IllumiCore: Optimization Modeling and Implementation for Efficient VNF Placement

Leo Popokh
HPE & SMU
Dallas, USA
leonid.i.popokh@hpe.com

Jing Su, Suku Nair, Eli Olinick
SMU AT&T Center for Virtualization
Dallas, USA
[suj, nair, olinick]@smu.edu

***Abstract*—Network Function Virtualization (NFV) enables telecommunication operators to place and allocate resources dynamically to address the needs of Internet of Things (IoT), Intelligent Edge Computing (IEC) and emerging 5G services in a highly efficient manner. The efficient Virtual Network Function (VNF) placement and deployment largely depend on optimizing Virtual Machines (VM) compute, storage, and network resource allocation in the cloud-based platforms and their physical hosts. This research further extends our previously defined Information Model of mapped NFV Infrastructure (NFVI) and Virtualized Infrastructure Management (VIM) resources to derive VNF's placement and optimal resource allocation. Our optimization solution IllumiCore derives the VNF's optimal placement and minimizes the communication latency among VMs that are part of the VNF and entire communication network. The results demonstrate optimal and improved VNF placement and resource management.**

Keywords—NFV, VNF, VIM, NFVI, efficient VNF placement, resource allocation and management, communication latencies

I. INTRODUCTION

The conventional Data Centers (DCs) and communication network functions, such as routing, switching, firewall and load balancers are traditionally hardware-based and allocated for specific services. This reliance on hardware-based resources results in various deployment and maintenance challenges (i.e., slowing down the introduction of new and emerging services, increasing operating costs) [2]. The innovative methodologies called NFV, and Software-Defined Networking (SDN) were proposed by several world-leading telecommunication operators in 2012 [3], [4] to address these challenges. NFV [5] allows network operators to accelerate the deployment of new virtual devices, called VNFs, in a more dynamic and automated approach. VNFs play a vital role in today and future networks. They support a diverse set of functions ranging from Enterprise to Wireless networks. Optimizing the Compute, Storage, and Networking $\{C, S, N\}$ resources allocation in a coordinated way between physical and virtual hosting devices for VNFs in an NFV-based network architecture remains an open challenge [6]. The "Previous Work" section summarizes these challenges and gaps. Our review recognizes that different schemes and algorithms did not consider some essential parameters, such as combining the $\{C, S, N\}$ across physical and virtual platforms. Efficient VNF placement is particularly challenging, mainly for two reasons. First, depending on VNFs modeling, maturity, and standards compliance, end-to-end (E2E) latencies may become intolerable. Second, resource allocation is a cost and time-effective task. $\{C, S, N\}$ resources allocated to a VNF instance will impact the latency and performance of the VNF and corresponding Network Service (NS). Therefore, this paper formulates and implements the comprehensive Optimization Model and the Objective Function to minimize communication latency between VNF Components (VNFC) and their corresponding VMs based on the previously defined Information Model (IM) [1] of mapped physical and virtual resources to derive the hint for efficient VNF placement.

The rest of the paper is organized as follows. In Section II, we give an overview of previous and related work. Section III presents the optimization model with constraints and the objective function, and Section IV describes the details of our optimization solution. We discuss the complexity of VNF, VIM, NFVI parameters and constraints and formulate the optimization function as a constraint programming solution. We present use cases and scenarios and the evaluation results of our solutions in Section IV before concluding the paper in Section V.

II. PREVIOUS WORK

The VNFs deployment and VM placement is a well-studied and researched problem in the literature and industry due to its importance for the telecommunication operators. This section reviews the OpenStack approach for VM deployment, placement techniques, algorithms, and various resource allocation approaches introduced to realize it.

A. OpenStack

OpenStack [7], the popular open-source cloud operating system, is established to control and manage the converged (i.e., $\{C, S, N\}$ combined) resources. Even though OpenStack is a mature software platform, the developers still should overcome various challenges to orchestrate OpenStack resources and services. For example, when utilizing the Infrastructure as a Service (IaaS) resource provisioning, the developers need to figure out where and how they create VMs. In placing VMs, the default OpenStack resources schedulers (Nova, Heat, and Ceilometer) only consider computing loads and do not consider networking and storage conditions yet. The necessity of a network-aware scheduler has been discussed in the OpenStack community [8] and later in [24]. OpenStack Quality of Service (QoS) defines the ability to guarantee specific network requirements like bandwidth, latency, jitter, and reliability to satisfy a Service Level Agreement (SLA) between an application provider and end-users. Network devices such as switches and routers can mark traffic to handle a higher priority to fulfill the QoS conditions agreed under the SLA. In other cases, specific network traffic such as Voice over IP (VoIP) and

video streaming must be transmitted with minimal bandwidth constraints. On a system without network QoS management, all traffic will be transmitted in a "best-effort" manner, making it impossible to guarantee service delivery to customers. The example of extended OpenStack architecture for a dynamic resource allocation can be found in [9], where the authors provide an extensible set of management objectives. The system can switch at runtime during the process of resource allocation for interactive and computationally intensive applications. However, they do not address VNF deployment. In [10], the authors propose modifications to OpenStack's Nova scheduler to solve the NFVI resources' joint optimization problem. They introduce constraints for the VNF deployment related to QoS, fault-tolerance, and network topology redundancy, but they do not discuss the interaction between OpenStack and the network controller. IllumiCore enhances the OpenStack QoS approach with optimal resource allocation based on the IM from VIM and NFVI substrate layers and networks.

B. VNF Deployment

In [11], the authors present an optimization model for VNFs deployment as part of Virtual Mobile Core Networks. They face the problem of resource allocation for a core network service chain as a combination of VNF. In this work, latency is a combination of processing, packet queuing and propagation delay. The first two variables depend on the traffic utilization of the node the VNF is placed on, while the last one is a function of the path length. The assumption is to know in advance the precise input parameters, such as processing, packet queuing and propagation delay, but it is not the case. Another paper, [12], presents an Integer Linear Programming (ILP) model for VNF orchestration. The model is solved to determine the optimal number of VNFs and place them at the optimal locations to optimize network operational cost and resource utilization. The ILP model-based solution is suitable for small networks; however, more extensive networks' heuristics may not resemble large production networks. The work in [14] investigates the benefits of using two approaches: the NFV and SDN. The proposed VNF placement objective minimizes the total network load overhead by considering several parameters, such as the data plane delay and the SDN control overhead. In [15], the authors discuss applying constraint-based heuristics to deploy VNFs for Evolved Packet Core services (EPC). They show the results in terms of the average number of used CPU cores and aggregate throughput for placement strategies. In [16], the authors argue placing VNFs and learning algorithms for efficient replacement over time. We find this a very inefficient approach. It is challenging to reposition VNF after deployment into production.

C. VM Placement Techniques

Proper VM placement helps improving network resources efficiency, reduce communication delays and latency. Both [17] and [18] consider VNF placement and steer traffic through NS, while neither has the bandwidth and latency optimization. The work in [19] only focuses on initial placement by minimizing VM communication distance and setup cost, which ignores the VNF scaling problem. In [20], authors build a network-aware orchestration layer for virtual middleboxes. It considers a rack-aware VM placement, while its elastic placement strategy brings a lot of migration overhead. Perfect knowledge for all the parameters is assumed in all those models and algorithms, and optimal values are computed based on such precise input assumptions. However, if input parameters later vary, the optimal solution previously found may be infeasible, making those approaches impractical.

Our approach addresses VM deployment and placement techniques deficiencies. IllumiCore is modeling the $\{C, S, N\}$ efficient resource allocation for VNFs placement while minimizing the communications latencies between VMs and satisfying ETSI VNF operations (i.e., deploy, un-deploy, scale, heal and operate). We consider physical and network topology in the model, along with the compute and storage. Our approach is, in comparison, a broader, optimal solution and offers efficient and optimal placement of multiple VNF instances on-demand and chain service functions.

III. OPTIMIZATION MODEL

IllumiCore's mathematical optimization model attempts to minimize the latency between VMs that are part of the VNF without violating VNF, VIM and NFVI resource constraints. The proposed IllumiCore optimization model, objective function, algorithm, and resource allocation are proposed and implemented for VNFs and later for NS functional blocks.

A. Optimization Model Definition

In our approach, we based the optimization model for efficient VNF placement on the IM previously derived in [1]. "Fig. 1" describes the IM resources and substrate network (NFVI) topology discovery. VNF vnf_i deployment request (step #1) triggers the IM update from VIM and NFVI layers. Requested VNF vnf_i will require certain number of virtual-and-required resources:

$$R_{vm_{i.j}} \exists \left\{C^{vr}_{vm_{i.j}}, S^{vr}_{vm_{i.j}}, N^{vr}_{vm_{i.j}}\right\} \mid vm_{i.j} \in vnfc_{i.j} \in vnf_i$$

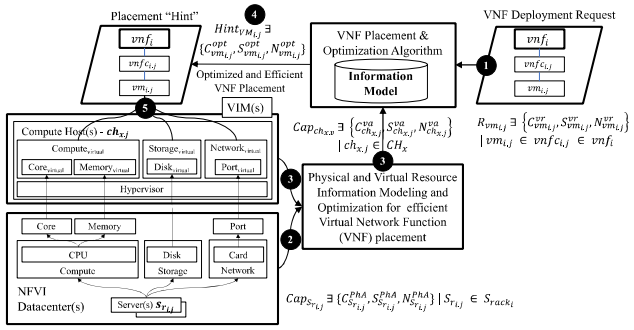

Fig. 1. Information Model, resources and topology discovery

We discover all virtual resources from the VIM (step #3) and define them as follows:
$Cap_{ch_{k.j}} \exists \{C^{va}_{ch_{k.j}}, S^{va}_{ch_{k.j}}, N^{va}_{ch_{k.j}}\} \mid ch_{k.j} \in CH_x$. There exists Compute Host $ch_{k.j}$ with capacity $Cap_{ch_{k.j}}$ at each VIM_k in terms of virtual-and-available Compute - $C^{va}_{ch_{k.j}}$, Storage - $S^{va}_{ch_{k.j}}$ and Network - $N^{va}_{ch_{k.j}}$, such that every Compute Host (CH) $ch_{k.j}$ belongs to the set of CH_k. "TABLE II" (9, 10)

We discover all physical resources from NFVI (step #2) and define them as follows:
$Cap_{S_{r_{i.j}}} \exists \{C^{PhA}_{S_{r_{i.j}}}, S^{PhA}_{S_{r_{i.j}}}, N^{PhA}_{S_{r_{i.j}}}\} \mid S_{r_{i.j}} \in S_{rack_i}$. There exists

Physical Server $S_{r_{i.j}}$ with capacity $Cap_{S_{r_{i.j}}}$ at each VIM_x in terms of Physical-and-Available Compute - $C^{PhA}_{S_{r_{i.j}}}$, Storage - $S^{PhA}_{S_{r_{i.j}}}$ and Network - $N^{PhA}_{S_{r_{i.j}}}$, such that every server $S_{r_{i.j}}$ belongs to the set of physical server in the rack - S_{rack_i}. "TABLE III" (23).

We describe IllumiCore's optimization model mathematical formulations with inputs (i.e., VNF deployment request: $R_{vm_{i.j}} \exists \left\{C^{vr}_{vm_{i.j}}, S^{vr}_{vm_{i.j}}, N^{vr}_{vm_{i.j}}\right\} | \, vm_{i.j} \in vnfc_{i.j} \in vnf_i$ for each virtual machine $vm_{i.j}$ requested to be deployed $R_{vm_{i.j}}$, there exists a set of virtual & required compute : $C^{vr}_{vm_{i.j}}$, storage: $S^{vr}_{vm_{i.j}}$ and network : $N^{vr}_{vm_{i.j}}$, such that every virtual machine $vm_{i.j}$ belongs to a VNFC $vnfc_{i.j}$ and each VNFC belongs to a VNF - vnf_i), constraints and objectives. IllumiCore utilizes a constraint programming solution (step #4) to find the optimal VNF placement and produce the hint to NFV Orchestrators for VNF efficient and optimal deployment (step #5). We analyze and relate physical and virtual information not only in terms of compute nodes (i.e., VM to PM mapping) but all the locations, distances, and network connectivity across both physical and virtual resources as VNF building blocks.

Our objective function is to minimize the latency between VMs that are part of the VNF:

$$Min\left(Lat_{vnf_i}\right) = Min\left(\sum_{vlink_l} \sum_{vm_{i.j} \exists \, vnfc_{i.j} \in vnf_i} x_{vm_{i.j}ch_{x.v}} \; * \; lat_{vlink_l}\right)$$

IllumiCore's efficient placement algorithm generates the Hint with optimal virtual $\{C, S, N\}$ resources allocations for VNF deployment - $Hint_{vm_{i.j}} \exists \{C^{opt}_{vm_{i.j}}, S^{opt}_{vm_{i.j}}, N^{opt}_{vm_{i.j}}\}$ for each $vm_{i.j}$

B. Optimization Model Formulations, Input, Constraints and Objective

1) Optimization Model Input

TABLE I. VNF PARAMETERS

(1)	There are many VNFs in the telecommunications operators' networks, both Enterprise and Wireless. Notation vnf_i represents the i^{th} VNF in set VNF $VNF = \{vnf_1, vnf_2, \ldots, vnf_i, \ldots, vnf_n\}$
(2)	There could be one or many (i.e., n) VNFs to be deployed at the same time. The number of VNFs to be deployed is $n = \lvert VNF \rvert$
(3)	Total resources required from the VIM for the n number of VNFs to be deployed. $R_{VNF} = \sum_{i=1}^{n} R_{vnf_i}, \lvert \, vnf_i \in VNF$
(4)	R_{vnf_i} defines the total resources required for the single vnf_i deployment. $R_{vnf_i} = \sum_{j=1}^{m} R_{vnfc_{i.j}} = \sum_{j=1}^{m} R_{vm_{i.j}}$ where m is the total number of VNFCs and VMs for VNF vnf_i
(5)	For each $vm_{i.j}$, there are virtual-required resources: $C^{vr}_{vm_{i.j}}$: compute, $S^{vr}_{vm_{i.j}}$: storage and $N^{vr}_{vm_{i.j}}$: Network $R_{vm_{i.j}} \exists \left\{C^{vr}_{vm_{i.j}}, S^{vr}_{vm_{i.j}}, N^{vr}_{vm_{i.j}}\right\} | \, vm_{i.j} \in vnfc_{i.j} \in vnf_i$ $R_{vm_{i.j}}$ defines the total resources required for $vm_{i.j}$ deployment
(6)	Placement Policy (Affinity / Anti-Affinity) for vnf_i deployment. $P_{vnf_i} = \{1, 2, 0\} \in policy$ $policy = \begin{cases} 1, & \text{affinity} \\ 2, & \text{anti-affinity} \\ 0, & \text{otherwise} \end{cases}$
(7)	$MaxLat_{vnf_i}$ defines the Maximum Latency each vnf_i can tolerate
(8)	$MaxLat_{vm_{i.j}}$ defines the Maximum Latency each $vm_{i.j}$ can tolerate

TABLE II. VIM PARAMETERS

(9)	There could be one or many VIMs in $\{vim_1, \ldots, vim_k, \ldots, vim_w\}$ in the VIM set implemented across one or many data centers (DC): $VIM = \{vim_1, \ldots, vim_k, \ldots, vim_w\}$. CHs are defined per each VIM (i.e., vim_k). $vim_k \in VIM$ and $w = \lvert\{VIM\}\rvert$ is the number of VIMs. CH is the abstract definition of a server in OpenStack. There is a set of $CH_k = \{ch_{k.1}, \ldots, ch_{k.j}, \ldots, ch_{k.v}\}$ as part of the vim_k and $h = \lvert CH_x \rvert$ is the total number of CHs per individual vim_k
(10)	CH $ch_{k.j}$ will have its capacity of virtual-available {C, S, N} resources available for VM hosting/instantiation: $Cap_{ch_{k.j}} \exists \{C^{va}_{ch_{k.j}}, S^{va}_{ch_{k.j}}, N^{va}_{ch_{k.j}}\} | \, ch_{k.j} \in CH_k$

TABLE III. NFVI PARAMETERS

(11)	Set of Datacenters: $DC = \{dc_1, \ldots, dc_i, \ldots, dc_d\}$
(12)	Physical rack in the dc_i: $RACK_{dc_i} = \{r_{dc_{i.1}}, \ldots, r_{dc_{i.j}}, \ldots, r_{dc_{i.r}}\}$
(13)	Total number of racks across all DCs: $racks_{total} = \sum_{i=1}^{\lvert DC \rvert} \lvert dc_i \rvert$
(14)	Set of physical servers enclosed in the Rack with the single Top of the Rack (TOR). There could be one or many servers in the rack: $S_{rack_i} = \left\{s_{r_{i.1}}, \ldots, s_{r_{i.j}}, \ldots, s_{r_{i.p}}\right\}$, $rack_i$ is from $RACK_{dc_i}$ set
(15)	Number of servers per rack: $num_{ser_{rack}} = \lvert S_{rack_i} \rvert$
(16)	Total number of servers: $servers_{total} = num_{ser_{rack}} * racks_{total}$
(17)	There is a set of TORs per $dc_{i.j}$: $TOR_{dc_i} = \{tor_{dc_{i.1}}, \ldots, tor_{dc_{i.j}}, \ldots, tor_{dc_{i.k}}\}$
(19)	Set of aggregation Switches per DC $dc_{i.j}$: $SW_{AGG_{dc_{i.j}}} = \left\{sw_{agg_{dc_{i.1}}}, \ldots, sw_{agg_{dc_{i.j}}}, \ldots, sw_{agg_{dc_{i.s}}}\right\}$
(20)	Number of the sw_agg per DC dc_i : $num_{sw\_agg_{dc_i}} = \lvert SW\_AGG_{dc_{i.j}} \rvert$
(22)	$sw_{core} = 1$ Core Switch. There is one active Core Switch in the DC architecture.
(23)	Each DC's dc_i physical Server will have its Physical-Available {C, S, N} resources: $Cap_{S_{r_{i.j}}} \exists \{C^{PhA}_{S_{r_{i.j}}}, S^{PhA}_{S_{r_{i.j}}}, N^{PhA}_{S_{r_{i.j}}}\} | \, S_{r_{i.j}} \in S_{rack_i}$

TABLE IV. OTHER VARIABLE & PARAMETERS

(24)	Binary decision variable denoting if VM $vm_{i.j}$ is deployed on the CH $ch_{x.j}$: $x_{vm_{i.j}ch_{x.j}} = \begin{cases} 1, \text{if } vm_{i.j} \text{ is allocated to } ch_{x.j} \\ 0, \text{otherwise} \end{cases}$
(25)	Total time for vnf_i resource allocation: $T_{R_{vnf_i}}$
(26)	Size of the problem: i.e., number of DCs (c), number of VIMs (w), number of racks, servers, networks, and number of VNFs (n): $Size = \{d, w, n\}$

2) Constraints

TABLE V. VNF CONSTRAINTS

(27)	Each vnf_i will have a set $VNFC_i$ with VNFC belonging to it and a set VM_i with Virtual Machines $VM_i \subseteq VNFC_i \in vnf_i$ $\{vm_{i.1}, \ldots, vm_{i.j}, \ldots, vm_{i.m}\} \subseteq \{vnfc_{i.1}, \ldots, vnfc_{i.j}, \ldots, vnfc_{i.m}\} \in vnf_i$
(28)	Each vnf_i has a set of $VNFC$. There is 1: m relationship between VNF and its VNFCs. m is the number of VNFCs in the VNF. Each VNF can have a different number of VNFCs. $\forall \, vnf_i \in VNF \; \exists \, VNFC_i$ $VNFC_i = \{vnfc_{i.1}, \ldots, vnfc_{i.j}, \ldots, vnfc_{i.m}\}$ $m = \lvert\{vnfc_{i.1}, \ldots, vnfc_{i.m}\}\rvert$.
(29)	Each VNFC runs on a VM. The $VNFC_i$ set has the same cardinality as the VM_i set. There is a bijection (i.e., 1:1 correspondence) from the $VNFC_i$ to VM_i set. Each $vm_{i.j}$ of the VM_i set is paired with exactly one $vnfc_{i.j}$ from the $VNFC_i$ set. There are no unpaired VNFC and VM. $VM_i = \{vm_{i.1}, \ldots, vm_{i.j}, \ldots, vm_{i.m}\}$ $m = \lvert\{vnfc_{i.1}, \ldots, vnfc_{i.j}, \ldots, vnfc_{i.m}\}\rvert = \lvert\{vm_{i.1}, \ldots, vm_{i.j}, \ldots, vm_{i.m}\}\rvert$ $\forall \, vnfc_{i.j} \exists \, vm_{i.j} \mid vnfc_{i.j} \in VNFC_i \wedge vm_{i.j} \in VM_{i.j} \wedge f: VNFC_i \rightarrow VM_i$
(30)	$VLINK = \{vlink_1, \ldots, vlink_l, \ldots, vlink_k\} \in vnf_i$ $vlink_l \subseteq \{(vm_{i.j}, vm_{i.m}) \mid (vm_{i.j}, vm_{i.m}) \in VM_i\} \in VLINK$ Virtual Link $vlink_l$ is an edge associated with two distinct VMs comprising vnf_i; such that $\{(vm_{i.j}, vm_{i.m}) | (vm_{i.j}, vm_{i.m}) \in VM_i$

TABLE VI. NFVI CONSTRAINTS

(31)	TOR connects servers in the rack and serves as access to the DC's aggregation network. There is a bijection (i.e., 1:1 correspondence) from the $tor_{dc_{i,j}}$ to $r_{dc_{i,j}}$. Each $tor_{dc_{i,j}}$ is paired with exactly one $r_{dc_{i,j}}$ rack. There are no unpaired TORs and Racks. $\forall\, r_{dc_{i,j}} \exists\, tor_{dc_{i,j}} \mid tor_{dc_{i,j}} \in TOR_{dc_i} \wedge r_{dc_{i,j}} \in RACK_{dc_i}$
(32)	$V = \{SWITCH_{CORE}, SWITCHES_{AGG}, TORs, SERVERS\}$ $NFVI = (V, E), v \in V(NFVI)$ $N_{NFVI}(v) = \{u \in V(NFVI) \mid v, u \in E(NFVI)\}$ $N_G[v] = N(v) \cup \{v\} = \{u \in V \mid u, v \in E\} \cup \{v\}$ Next Hop Neighbor (NHN) at the physical level, among servers, TORs, and switches. $\{u \in V \mid u, v \in E\}$ all the neighbors $\{u \in V(NFVI) \mid v, u \in E(NFVI)\}$ all vertices adjacent to v $N_G[v]$ closed neighborhood of v or NHN, N – neighborhood
(33)	The path is the total number of connected hops from one VM to another. We will assign a different weight for the hops: $w = 1$: within the same server, $w = 2$: from server to server in the same rack via TOR, $w = 4$: from server to server in different racks via respective rack's TORs and first-level Switch in the same datacenter, $w = 6$: from server to server in different racks via respective rack's TORs, first-level Switch and the 2nd level/aggregation Switch in the same datacenter
(34)	There is 1:1 mapping between a physical server and CHs $Cap_{S_{r_{i,j}}} \exists \{C^{PhA}_{S_{r_{i,j}}}, S^{PhA}_{S_{r_{i,j}}}, N^{PhA}_{S_{r_{i,j}}}\} \mid S_{r_{i,j}} \in S_{rack_i}$ = $Cap_{ch_{x,j}} \exists \{C^{va}_{ch_{x,j}}, S^{va}_{ch_{x,j}}, N^{va}_{ch_{x,j}}\} \mid ch_{x,j} \in CH_x$

TABLE VII. DEPLOYMENT CONSTRAINTS

(35)	This constraint defines that all VNFs with their corresponding VMs are allocated to hosting devices. This constraint also ensures that VNFs cannot be placed on hosting devices if their capacity exceeds $\sum_{vnf_i \in VNF} vnf_i * R_{vnf_i} \leq Cap_{ch_{k,v}},\ \forall\, ch_{k,v} \in CH_k$
(36)	This constraint refers that the accumulated latency requirement is less than the VNFs maximum capacity $\sum_{ch_{k,v} \in CH_k} vnf_i * Lat_{vnf_i} \leq MaxLat_{vnf_i},\ \forall\, vnf_i \in VNF$
(37)	"M" is the total number of VMs and their respective VNFCs that are connected as part of the VNF $\sum_{vm_{i,j} \exists\, vnfc_{i,j} \in vnf_i} \sum_{ch_{k,v} \in CH_k} = M$
(38)	This constraint ensures that all the VNFs are allocated to exactly one hosting device, if possible. $\sum_{ch_{k,v} \in CH_k} vnf_i \leq 1,\ \forall\, vnf_i \in VNF$
(39)	This constraint ensures that the optimal {C, S, N} placement is within VIM's offered and available virtual {C, S, N} $Hint_{vm_{i,j}} \leq Cap_{ch_{k,v}}, \forall \{C^{opt}_{vm_{i,j}}, S^{opt}_{vm_{i,k}}, N^{opt}_{vm_{i,j}}\} \in \{C^{va}_{ch_{k,v}}, S^{va}_{ch_{k,v}}, N^{va}_{ch_{k,v}}\}$

3) *Objective Function*

TABLE VIII. OBJECTIVE FUNCTION

(40)	Minimize the latency between VMs that are part of the vnf_i $Min(Lat_{vnf_i}) = Min\left(\sum_{vlink_l} \sum_{vm_{i,j} \exists\, vnfc_{i,j} \in vnf_i} x_{vm_{i,j} ch_{k,v}} * lat_{vlink_l}\right)$

IV. OUR SOLUTION

IllumiCore achieves the desired objective function with the extensive set of complex VNF, VIM and NFVI constraints. For its constraint programming solution, the IllumiCore utilizes the OR-Tools [21]. It is an open-source software suite originally developed by Google for optimization, tuned for tackling the world's most challenging optimization problems in routing, flows, integer, linear programming, and in our case, constraint programming. IllumiCore achieves optimal solution feasibility focusing on the virtual and physical constraints and variables.

"Algorithm 1" presents the pseudocode for IllumiCore optimal VNF placement implementation solution. We first declare the model and create all required variables. We then define VNF, VIM and NFVI constraints and objective function. The CpSolver provides the optimal and efficient {C, S, N} VIM resources allocations for VNF deployment.

Algorithm 1: IllumiCore Algorithm and pseudocode

```
Import packages & Setup environment
    ORTools: Constraint Programming
Initialize global variables
    Total time for resource allocation ≤ 3 mins
Define data model
    VNF data model
    VIM data model
    NFVI layer data model
Input VNF deployment request & IM numerical data for VIM & NFVI
Define Constraint Programming Solver
    Define next-hop neighbor
    Add resources constraint
    Add affinity constraint
    Add deployment constraints
    Add VM max latency constraint
    Add VNF max latency constraint
Set Objective Function
    Minimize the latency between VMs that are part of the VNF
Output assignment status, statistics & results
    Result: VM to CHs and Servers' assignment matrix
```

A. *Implemented Architecture*

A DC is a pool of $\{C, S, N\}$ resources clustered together using communication networks to host applications and store data. The DC's primary information and communication technology components are servers and network infrastructure. The DC network (DCN) is typically based on a three-tier architecture. Three-tier DC architecture is a hierarchical tree-based structure comprised of three layers of switching and routing elements having enterprise-class high-end equipment in higher layers of the hierarchy. An example of the three-tier DCN architecture is shown in "Fig. 2" and implemented in our IllumiCore solution.

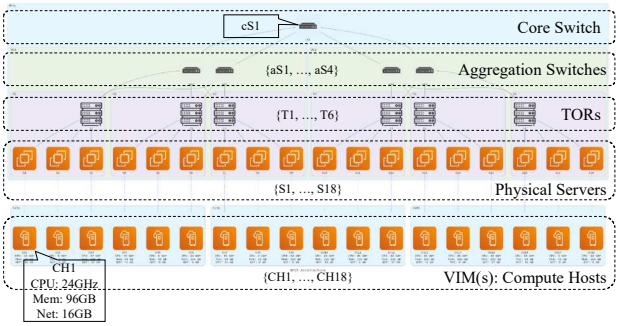

Fig. 2. Implemented NFVI / Datacenters & VIM architecture

Two DCs {DC1, DC2} are in 2 different geographical locations and connected over the network with three-tier network architecture – "Fig. 2". Each DC will have the aggregation switches {aS1, aS2} in DC1 and {aS3, aS4} in DC2 connected to the core switch - cS1 and TORs {T1, … T6} inside the DC's racks {R1, …, R6}. The internal rack network communications among servers within the rack are going through the TOR. The aggregation switches allow us to communicate among the Racks. The core switch allows us the connectivity between the datacenters. The VIM layer is implemented with the OpenStack. VIM's CHs {CH1, …, CH18} are mapped 1:1 to the NFVI layer's physical servers {S1, …, S18} - "Fig. 2".

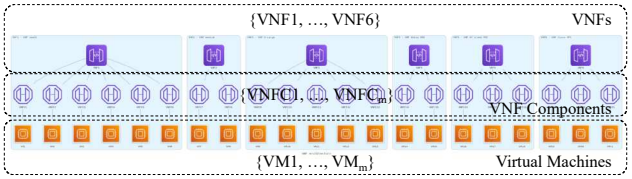

Fig. 3. Requested VNF placement example

The VMs {VM1, …, VM_m} that are part of VNFs are placed and deployed on VIM's CHs. According to ETSI standards, VNFs consist of VNFCs with 1:m mapping between VNF and VNFC. Each VNFC will have its VM with 1:1 mapping – "Fig. 3".

We measure distances at the NFVI across VMs in terms of hops. There is a direct correlation between the number of hops and the communication latency. Hop count is the number of routes through which a packet passes while traversing from the source to the destination host. The hop count in the optimization model is weighted (Table VI, 33). For example, a hop between DCs has much more latency than a hop between servers that are in a rack together. The critical, innovative approach proposed in our IM [1] was to discover the virtual resources out of the VIMs. Then map them into the related physical resources identified from the NFVI and create one common catalog view. This catalog will provide the mapping across virtual and physical resources across compute, storage and network dimensions and reflect the distances across these resources to minimize the communications latencies across the VMs using these resources. Hops' distance impacts virtual deployments even more than physical deployments as the VNFs are divided into components and VMs. There are more chances for them to be scattered across servers, racks, and data centers. We measure distances across VMs in terms of hops from each other. There is a direct correlation between the number of hops and the communication latency [22]. Bandwidth on the hop can be increased, but the latency has the speed of light [23].

For example, if the VMs are placed on the same server, the compute distance is 0, as all compute communications are across Intra-Server compute resources. Another example is that if VMs are placed on two different servers in the same rack, all compute communications will be across Inter-Server and Intra-Rack through the TOR. The hop distance will be 2 (from server1 to TOR and from TOR to server2). The distance between 2 VMs in two different racks is 4 (from VM in server1 (rack 1) to TOR1, from TOR1 to agg_Switch1, from agg_Switch1 to TOR2, and finally from TOR2 to VM in server5 (rack 2)). Inter-Server and Intra-Rack distances and hops are the same unless a stand-alone server is outside of the rack. The same applies to Inter-Rack and Intra-DC distances and hops. Our current research does not consider stand-alone servers outside of the racks, as this is not a traditional DC implementation.

In our research, we are considering only local storage for the storage distance estimations.

B. Use Cases and Scenarios

We executed an extensive set of test cases to generate the hint for the efficient and optimal placement for six different VNFs – "Fig. 3". We are considering three VNFs based on the OpenStack default flavors (i.e., small, medium, and X-Large) and three industry VNFs (i.e., Nokia HSS, Affirmed MCC and Cisco VPC). All six VNFs are different in their virtual compute, memory, and network requirements.

TABLE III. TYPES OF REQUESTED VNFS AND VIRTUAL MACHINES

VNF		small			medium		X-Large				Nokia HSS		Affirmed MCC			Cisco VPC		
VNF Components		1	2	3	4	5	6	1	2	1	2	3	4	5	1	2	1	2	3	1	2	3
VM		1	2	3	4	5	6	1	2	1	2	3	4	5	1	2	1	2	3	1	2	3
vCPU	GHz	4	4	2	4	4	8	2	2	4	2	2	4	8	8	8	8	16	20	8	8	8
vMemory	GB	1	6	1	1	6	1	1	1	2	2	2	1	1	16	16	32	62	56	16	16	16
vNetwork	GB	1	1	5	1	1	1	3	1	3	4	5	5	3	3	3	3	3	3	3	3	3

TABLE IV. TYPES OF NFVI PHYSICAL MACHINES

"Table III" presents the VNFs used in our testing. The numbers used in "Table III" present the number of VNFCs that belong to a VNF. For example, a small VNF has 6 VNFC and 6 VMs. Each VM has virtual required vCPU, vMemory, and vNetwork requirements. For example, small VNF's VM1 requires 4GHz of vCPU, 1 GHz of vMemory, and 1GB of vNetwork.

		S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10	S 11	S 12	S 13	S 14	S 15	S 16	S 17	S 18
phy avail CPU	GHz	24	6	32	16	8	64	16	8	64	48	48	48	48	48	48	16	8	64
phy avail Mem	GB	96	4	128	64	32	256	64	32	256	256	256	256	256	256	256	64	32	256
phy avail Net	GB	16	6	32	16	8	8	16	8	8	16	6	32	16	6	32	16	8	8

"Table IV" describes 18 physical servers {S1, …, S18} we are using on our NFVI layer with their respective physical available CPU, Memory and Networking resources availability. For example, server 1 (i.e., S1) has 24Ghz physical and available CPU, 96GB physical and available memory, and 16GB of physical and available Network connectivity.

TABLE V. TYPES OF VIM COMPUTE HOSTS

		CH 1	CH 2	CH 3	CH 4	CH 5	CH 6	CH 7	CH 8	CH 9	CH 10	CH 11	CH 12	CH 13	CH 14	CH 15	CH 16	CH 17	CH 18
VIM ID		1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
virt avail CPU	GHz	24	6	32	16	8	64	16	8	64	48	48	48	48	48	48	16	8	64
virt avail Mem	GB	96	4	128	64	32	256	64	32	256	256	256	256	256	256	256	64	32	256
virt avail Net	GB	16	6	32	16	8	8	16	8	8	16	6	32	16	6	32	16	8	8

"Table V" describes 18 CHs {CH1, …, CH18} in three different VIMs with their corresponding virtual and available CPU, memory, and network.

C. Results

IllumiCore provides (a) OPTIMAL, (b) FEASIBLE, and (c) INFEASIBLE solutions. In the OPTIMAL solution, the Objective Function score is based on the latency hops between the VMs. Additionally, the OPTIMAL solution provides the placement Hint with text – "Fig. 4" and graphical placement examples – "Fig.5".

```
Assign Result

TOR [ 1] -> Server [ 1]
VIM [ 1] -> Compute Host [ 1] (CPU:  24 GHz, Memory:  96 GB, Network:  16 GB)
        VNF [ 6] (Policy: 0, Max Latency: 25) -> VNFC [19] -> VM [19] (CPU:   8 GHz, Memory:  16 GB, Network:   3 GB, Max Latency:  7), Latency = 1
        VNF [ 6] (Policy: 0, Max Latency: 25) -> VNFC [20] -> VM [20] (CPU:   8 GHz, Memory:  16 GB, Network:   3 GB, Max Latency:  7), Latency = 1
        VNF [ 6] (Policy: 0, Max Latency: 25) -> VNFC [21] -> VM [21] (CPU:   8 GHz, Memory:  16 GB, Network:   3 GB, Max Latency:  7), Latency = 1
> Total Occupied:  3 VM (CPU:  24 GHz, Memory:  48 GB, Network:   9 GB)
> Total Remaining:      (CPU:   0 GHz, Memory:  48 GB, Network:   7 GB)


TOR [ 1] -> Server [ 2]
VIM [ 1] -> Compute Host [ 2] (CPU:   6 GHz, Memory:   6 GB, Network:   6 GB)
> Unoccupied


TOR [ 1] -> Server [ 3]
VIM [ 1] -> Compute Host [ 3] (CPU:  32 GHz, Memory: 128 GB, Network:  32 GB)
        VNF [ 2] (Policy: 0, Max Latency: 25) -> VNFC [ 7] -> VM [ 7] (CPU:   2 GHz, Memory:   1 GB, Network:   3 GB, Max Latency:  7), Latency = 1
        VNF [ 2] (Policy: 0, Max Latency: 25) -> VNFC [ 8] -> VM [ 8] (CPU:   2 GHz, Memory:   1 GB, Network:   1 GB, Max Latency:  9), Latency = 1
        VNF [ 3] (Policy: 0, Max Latency: 18) -> VNFC [ 9] -> VM [ 9] (CPU:   4 GHz, Memory:   2 GB, Network:   3 GB, Max Latency: 11), Latency = 1
        VNF [ 3] (Policy: 0, Max Latency: 18) -> VNFC [10] -> VM [10] (CPU:   3 GHz, Memory:   2 GB, Network:   4 GB, Max Latency:  9), Latency = 1
        VNF [ 3] (Policy: 0, Max Latency: 18) -> VNFC [11] -> VM [11] (CPU:   2 GHz, Memory:   2 GB, Network:   5 GB, Max Latency: 12), Latency = 1
        VNF [ 3] (Policy: 0, Max Latency: 18) -> VNFC [12] -> VM [12] (CPU:   4 GHz, Memory:   1 GB, Network:   5 GB, Max Latency: 13), Latency = 1
        VNF [ 3] (Policy: 0, Max Latency: 18) -> VNFC [13] -> VM [13] (CPU:   8 GHz, Memory:   1 GB, Network:   3 GB, Max Latency:  9), Latency = 1
> Total Occupied:  7 VM (CPU:  24 GHz, Memory:  16 GB, Network:  24 GB)
> Total Remaining:      (CPU:   8 GHz, Memory: 112 GB, Network:   8 GB)
```

Fig. 4. A snippet of IllumiCore placement hint in text form (Note: this is not a full-text output result)

The FEASIBLE solution provides the placement suggestion, but not OPTIMAL, and exceeds the resource allocation time window. IllumiCore will not provide the resource allocation and VNF placement hint in the INFEASIBLE result based on the input and CH virtual-and-available resources. For each VM $vm_{i.j}$, IllumiCore will generate the Hint with optimal virtual $\{C, S, N\}$ resources allocations: $Hint_{vm_{i.j}} \exists \{C^{opt}_{vm_{i.j}}, S^{opt}_{vm_{i.j}}, N^{opt}_{vm_{i.j}}\}$. The results also demonstrate VIM's current CHs virtual and available CPU, Memory, and Network. Upon resource allocation for VNF placement, the results demonstrate the hosts' total occupied and total remaining resource capacity.

IllumiCore utilizes Diagrams python module that allows IllumiCore to express input and output results diagrams as Code and graphical form. "Fig. 5" presents the output of IllumiCore optimization placement results across all layers of NFV (VNF, VIM, and NFVI). As a result, IllumiCore presents VNF's VMs mapped to their CHs for optimal placement. We also present how the CHs are mapped to their respective physical services and from where these physical services are deployed based on our previously defined and discovered IM.

IllumiCore stores all the testing and optimization results in text and graphical form. Text results are stored for all VNF placements requests, and graphical results are stored only for OPTIMAL and FEASIBLE results.

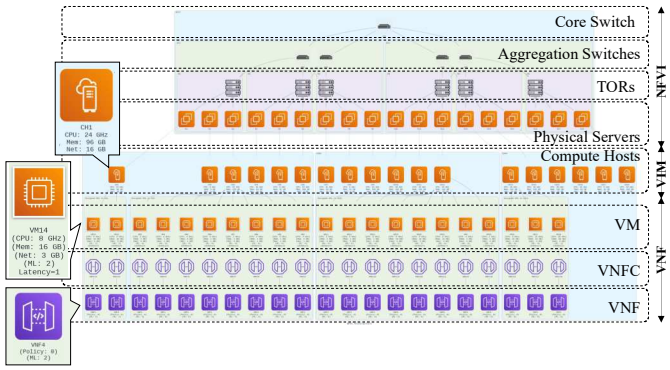

Fig. 5. Graphical hint representation for optimal and efficient VNF placement.

D. Comparisons

To evaluate our IllumiCore optimal VNF placement and efficient resource allocation, we compare IllumiCore's optimal results output vs. the simulated VIM-only-based VNF placement. "Fig. 6" presents an example output from IllumiCore VNF optimization placement. We are comparing the results of IllumiCore efficient VNF placement (i.e., Optimal Placement) vs. VIM-only-based default placement (without NFVI constraints and knowledge of physical substrate network). We present 18 CHs {CH1, …, CH18} allocated across three different VIMs {VIM1, VIM2, VIM3}. We overlay VIMs with DC's NFVI infrastructure. The CHs are mapped 1:1 to the physical servers. Each server is in the racks {Rack1, …, Rack6} and across two DCs {DC1, DC2}. IllumiCore optimally places VNF1 and VNF2 on the CH3 (hop count = 0) vs. CH4 and CH10 with the hope count = 6 (due to physical resource fulfillment from different servers, racks, and even DCs). Similarly, IllumiCore places VNF3 & VNF6 on CH12 (hope count = 0) vs. VIM-based placement on CH1 and CH3 with the hop count = 4. Additionally, IllumiCore places VNF4 on CH1 vs. CH7, allowing for VNF4 scaling within CH1 vs. allocating and depleting all CH7 resources in the VIM-based mode.

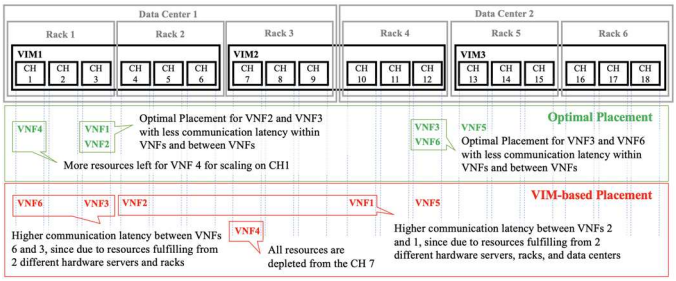

Fig. 6. Testing results

In contrast, testing VNF placement without IllumiCore placement and based on the default VIM-based placement algorithms (Heat, Nova, and Ceilometer) provided VNF placement on the distant hardware servers resulting in larger inter-server, rack, and data centers communication latencies and delays.

V. Conclusions and Future Work

We presented a comprehensive approach for defining, creating, and implementing the Optimization Model to minimize the latency between VMs that are part of the VNF. We further use the developed Optimization Model for efficient and optimal VNF placement based on the compute, storage, and network granularity while considering both virtual and physical layer resources and constraints. IllumiCore reduces communication latencies and delays by allocating resources at the physical layer in close proximity (i.e., intra-server communications vs. inter-server/intra-rack and vs. intra-DC) to improve overall VNF performance.

While IllumiCore helps to allocate resources for VNF placement efficiently, it could be further improved with predictive Machine Learning (ML) implementation. In our future work, we are planning to use ML to automatically learn from real VNF, VIM, and NFVI data, deriving models that can accurately predict optimal {C, S, N} resource requirements for the efficient VNF Placement and further enhance the fidelity of VNF latency modeling. ML for VNF dynamic resource allocation can help reduce the VNF latency and significantly improve the network service that the VNF is part of. We are considering assembling a learning machine with this archived data, which will allow us to find "similar" known VNFs. If there is a close match, we will examine the placement scheme (based on the compute, storage, and network virtual and physical resources) and reuse it. Approximate Nearest Neighbor (ANN) problem will help us identify these virtual and physical resources for the VNF deployment. We plan to test and investigate batch vs. sequential order of arrival for VNF placement. While the ML for real-time resource allocation and optimization for presently available resources is one approach, we consider ML for predictive VNF placement. Placing and assigning resources for efficient VNF placement ML will predict what similar VNF can arrive for deployment and leave out required resources for future VNF placement that could be

part of the same Network Service. If the optimization process is slow and takes more than the allocated time, we may opt out for the default OpenStack Nova placement. Still, with the ML, we can improve the optimization running time.

REFERENCES

[1] L. Popokh, P. Olive, I. Aldama, Y. Al-Doori and S. Nair, "Physical and Virtual Resources Inventory Modeling for Efficient VNF Placement," *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, 2020, pp. 1-6, doi: 10.1109/ICCE-Berlin50680.2020.9352159.

[2] C. Zhang, H. P. Joshi, G. F. Riley, and S. A. Wright, "Towards a virtual network function research agenda: A systematic literature review of vnf design considerations," Journal of Network and Computer Applications, p. 102417, 2019.

[3] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an sdn-enabled nfv architecture," IEEE Communications Magazine, vol. 53, no. 4, pp. 187–193, 2015.

[4] K. S. Ghaia, S. Choudhurya, and A. Yassineb, "A stable matching based algorithm to minimize the end-to-end latency of edge nfv," Procedia Computer Science, vol. 151, pp. 377–384, 2019.

[5] ETSI White Paper No. #32. Network Transformation;. (Orchestration, Network and. Service Management. Framework). 1st edition – October-2019. ISBN No 979-10-92620-29-0. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_White_Paper_Network_Transformation_2019_N32.pdf

[6] J.G.Herreraand, J.F.Botero, "Resource allocation in nfv: A comprehensive survey," IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518–532, 2016.

[7] Openstack. [Online]. Available: http://openstack.org

[8] "Network-aware scheduler in OpenStack," [Online]. Available: https://blueprints.launchpad.net/nova/+spec/network-aware-scheduler/

[9] F. Wuhib, R. Stadler, and H. Lindgren, "Dynamic resource allocation with management objectives: Implementation for an openstack cloud," in Proceedings of the 8th International Conference on Network and Service Management, ser. CNSM '12. Laxenburg, Austria, Austria: International Federation for Information Processing, 2019, pp. 309–315. [Online]. Available: http://dl.acm.org/citation.cfm?id=2499406.2499456

[10] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "Morsa: A multi-objective resource scheduling algorithm for NFV infrastructure," in Network Operations and Management Symposium (APNOMS), 2019 16th Asia-Pacific, Sept 2019, pp. 1–6.

[11] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Combined virtual mobile core network function placement and topology optimization with latency bounds," in Software Defined Networks (EWSDN), 2015 Fourth European Workshop on, Sept 2015, pp. 97–102.

[12] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in NFV," CoRR, vol. abs/1503.06377, 2015. [Online]. Available: http://arxiv.org/abs/1503.06377

[13] A. P. Bianzino, C. Chaudet, D. Rossi, J. L. Rougier, and S. Moretti, "The green-game: Striking a balance between qos and energy saving," in Teletraffic Congress (ITC), 2011 23rd International, Sept 2011, pp. 262–269.

[14] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE Mobile Core Gateways, the Functions Placement Problem," in Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges, ser. AllThingsCellular '14. New York, NY, USA: ACM, 2014, pp. 33-38. [Online]. Available: http://doi.acm.org/10.1145/2627585.2627592

[15] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch, "Cost analysis of initial deployment strategies for virtualized mobile core network functions," IEEE Communications Magazine, vol. 53, no. 12, pp. 60–66, Dec 2015.

[16] Sheoran, Amit & Sharma, Puneet & Fahmy, Sonia & Saxena, Vinay. (2017). Contain-ed: An NFV Micro-Service System for Containing e2e Latency. ACM GICCOMM Computer Communication Review. 47. 54-60. 10.1145/3094405.3094408.

[17] Y. Zhang, N. Beheshti et al., "Steering: A software-defined networking for inline service chaining," in Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, 2013, pp. 1–10.

[18] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," Mij, vol. 101, p. 1.

[19] L.-E. Liane, S. N. Joseph, C. Rami, and R. Danny, "Near optimal placement of virtual network functions," in Proc. IEEE INFOCOM, 2015.

[20] A. Gember et al., "Stratos: A network-aware orchestration layer for virtual middleboxes in clouds," arXiv preprint arXiv:1305.0209, 2013.

[21] Google Optimization Tools (a.k.a., OR-Tools) is an open-source, fast and portable software suite for solving combinatorial optimization problems. [Online]. Available: https://github.com/google/or-tools#readme, https://developers.google.com/optimization

[22] Correlation between latency and hop count, Website, [Online], Available: https://www.researchgate.net/publication/267511260_CORRELATION_BETWEEN_LATENCY_AND_HOP_COUNT

[23] Primer on the latency and bandwidth, Website, [Online], Available: https://hpbn.co/primer-on-latency-and-bandwidth/

[24] OpenStack QoS, Website, [Online], Available: https://docs.openstack.org/neutron/latest/admin/config-qos.html