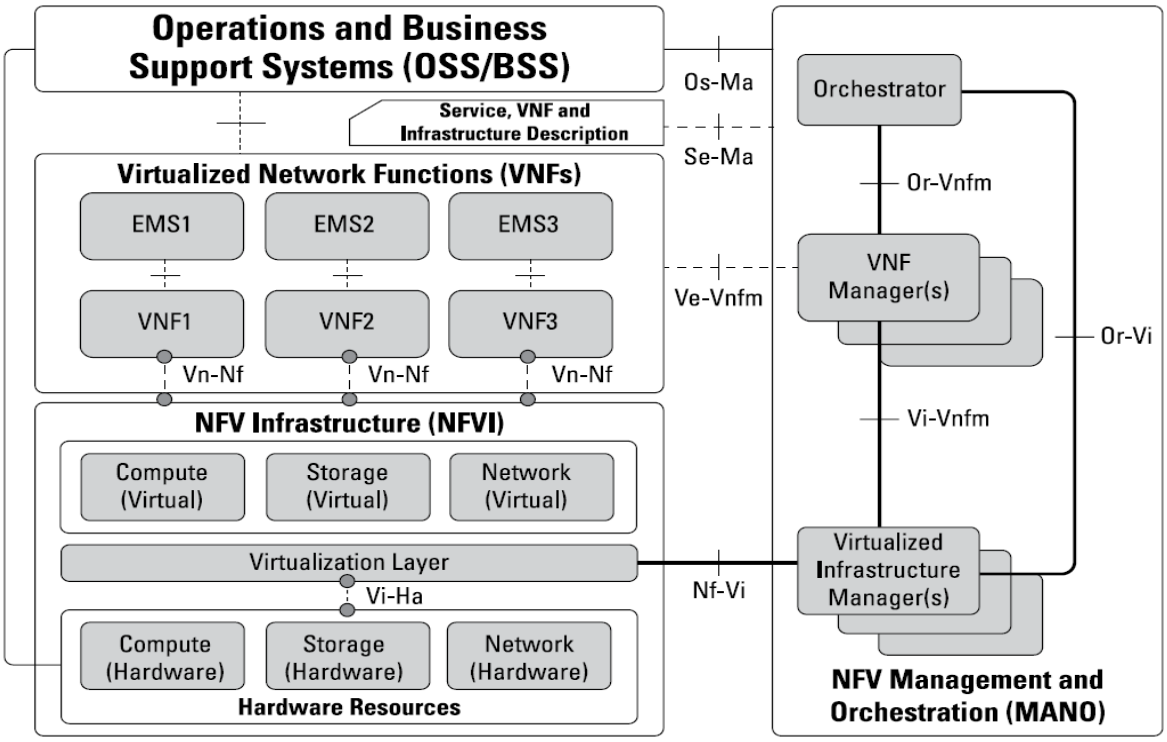
Chapter 1

Introduction:

Network Functions Virtualization (NFV) is a network architecture concept that implements traditional network functions and associated applications using virtualization technologies effectively improving cost and reduction deployment time as compared to traditional methods.

In a traditional network, network functions such as Firewalls, Proxies, Load Balancers etc. are deployed in the form of hardware middleboxes. A middlebox is any traffic processing device except for routers and switches. The increase in size of a network increases the number of physical middleboxes, which in turn increases the procurement and management of the same. These appliances are critical for the security, performance and compliance of the network but prove to be expensive, complex and difficult to manage. NFV turns these hardware middleboxes into software-based virtualized entities.

NFV Architecture:



OSS: Operations Support Systems

BSS: Business Support Systems

EMS: Element Management System

Vn-Nf: VNF - NFV Infrastructure

Vi-Ha: Virtualization Layer - Hardware Resources

Os-Ma: OSS/BSS - NFV Management and Orchestration

Se-Ma: Service, VNF and Infrastructure Description - NFV Management and Orchestration

Ve-Vnfm: VNF/EMS - VNF Manager

Nf-Vi: NFVI - Virtualized Infrastructure Manager

Or-Vnfm: Orchestrator - VNF Manager

Vi-Vnfm: Virtualized Infrastructure Manager

Or-Vi: Orchestrator - Virtualized Infrastructure Manager

The diagram shows the ETSI NFV high-level architectural framework. The various components along with horizontal and vertical interfaces are shown. The architecture consists of 3 major components which are described in detail as follows:

1. NFV Infrastructure (NFVI) space:

   This subsystem constitutes all hardware components namely servers, storage and networking along with software components on which Virtual Network Functions (VNFs) are deployed. The subsystems include compute, storage, networking resources and hypervisor. The hardware resources are pooled together and are then logically partitioned to form independent virtual resources.

2. NFV Management and Orchestration (MANO) space:

   This subsystem includes Network Function Virtualization Orchestrator (NFVO), the Virtualized Infrastructure Manager (VIM) and the Virtual Network Functions Manager (VNFM). The management entities provide orchestration and lifecycle management of software resources of NFVI and VNFs.

3. Virtual Network Functions (VNFs) space**:**

   This subsystem comprises of software implementations of hardware middleboxes that can be instantiated as one or more virtual machines (VMs) on the NFVI. The VNFS are software-only components that run on virtual assets created by the virtualization layer from a set of physical hardware resources.

Chapter 2

The Algorithm:

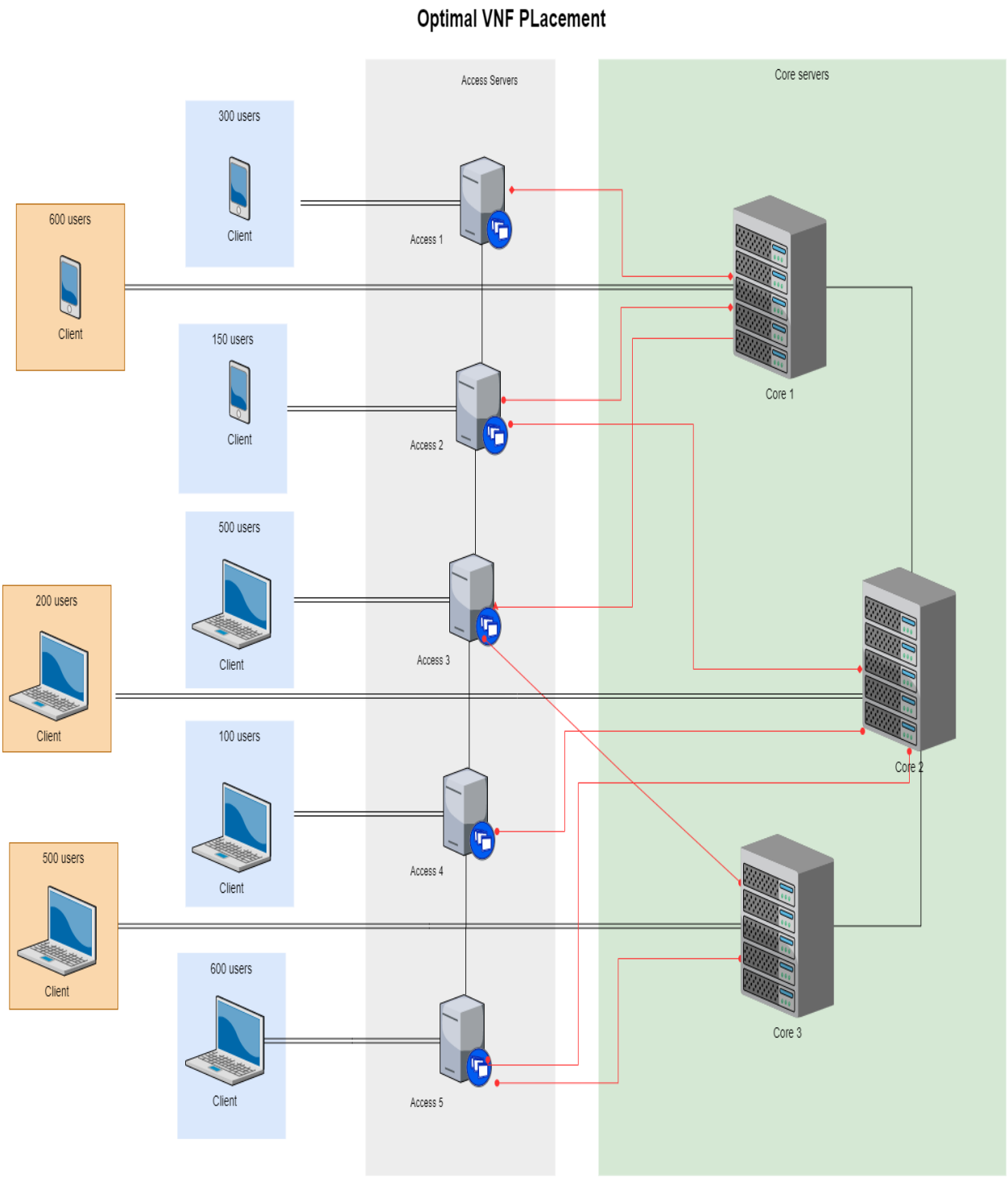
Background:

The NFV MANO Entity consists of the Orchestrator, VNFM and VIM. The VIM manages the hardware resources of the NFVI. Responsibilities of VIM include resource and operations management, detecting and analyzing performance issues, collecting fault, performance and capacity data. The VNFM provides life cycle management support for VNFs. The Orchestrator is responsible for onboarding, instantiating, scaling, updating and managing a network service. A network service can span over multiple VNFs. The orchestrator also communications with authentication and authorizations entities such as Operations Support System (OSS) and Business Support System (BSS).

Model Design:

The objective of this algorithm is to deploy VNFs on nodes based on the user requirement and capacity constraints. If the capacity of a node is depleted, VNFs should be deployed on adjacent nodes taking into account spare capacities as well. The OVNF model runs in the VNFM sub-system of the NFV MANO entity. The algorithm is based on Topological Design model discussed in the course. As we know, the VNFM sub-system is responsible for deployment of VNFs, the algorithm uses capacity data obtained by VIM sub-system and user requirement data from the Orchestrator. The first stage of the algorithm involves deployment of VNFs based on requirement of each node, i.e. the number of users connected to that node and the capacity of that node. The second stage of the model involves deployment of VNFs based on adjacency of nodes and spare capacity of each adjacent node.

Topology:


Optimal VNF PLacement
Access Servers
Core servers
300 users
Client
600 users
Client
Access 1
Core 1
150 users
Client
Access 2
500 users
Client
200 users
Client
Access 3
Core 2
100 users
Client
Access 4
500 users
Client
Core 3
600 users
Client
Access 5

We consider a two-level network containing Core and Access Nodes. We analyze the algorithm based on two scenarios. In the first scenario, VNFs are allowed to be deployed on both Core and Access nodes. In the second scenario, VNFs are allowed to be deployed only on Access nodes. In each scenario, the 2-stage process described in the Model Design section above is followed.

We assign different cost of deployment for VNFs on both core and access nodes. In a real-world scenario, it is easier to instantiate an access node as compared to a core node based on user demands. Deployment of core nodes requires long term commitment and considerable budget. That is why, cost of deployment of VNFs is higher in core nodes than in access nodes. Our model maximizes deployment of VNFs but only does so in the limits of Capacity and requirement, therefore no extra VNFs are deployed which in turn optimizes our deployment cost too.

Scenario 1**:**

Model 1:

Indices:

i =	1..I	:Core Nodes
j =	1..J	:Access Nodes
k1 =	1..K1	:No of VNFs for step 1
k2 =	1..K2	:No of VNFs for step 2
Links =	1..L	:Links between nodes

Constants:

C_{ci}	:Capacity of i^{th} Core node
C_{aj}	:Capacity of j^{th} Access node
f	:Size of VNF
L_{ci}	:No of Users connected to i^{th} Core node
L_{aj}	:No of VNFs connected to j^{th} Access node
U	:No of users a single VNF can handle
δ_{cii}	: =1 if i^{th} Core node is connected to i^{th} Core node i != i, otherwise 0
δ_{ajj}	: =1 if j^{th} Access node is connected to j^{th} Access node j != j, otherwise 0
e_1, e_2, e_3, e_4	: costs associated with deploying VNFs in both stages

Variables:

x_{ik}	: =1 if k^{th} VNF is deployed on i^{th} Core node, otherwise 0
r_{jk}	: =1 if k^{th} VNF is deployed on j^{th} Access node, otherwise 0
SC_{ci}	: capacity left after initial deployment on i^{th} Core node
SC_{aj}	: capacity left after deployment on j^{th} Access node
SU_{ci}	: no of users left after initial deployment on i^{th} Core node
SU_{aj}	: no of users left after deployment on j^{th} Access node
$P1_{ij}$	: variable showing spare capacities of adjacent nodes of i^{th} Core node based on links in adjacency matrix
$P2_{ij}$	: variable showing spare capacities of adjacent nodes of i^{th} Core node based on links in adjacency matrix
w1	: sum of all unserved users in Core Nodes
w2	: sum of all unserved users in Access Nodes

Objective function:

maximize Deployment1: $\sum_{k=1}^{K1} \sum_{i=1}^{I} x_{ik} + \sum_{k=1}^{K1} \sum_{j=1}^{J} r_{jk}$

Constraints:

$\sum_{k=1}^{K1}(\sum_{i=1}^{I} x_{ik} + \sum_{j=1}^{J} r_{jk}) \leq 1$	: VNF Deployment constraint for step 1
$\sum_{k=1}^{K1} x_{ik} * f \leq C_{ci}$	: i = 1..I, Core capacity constraint
$\sum_{k=1}^{K1} r_{jk} * f \leq C_{aj}$	: j = 1..J, Access capacity constraint
$\sum_{k=1}^{K1} x_{ik} * U \leq L_{ci}$	: i = 1..I, Core user constraint
$\sum_{k=1}^{K1} r_{jk} * U \leq L_{aj}$	: j = 1..J, Access user constraint
$SC_{ci} = C_{ci} - \sum_{k=1}^{K1} x_{ik} * f$	: i = 1..I, Spare Core Capacity calculation
$SC_{aj} = C_{aj} - \sum_{k=1}^{K1} r_{jk} * f$	: j = 1..J, Spare Access Capacity calculation
$SU_{ci} = L_{ci} - \sum_{k=1}^{K1} x_{ik} * U$	: i = 1..I, Spare Core users calculation
$SU_{aj} = L_{aj} - \sum_{k=1}^{K1} r_{jk} * U$	: j = 1..J, Spare Access users calculation

w1 = $\sum_{i=1}^{I} SU_{ci}$: Total spare core users calculation

w2 = $\sum_{j=1}^{J} SU_{aj}$: Total spare access users calculation

$P1_{ij} = \delta_{cii} * SC_{ci}$: Adjacent core capacity calculation

$P2_{ij} = \delta_{ajj} * SC_{aj}$: Adjacent access capacity calculation

The first model deploys VNFs based on user requirement and capacity constraints of core and access nodes. In this same step, the number of users that remain to be serviced and the spare capacity of all nodes are calculated.

Model 2:

Variables:

t_{ik} : =1 if k^{th} VNF is deployed on i^{th} Core node, otherwise 0

d_{jk} : =1 if k^{th} VNF is deployed on j^{th} Access node, otherwise 0

Objective function:

maximize Deployment1: $\sum_{k=1}^{K1} \sum_{i=1}^{I} t_{ik} + \sum_{k=1}^{K1} \sum_{j=1}^{J} d_{jk}$

Constraints:

$\sum_{k=1}^{K1} (\sum_{i=1}^{I} t_{ik} + \sum_{j=1}^{J} d_{jk}) \leq 1$: VNF Deployment constraint for step 2

$\sum_{k=1}^{K1} t_{ik} * f \leq P1_{ij}$: i = 1..I, Core capacity constraint based on adjacency

$\sum_{k=1}^{K1} d_{jk} * f \leq P2_{ij}$:J=1..J, Access capacity constraint based adjacency

$\sum_{k=1}^{K1} t_{ik} * U \leq w1$: i = 1..I, Core user constraint based on spare users

$\sum_{k=1}^{K1} d_{jk} * U \leq w2$: j = 1..J, Access user constraint based on spare users

The second model takes our findings from the first model and deploys VNFs based on additional adjacency constraint. Each node takes into consideration whether there is spare capacity available in adjacent nodes and if there are users left to be serviced on that particular node, VNFs are deployed on adjacent nodes to cater it's needs.

Scenario 2:

Model 1:

Indices:

i = 1..I :Core Nodes

j = 1..J :Access Nodes

k1 = 1..K1 :No of VNFs for step 1

k2 = 1..K2 :No of VNFs for step 2

Links = 1..L :Links between nodes

Constants:

C_{ci} :Capacity of i^{th} Core node

C_{aj} :Capacity of j^{th} Access node

f :Size of VNF

L_{ci} :No of Users connected to i^{th} Core node

L_{aj} :No of VNFs connected to j^{th} Access node

U :No of users a single VNF can handle

δ_{ajj} : =1 if j^{th} Access node is connected to j^{th} Access node j != j, otherwise 0

e_1, e_2 : costs associated with deploying VNFs in both stages

Variables:

r_{jk}	: =1 if k^{th} VNF is deployed on j^{th} Access node, otherwise 0
m_{jk}	: =1 if k^{th} VNF is deployed on j^{th} Access node, otherwise 0
SC_{aj}	: capacity left after deployment on j^{th} Access node
SU_{ci}	: no of users left on i^{th} Core node
SU_{aj}	: no of users left after deployment on j^{th} Access node
$P2_{ij}$	: variable showing spare capacities of adjacent nodes of i^{th} Core node based on links in adjacency matrix
w1	: sum of all unserved users in Access Nodes
w2	: sum of all unserved users in Core Nodes

Objective function:

maximize Deployment1: $\sum_{k=1}^{K1} \sum_{j=1}^{J} r_{jk}$

Constraints:

$\sum_{k=1}^{K1} \sum_{j=1}^{J} r_{jk} \leq 1$	: VNF Deployment constraint for step 1
$\sum_{k=1}^{K1} r_{jk} * f \leq C_{aj}$	: j = 1..J, Access capacity constraint
$\sum_{k=1}^{K1} r_{jk} * U \leq L_{aj}$	: j = 1..J, Access user constraint
$SC_{aj} = C_{aj} - \sum_{k=1}^{K1} r_{jk} * f$	: j = 1..J, Spare Access Capacity calculation
$SU_{aj} = L_{aj} - \sum_{k=1}^{K1} r_{jk} * U$	: j = 1..J, Spare Access users calculation
$w1 = \sum_{j=1}^{J} SU_{aj}$	: Total spare access users calculation
$w2 = \sum_{i=1}^{I} L_{ci}$	: Total core users calculation
$P2_{ij} = \delta_{ajj} * SC_{aj}$	: Adjacent access capacity calculation

In this scenario, we take the same topology and data as considered in scenario 1. The only difference is that VNFs are deployed only on Access Nodes and not on Core nodes. The constraints take into consideration only the capacity and spare capacity of Access nodes in both stages. In the first stage, VNFs are deployed and spare capacity plus spare users are calculated on the Access nodes. The spare users from Core Nodes are equal to the number of users connected to the Core nodes.

Model 2:

Variables:

m_{jk} : =1 if k^{th} VNF is deployed on j^{th} Access node, otherwise 0

Objective function:

maximize Deployment1: $\sum_{k=1}^{K1} \sum_{j=1}^{J} m_{jk}$

Constraints:

$\sum_{k=1}^{K1} \sum_{j=1}^{J} m_{jk} \leq 1$: VNF Deployment constraint for step 2

$\sum_{k=1}^{K1} m_{jk} * f \leq P2_{ij}$:j=1..J, Access capacity constraint based adjacency

$\sum_{k=1}^{K1} m_{jk} * U \leq w1 + w2$:j = 1..J, Access user constraint based on spare users

The constraints of the second model of this scenario take into consideration the spare capacity from the access nodes and the spare users to both core and access nodes. VNFs are only deployed on Access nodes that are connected to the Core Nodes.

Results:

For Scenario 1:

x [*,*] (tr)

:	1	2	3	:=
1	1	0	0	
2	1	0	0	
3	1	0	0	
4	0	1	0	
5	0	1	0	
6	0	0	1	
7	0	0	1	
8	0	0	1	
9	0	0	1	
10	0	0	1	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	
15	0	0	0	
16	0	0	0	
17	0	0	0	
18	0	0	0	
19	0	0	0	
20	0	0	0	

Initial Deployment on Core Nodes

r [*,*] (tr)

:	1	2	3	4	5	:=
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
5	0	0	0	0	0	
6	0	0	0	0	0	
7	0	0	0	0	0	
8	0	0	0	0	0	
9	0	0	0	0	0	
10	0	0	0	0	0	
11	1	0	0	0	0	
12	1	0	0	0	0	
13	1	0	0	0	0	
14	0	1	0	0	0	
15	0	0	0	1	0	
16	0	0	0	0	1	
17	0	0	0	0	1	
18	0	0	0	0	1	
19	0	0	0	0	1	
20	0	0	0	0	1	

Initial Deployment on Access nodes

t [*,*] (tr)

:	1	2	3	:=
1	0	1	0	
2	0	1	0	
3	0	0	1	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	

;

Deployment on Core Nodes based on previous step

d [*,*] (tr)

:	1	2	3	4	5	:=
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	1	0	0	0	0	
5	1	0	0	0	0	
6	1	0	0	0	0	
7	0	1	0	0	0	
8	0	1	0	0	0	
9	0	0	0	1	0	
10	0	0	0	1	0	

;

Deployment on Access Nodes based on previous step

For Scenario 2:

r [*,*] (tr)

:	1	2	3	4	5
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	0	1	0	0	0
5	0	0	0	1	0
6	0	0	0	0	1
7	0	0	0	0	1
8	0	0	0	0	1
9	0	0	0	0	1
10	0	0	0	0	1
11	0	0	0	0	1
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	0
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	0	0
20	0	0	0	0	0

Initial Deployment on Access nodes

m [*,*] (tr)

:	1	2	3	4	5	:=
1	1	0	0	0	0	
2	1	0	0	0	0	
3	1	0	0	0	0	
4	0	1	0	0	0	
5	0	1	0	0	0	
6	0	0	0	1	0	
7	0	0	0	1	0	
8	0	0	0	1	0	
9	0	0	0	1	0	
10	0	0	0	0	1	

Deployment on Access Nodes based on previous step

e1 = 500

e2 = 300

e3 = 150

e4 = 210

Cost Coefficients found in Scenario 1

e1 = 330

e2 = 300

Cost Coefficients found in Scenario 2

Chapter 3

Conclusion:

In this project we analyzed various scenarios in which we can obtain optimal deployment of Virtual Network Functions in a Network Functions Virtualization Infrastructure. We found that there are pros and cons of deploying VNFs on both Core and Access nodes and only on Access nodes. Based on our given parameters, we got to know that deployment of VNFs on only Access nodes is cheaper but deploying on both sets of nodes provides more flexibility. Our analysis is in alignment with real-world scenarios wherein in case of capacity overload, creating a new access node and deploying VNFs on it is much cheaper and less complex than creating and deploying on a core node.

Chapter 4

Future Scope:

The algorithm can be further developed by introducing traffic and optimizing various paths, load balancing for improving network efficiency etc. Furthermore, users can be grouped based on various QoS parameters and VNFs can be optimally placed accordingly. On a final note, the same algorithm with minor changes and based on the same constraints can be used for optimal deployment of containers on an infrastructure that contains both bare-metal and virtual machines.