IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received December 2, 2018, accepted December 18, 2018, date of publication January 7, 2019, date of current version January 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2889943

Software Defined Network-Based Edge Cloud Resource Allocation Framework

FAISAL A. ZAMAN[1], ABDALLAH JARRAY[2], (Member, IEEE), AND AHMED KARMOUCH[3], (Member, IEEE)
[1]University of Ottawa, Ottawa, ON, Canada
[2]Electrical and Computer Engineering Department, University of Ottawa, Ottawa, ON, Canada
[3]SITE, University of Ottawa, Ottawa, ON, Canada

Corresponding author: Faisal A. Zaman (fzama075@uottawa.ca)

This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT A growing number of user applications rely on the computing resources of edge cloud data centers (ECDCs). The efficient management of ECDC resources has a positive impact on the cloud service providers' profitability and on the stringent quality-of-service (QoS) requirements of user applications. This paper proposes a resource allocation framework of interconnected ECDCs using software-defined networking (SDN). SDN technology is used to ensure QoS and to efficiently embed user applications into the ECDCs. The proposed framework, called infrastructure as a service provisioning using SDN (IaaSP-SDN), includes the two-phase coordinated IaaS requests. Provisioning approach and a set of SDN management modules to set up accepted IaaS. The performance of IaaSP-SDN is evaluated in two steps: 1) a prototype is compared with generalized multi-protocol label switching (GMPLS) and 2) the impact of SDN controller physical location on the performance of the IaaSP-SDN framework is evaluated. The results illustrate that the SDN framework is more stable and scalable than GMPLS. We also show that SDN controller location attributes have a significant effect on the acceptance ratio of IaaS requests.

INDEX TERMS Controller placement, edge data-centers, IaaS provisioning, metro network, software defined networking.

I. INTRODUCTION

From its inception, Software Defined Networking (SDN) rapidly captured the attention of Cloud Service Providers (CSPs). SDN enhanced the use of network resources by automating their orchestration and providing a global view. However, the automation could not fully achieved for two major reasons. First, because of the inflexibility of some legacy networking equipment [1] and second, because the QoS requirements of IaaS requests may not be met by traditional Cloud Data-Center (CDC) architecture.

Traditionally, CDCs are located far from end-users and are geographically distributed in regions with lower deployment and operational costs [2]. In addition, CDCs were designed for non-real-time applications and may not efficiently satisfy the QoS requirements of user applications. To overcome these limitations, a new class of architecture, Edge Cloud Data-Centers (ECDCs) [3], was introduced. ECDCs are kept in the last mile, closer to the end users. ECDCs not only provide a better Quality of Experience (QoE), but they also reduce the deployment costs. ECDCs provide opportunities for a broad range of new, over-the-top (OTT) user applications such as Netflix video streaming, Map Reduce applications, Facebook, Skype and Twitter [3], [4]. The distributed nature of ECDCs helps the CSPs and application providers to generate more revenue by using network virtualization, since this can improve the performance of the user application by placing needed data in the closest ECDC.

Network virtualization has been regarded as a promising solution for the Internet ossification that results from legacy networking equipment and complex routing protocols. A well-known example of a complex protocol is packet switching, the most commonly used technology for communication, that could therefore help to automate resource allocation. However, the cost and complexity involved would be inadequate to meet the rising demand for user applications. Packet switching makes it difficult for CSPs to guarantee QoS for user applications. Consequently, CSPs have adopted the Metro Optical Network (MON) for sensitive delay/bandwidth applications. In addition, the maturity of the Dense Wavelength Division Multiplexing technology (DWDM) and new optical switching equipment such as Reconfigurable Optical Add Drop Multiplexers (ROADMs) has provided MONs with more flexibility. Using complex algorithms, MON's resource allocation can now operate autonomously [8].

The efficient use of interconnected ECDCs involves two main tasks: (a) allocation of ECDC resources, and (b) simplifying the management of the MON by moving the control logic from the switches to a centralized controller. Separating the control plane and the data plane, as proposed in the SDN paradigm [10], simplified the development of new services and virtualization of the ECDC resources for CSPs.

The allocation of network and ECDC resources, known as the Cloud resource provisioning problem, is well investigated in the literature [12]–[19]. In modeling the Infrastructure as a Service (IaaS) provisioning problem, the resource requirements of a given IaaS are represented as a Virtual Network (VN), made up of a set of nodes and a set of links. An IaaS provisioning solution calculates and maps the nodes and links into the ECDC infrastructure. However, most proposals in the literature involve traditional networking equipment which do not adapt well in case of a variation in QoS of user applications requirement or in the case of networking failure events. Indeed, in traditional network architecture data and control planes are coupled in the layer 2 and 3 equipment which result in high adaptation / reconfiguration time (i.e., time to adapt to any changes in the networking and computing resources used for the provisioning of IaaS requests).

In our previous work [30], we proposed an approach that provisions IaaS to interconnected ECDCs using an SDN-based MON. This preliminary work confirms the positive impact of using SDN architecture on the performance of the IaaS provisioning approach. In this paper, therefore, we want to build on our preliminary results and answer another question: Does the SDN controller's location plays any role in the performance of the proposed provisioning approach? [20] implemented a primary test-bed to evaluate the impacts of the SDN controller location on minimizing control plane (SDN controller) to data plane (Switches) latency. The study confirmed SDN location impact on the latency between data-plane and control plane. Further suggested that a single controller is sufficient to satisfy the latency requirement in a small network.

Most proposals in the literature use the minimization of latency to determine the optimal SDN controller location [21]–[29]. However, in MON, it is not only the link latency that matters but also the response time associated with each switch. The constant change in traffic patterns with web 2.0 applications makes a controller's response time vital and can affect its performance, as can resource availability. These are the factors that determine the optimal controller location within a MON topology; failing to consider them can lead to unusual delays or high response times.

The contributions of this paper can therefore be summarized as follows:

- We propose a SDN-based IaaS Provisioning framework (IaaSP-SDN) that is based on coordinated node and link Provisioning of IaaSs into the ECDC infrastructure and a set of SDN management modules to set up the accepted IaaS requests.
- We design a provisioning framework according to SDN principles of separating network data and control plans. By doing so, we ensure that node and link provisioning phases are better coordinated.
- We compare IaaSP-SDN's performance to the GMPLS approach, the current state-of-the-art control plane for MONs.
- We propose different strategies for determining the optimal SDN controller location within the MON. IaaSP-SDN performances are evaluated accordingly.

The rest of the paper is organized as follows: Section II discusses related work. Section III presents the networked ECDCs infrastructure. Section IV presents the proposed framework for IaaS Provisioning using SDN (IaaSP-SDN). Section V presents the approach used for finding the optimal location of the SDN controller within a given MON topology. In Section VI, the performances of the IaaSP-SDN with different controller location selection strategies are analyzed and evaluated with respect to the GMPLS framework. Section VII concludes the paper and suggests future work.

II. RELATED WORK

This section presents an overview of related works in the literature dealing with SDN-based IaaS Provisioning on ECDCs infrastructure and those addressing the SDN controller location problem.

A. IaaS PROVISIONING

The simplicity of SDN-managed networks has captured the attention of the research community, including the integration of IaaS and Cloud Data-Center resources into the SDN framework.

Gu *et al.* [12] introduce an IaaS provisioning model using a reactive approach, where the IaaS resource allocation is triggered during optical network failures.

Jiachen *et al.* [13] use a node/link failure probability factor in their IaaS provisioning model to decide on the locations of virtual links and nodes. This work uses the SDN nature of the optical network to enhance its resilience and to make an informed decision on the provisioning of accepted IaaSs. However, the proposal overlooks the QoS requirements of user applications and the SDN controller's impact on the proposed solution.

References [14] and [15] present dynamic provisioning approaches based on switch load, memory and storage. Reference [15] emphasizes that number of flows through a location is an important consideration for a controller location. However, the proposals in both [14] and [15] focus only on the attributes of the data plane (switching layer).

A framework for IaaS provisioning of a converged fiber and wireless network is proposed in [16]. But, with no separation between control and data plan, routing decisions are made at switch level. The approach accepts IaaS requests if the computational requirement is met, while a heuristic approach reallocates channels if link provisioning is not satisfactory.

Wen *et al.* [17] consider the computational attributes of IaaS requests along with link and switch requirements. The key objective of their approach is to increase the acceptance ratio and the CSP's revenue. Less focus was given to the stringent QoS requirements of IaaS requests.

Reference [18] introduces a Mixed Integer linear programming model that allows for the partial provisioning of IaaS requests. QoS requirements were considered only to increase the CSP's profit.

Reference [19] proposes an ILP-based approach for IaaS resource provisioning. The resiliency of the accepted IaaS requests was tested against different factors that can affect resiliency. The topology consists of large, geo-distributed DCs with no limit on computing resources.

B. SDN CONTROLLER LOCATION

Does the SDN controller's location have any role in the performance of the IaaS resource provisioning approach? Most approaches in the literature have focused simply on the optimal location of the controller, with little or no focus on the impact of that location. Most relevant proposals are presented below.

Brandon *et al.* [20] and Zhang *et al.* [21] attempt to highlight the importance of the SDN controller location in a network. The former uses the link latency approach to determine the optimal location, while the latter uses a reliability-aware location by splitting the substrate network. The split architecture approach uses the smallest number of SDN controllers needed to maintain the reliability of the data plane in a given topology. No analysis was provided on the impact of the selected locations on the allocation of network resources.

References [22]–[24] propose a dynamic location for the SDN controllers in a large network. The location is decided based on factors such as the current IaaS demand and the load on the controllers. Again, however, no analysis was provided on the impact of the controller location.

References [25] and [26] propose optimization techniques for the SDN controller location to improve the QoS of the control plane. Less focus was given to the management of network resources with respect to the location.

Using different graph theory concepts, Vizarreta *et al.* [27] and Guo *et al.* [28] introduce a robust and resilient location for SDN controllers.

Reference [29] present a heuristic approach for the controller placement in a large-scale core network using the warehouse location optimization technique. None of these proposals analyzes the impact of the location.

Two main observations can be made about these works: *(a)* the focus is more on the data plane use in general and less on the constraints associated with the type of networking layer used, and *(b)* little or no focus is given to the impact of the SDN controller location. A summary of the related work compared to the proposed work is presented in table 1

TABLE 1. Summary of related works.

Related Works	Proposed Solution
[12], [13], [17], [18] proposes resource allocation model for IaaS but overlook the QoS requirements.	IaaS's Network as well as computation QoS are taken into consideration while modeling
[14], [15] considered SDN-based setup and focus on optimizing based on data-plane resources.	we considered both control plane as well as data-plane resources for modeling and evaluation.
[22], [23], [24] proposed optimization of controller location but did not focus on the performance of services on the selected location	In the proposed controller study, we investigate the influence of controller location on the IaaS performance.
[27], [28] [29] use heuristics and graph theory approaches to determine controller location but doesn't investigate the impact of controller location on the services in a network	We investigate the performance of IaaS services for different controller location using emulated network traffic.

III. SDN-BASED NETWORKED EDGE CLOUD DATA-CENTERS

Networked Edge Cloud Data-Centers allow CSPs to reduce their capital costs and to benefit from the elasticity of the Cloud architecture by placing VMs running user processing tasks closer to the end-users. The Edge Cloud infrastructure uses smaller DCs located in the last mile, closer to major population centers, in order to honor the QoS requirements of user applications. The main elements of the proposed resource provisioning framework using an ECDC topology and an SDN-based Metro Network are presented below.

A. EDGE CLOUD DATA-CENTERS

Today, 55% of the Internet traffic is from multimedia applications and is predicted to reach 92% by the end of 2020 [35]. These applications require a large amount of computing resources and have stringent processing requirements and constraints. Failing to consider these requirements can lead to poor performance and inefficient network use [36]. Networked Edge Cloud Data-Centers were introduced to

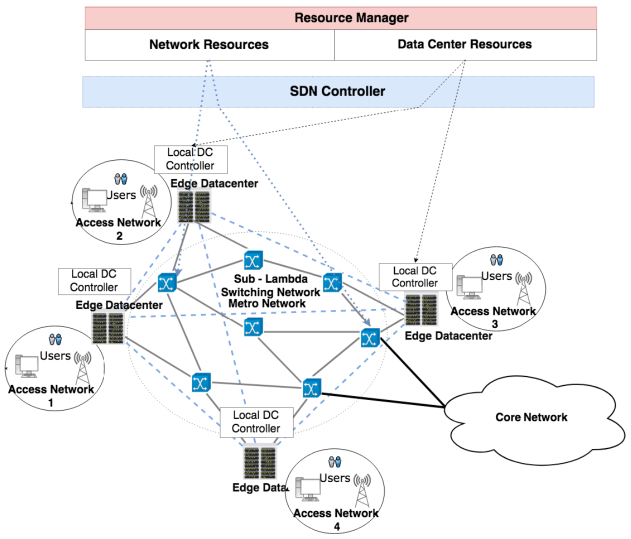

FIGURE 1. High-level view of SDN-enabled ECDC network and network services.

address these requirements. Fig.1 depicts the high-level view of SDN-enabled ECDC network and Network Services. As described in the following section, IaaS are composed of set of nodes and a set of interconnecting links. The IaaSs are deployed (embedded) over the ECDC infrastructure using an SDN-based MON to provide greater flexibility and more efficient resource allocation.

B. SDN-BASED METRO NETWORK (MON)

Traditionally, MONs have been used to integrate various access networks into Wide Area Network (WAN). With the introduction of multimedia services and web 2.0, MONs need to become more agile and flexible [7].

Network Management System (NMS) and GMPLS are among the best-known networking management architectures. NMS is based on a static, manual configuration, with human intervention between the control plane and data plane. This rigidity makes NMS-based MONs unsuitable for multimedia services. To automate the system, GMPLS was introduced. GMPLS is not simply an extension of the MPLS protocol; it also acts as a control plane to the network and allows dynamic configuration. The core protocols of GMPLS are Link Management Protocol (LMP), Reservation Protocol-TE and OSPF-TE. While these allow increased complexity, GMPLS is used only for controlling and routing devices; it cannot handle tasks such as the orchestration of ECDC resources, and coordination with the QoS requirements of applications. The complications and sophistication involved with the number of protocol interactions led to the downfall of the GMPLS paradigm [34]. In contrast, a SDN-enabled network can virtualize the data plane and run many user-defined applications with different QoS requirements. Figure 2 gives an overview of different management and coordination done by the SDN controller. The centralized nature of SDN made it easy to integrate new technologies, to improve flexibility, and to automate the orchestration of network resources. Accordingly, an SDN-based MON was selected to interconnect and manage the Edge Cloud infrastructure. The proposed network view and the centralized control plane are shown in Fig. 3.

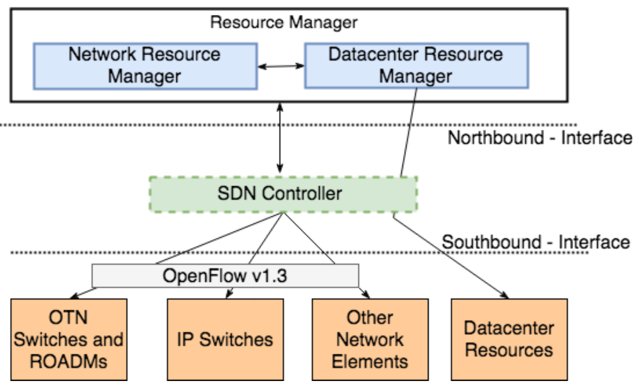

FIGURE 2. SDN controller duties and positioning.

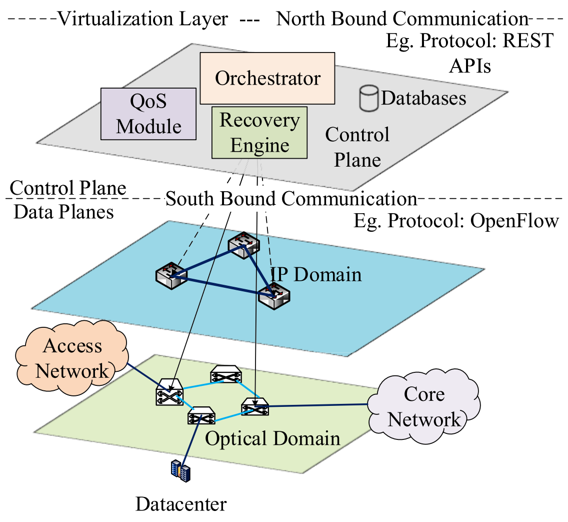

FIGURE 3. Multi-Layer Network View in an SDN-based control system.

IV. SDN-BASED EDGE CLOUD COMPUTING RESOURCE ALLOCATION FRAMEWORK

Thyagaturu *et al.* [50] states that with the increase in SDN paradigm prominence, Orchestration functionality is becoming more integrated as a part of SDN controller. Resource allocation (RA) and management is one of the main function of ECDC management and Orchestration. RA is directly related to the cost and the QoS requirements of accepted IaaS requests. Efficient RA has a positive impact on the CSPs' profitability. The RA problem is in minimizing ECDC hosting and MON networking costs while preserving QoS requirements. The following section describes the mathematical modeling of the RA approach.

A. SDN-BASED ECDC INFRASTRUCTURE MODELLING

The SDN-networked ECDC infrastructure is represented by an undirected graph $G_s = (W_s, L_s)$, where W_s is the set of ECDC nodes and L_s is the set of Optical Fiber links that interconnect ECDCs' physical locations. Each ECDC node $u \in W_s$ has a Unit Processing CPU capacity P_u, a memory capacity R_u, a Graphic Processing capacity GPU G_u, and a storage capacity S_u. Similarly, optical fiber links $l \in L_s$ offer W wavelengths, each with bandwidth capacity B_l. We introduce a bandwidth unit cost c_l^b per substrate link $l \in L_s$, for load balancing purposes. Similarly, we associate for each ECDC node $u \in W_s$: a CPU unit cost c_u^p, a memory unit cost c_u^r, a GPU unit cost c_u^g, and a storage unit cost c_u^o. The upper part of Table 2 shows the interconnected ECDC infrastructure parameters.

B. IaaS REQUEST MODELLING

An IaaS request from ECDC resources is represented as Virtual Network $n \in N$, which is represented by a directed graph $G_n(A_n, E_n)$. A_n represents the set of nodes that requires computing, memory, graphic processing and storage resources. The set of virtual links E_n represents the networking requirements between virtual nodes.

TABLE 2. Notation of IaaS provisioning problem.

Parameters	Description
	Networked ECDCs: $G_s(W_s, L_s)$
W_s	Set of ECDC nodes.
L_s	Set of Optical Fiber links.
t	Time period.
$P_u(t)$	Available CPU in node u at period t.
$R_u(t)$	Available memory in node u at period t.
$G_u(t)$	Available GPU in node u at period t.
$O_u(t)$	Available storage in node u at period t.
$B_l(t)$	Capacity of a wavelength of optical link l at period t.
$N_r(u)$	Number of ROADMs in node u.
N_π	Number of hopes in a light-path π.
c^b_l	Bandwidth cost unit of optical link l.
c^p_u	CPU cost unit of ECDC node u.
c^r_u	Memory cost unit of ECDC node u.
c^g_u	GPU cost unit of ECDC node u.
c^o_u	Storage cost unit of ECDC node u.
c^d_u	Cost of using ROADM.
Π^e_{uv}	Set of all shortest light-paths between nodes (u, v) assigned to link e.
Π^s	set of all shortest light-paths in G_s.
W	Number of available wavelengths.
	IaaS: $G_n(A_n, E_n)$.
N	Set of virtual network requests.
A_n	Set of nodes of virtual network n.
E_n	Set of links of virtual network n.
s and d	Source and destination nodes.
p_a	Required CPU for node a.
r_a	Required memory for node a.
g_a	Required GPU for node a.
o_a	Required storage for node a.
b_e	Required bandwidth for link e.
d_e	Maximum number of hops for link e.
P^n_e	Revenue from provisioning link e of virtual network n.

The QoS requirements of each virtual node $a \in A_n$ is defined by a set of Virtual Machines (VMs) that require: (a) CPU capacity p_a, (b) memory capacity r_a, (c) GPU capacity g_a, and (d) storage capacity o_a. Similarly, each virtual link $e \in E_n$ has a set of QoS requirements: (a) a minimum required bandwidth b_e, and (b) a maximum number of hops (optical links) d_e. The bottom part of Table 2 summarizes IaaS parameter.

C. IaaS PROVISIONING

The resource provisioning of each IaaS request can be divided into node and link provisioning, as follows.

1) NODE PROVISIONING

Each virtual node $a \in A_n$ from the same IaaS request n is embedded to a different substrate node $u \in W_s$ by provisioning: $M_\mathrm{n} : A_n \mapsto W_s$.

2) LINK PROVISIONING

Similarly, each virtual link $e \in E_n$ from the same IaaS request n is embedded to a different substrate path $\pi \in \Pi^e_{uv} \subset \Pi^s$ by provisioning: $M_\mathrm{l} : E_n \mapsto \Pi^s$, where (u, v) are substrate nodes assigned to virtual nodes (s, d), the source and destination nodes of virtual link e.

D. CLOUD SERVICE PROVIDER OBJECTIVE FUNCTION

When an IaaS request arrives, the CSP has to determine whether to accept or reject it. The main criteria for this decision are the availability of ECDC resources and the economic benefit of accepting the request. We calculate the CSP's revenue of each request n as follows.

$$\mathrm{rev}(G_n) = P_n - \mathrm{cost}\left[M_\mathrm{n}(A_n), M_\mathrm{l}(E_n)\right] \tag{1}$$

where the first term of Equation (1) calculates the revenue collected from provisioning IaaS request n and the second term calculates the cost of assigned ECDC resources to IaaS request n.

E. FRAMEWORK OF IaaS PROVISIONING ARCHITECTURE

The proposed resource allocation framework is shown in Fig. 4. Since the architecture uses a SDN-enabled Metro Network for Provisioning IaaS requests, it is referred to as IaaSP-SDN. The following paragraphs describe how the major components interact.

- *Request Handler and Processed Request Queue:* This component manages each IaaS request and generates an

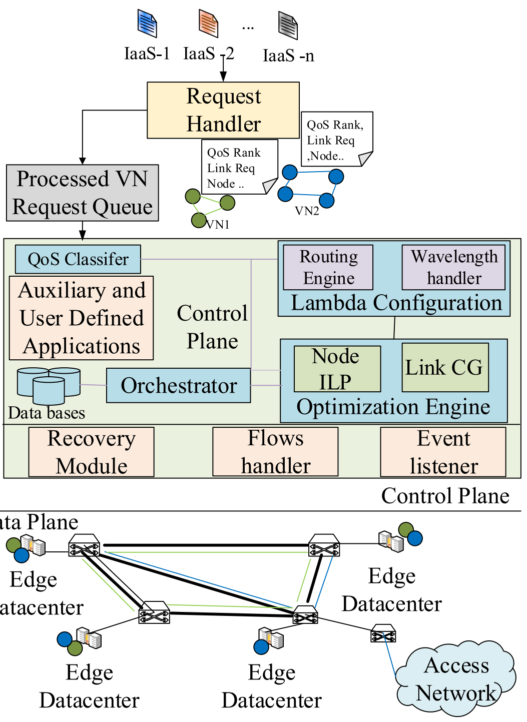

FIGURE 4. Framework architecture of SDN-Networked edge cloud data-centers.

Information Graph (IG) that provides detailed information about the request. It consists of the ECDC resources and the link requirements needed for the requests.

- *IaaS Requests Queue:* This is a buffer where incoming IaaS requests are kept on hold until the active provisioning process is completed.
- *SDN Orchestrator/ Modules Coordinator:* With the increase in the complexity of SDN controller, Orchestration has become a major function of SDN controller. Typically in SDN framework an Orchestrator is a module which co-ordinates among other software blocks or functions or modules to manage resources and to provision efficiently the incoming IaaS requests to the underlying physical network [50]. A more detailed figure on the role of Orchestrator can be found in [50, Fig. 2]. The Orchestrator converts the accumulated IGs into a set of flow rules and messages that inform the ECDC to reserve the cloud computing resources needed. Once the request arrives, the Orchestrator communicates it to the *Routing Engine*.
  Orchestrator also communicates the final set of lightpaths and ECDC nodes of each accepted IaaS request to the *Flow Handler* which is responsible for communicating directly with the SDN controller. Once the accepted IaaS requests are embedded, the SDN controller has to keep track of these resources to make sure that the QoS is not affected while an accepted IaaS is still active.
- *SDN Routing Engine:* This identifies the QoS class through a *QoS Classifier* and communicates with the *Databases* for available resources. Once the Routing Engine makes the decision, it responds to the Orchestrator with the set of routing paths. The Orchestrator gives raw inputs consisting of the available resources in ECDC nodes and the available routing paths to the *optimizer*.
- *Optimizer:* This processes the incoming IaaS requests, sends the information to the Orchestrator and prompts the CSP to update network usage. The Optimizer consists of two engines: the *Node Provisioning Engine* and the *Link Provisioning Engine*, both controlled by the Orchestrator. When an IaaS request arrives, the Orchestrator communicates with the Node Provisioning Engine to find the optimal location for the IaaS nodes or virtual Machines. It also provides the set of potential routing paths for each request. The information is then sent to the Link Provisioning Engine, which finds the most suitable routing paths. It evaluates the routing paths provided by the Node Provisioning Engine and provides the optimal routing solution. Node and link optimization are designed as separate modules. This provides the flexibility to run them independently, such as to regroom optical traffic during a network failure. The Link Provisioning Engine is executed whenever congestion is sensed in any network link. This is of particular importance in MON because of frequent sudden shifts in the traffic matrix, requiring traffic to be diverted to alternate paths without losing QoS.
- *Event Listener and Recovery modules:* These two modules are responsible for maintaining resiliency in the network and for recovery. We keep the discussion of these modules to a minimum as the details are beyond the scope of this work. However, it must be mentioned that the architecture includes segmented routing concepts available in the ONOS controller, as proposed in [12].

F. MATHEMATICAL MODELLING OF IaaS PROVISIONING PROBLEM

To evaluate the merits of the proposed framework, we propose a mathematical modeling that performs IaaS Provisioning in coordinated fashion [52]. The first phase determines the best set of ECDC nodes that (a) satisfies the computing and QoS requirements of the IaaS requests, and (b) finds some of the many possible routing paths between selected ECDCs. The solution determined by the first phase is coordinated by the solution determined in the second phase. The second phase selects one from the predetermined routing paths with the minimum cost.

Advantages of Two Coordinated Phase Approach: The Global view provided by SDN controller can be utilized to constantly restructure the traffic pattern to better utilize the existing capacity. Running IaaS optimizer for the whole set of network data for every period can be computationally intensive. Especially moving the IaaS locations from one Data-Center to another. Hence having two separate phases where in the global optimal solution is coordinated by the link solution can be leveraged. In this way, we can only run the Link optimizer phase more frequently. By doing so helps reduce the computational strain on the SDN controller's CPU. Mathematical modeling of node and link provisioning is explained in the following sections.

1) NODE PROVISIONING

Using our previous work [46], we formulate the node Provisioning problem as an Integer Linear Program. We call this model (Node-ILP). To decide on the ECDCs selected to handle the virtual nodes of each IaaS request $n \in N$, we define the following binary decision variables: (a) $z_n = 1$, if the IaaS request n is accepted and 0 otherwise, and (b) $x_a^u = 1$, if virtual node $a \in A_n$ is assigned to ECDC of node $u \in W_s$ and 0 otherwise.

As mentioned, CSP accepts an IaaS request $n \in N$ only if it increases his revenue while satisfying QoS requirements. For this, we used the following objective function.

A-Objective Function:

$$\sum_{n\in N} z_n \Bigg\{ \sum_{e=(s,d)\in E_n} P_e^n - \sum_{(u,v)\in W_s \cdot W_s} x_s^u \cdot x_d^v \cdot (C_u + C_v + \sum_{l\in \Pi_{uv}^e} c_l^b \cdot b_e) \Bigg\} \quad (2)$$

where, P_e^n is the revenue generated by accepting a virtual link e from an IaaS request n. In Eq: 2, $\sum_{l\in\Pi_{uv}^e} c_l^b \cdot b_e$ is used to coordinating with the next provisioning phase. C_u and C_v are the costs of ECDC resources used respectively in node u and v and calculated as follows.

$$C_u = c_u^p \times p_s + c_u^r \times r_s + c_u^o \times o_s + c_u^g \times g_s. \quad (3)$$

and

$$C_v = c_v^p \times p_d + c_v^r \times r_d + c_v^o \times o_d + c_v^g \times g_d. \quad (4)$$

We denote by Π_{uv}^e, the set of shortest paths between the ECDCs of nodes u and v assigned respectively to source node s and destination node d of virtual link, i.e., $e = (sd)$.

The resources allocated to any IaaS request $n \in N$ from ECDCs are governed by the following constraints:

B-Constraints:

1) All virtual nodes of an accepted IaaS request $n \in N$ should be embedded into the infrastructure, which is given by

$$z_n \leqslant \sum_{(u,v)\in W_s \cdot W_s} x_s^u \cdot x_d^v; \quad e = (sd) \in E_n. \quad (5)$$

2) We denote by W_a the subset of W_s, which represents the potential ECDC locations that can satisfy the QoS requirements of node $a \in A_n$. At any given period of time t, QoS requirements to be satisfied for a given virtual node $a \in A_n, n \in N$ are governed by the Eqs. (6) - (9).

   a) Available CPU capacity on a selected ECDC node u should be more than the requested for node a in a period t.

$$\sum_{n\in N}\sum_{a\in A_n} x_a^u \cdot p_a \leqslant P_u(t); \quad u \in W_s \quad (6)$$

   b) Available memory on a selected ECDC node u should be more than the requested for node a in a period t.

$$\sum_{n\in N}\sum_{a\in A_n} x_a^u \cdot r_a \leqslant R_u(t); \quad u \in W_s \quad (7)$$

   c) Available GPU capacity on a selected ECDC node u should be more than the requested for the virtual node a in a period t.

$$\sum_{n\in N}\sum_{a\in A_n} x_a^u \cdot g_a \leqslant G_u(t); \quad u \in W_s \quad (8)$$

   d) Available storage capacity on a selected ECDC node u should be more than the requested for node a in a period t.

$$\sum_{n\in N}\sum_{a\in A_n} x_a^u \cdot o_a \leqslant O_u(t); \quad u \in W_s \quad (9)$$

2) LINEARIZATION OF QUADRATIC TERMS

The constraint (5) and objective function (2) include quadratic terms $x_s^u x_d^v$. However, since this quadratic term is the product of two binary variables, it can be linearized easily by replacing the quadratic term by a new binary variable $y_{s,d}^{u,v}$ where $y_{s,d}^{u,v} = x_s^s x_d^v$ and by adding the following constraints.

$$y_{s,d}^{u,v} \le x_s^u \quad (10)$$
$$y_{s,d}^{u,v} \le x_v^d \quad (11)$$

Inequalities (10) and (11) ensure that $y_{s,d}^{u,v}$ will be zero if either x_s^u or x_d^v are zero.

$$y_{s,d}^{u,v} \ge x_d^v + x_s^u - 1 \quad (12)$$

Inequality (12) ensure that $y_{s,d}^{u,v}$ will be 1 if both binary variables x_s^u or x_v^d are set to 1. This linearization technique is done in our simulation by the CPLEX linear solver [49].

3) COMPLEXITY ANALYSIS

The proposed model Node-ILP can be solved in polynomial time. Indeed, as, for a given IaaS request $G_n(A_n, E_n)$, a virtual node $a \in A_n$ can be assigned to: $\mid W_s \mid$. Then an IaaS request of a maximum $\mid W_s \mid \times \mid W_s \mid$ nodes can be assigned to $\mid W_s \mid \times \mid W_s \mid$ possible provisioning node solutions that can be approximated by the polynomial number: $\simeq o(c^2)$. We used the CPLEX solver to solve the Node-ILP model.

G. LINK PROVISIONING

The link provisioning stage aims to select a set of routing paths to connect the ECDCs selected in the node provisioning stage at minimum cost and meeting the QoS requirements of the virtual links. A trivial formulation could be to use an ILP, as we did in the node provisioning stage. However, a formulation based on an Integer Linear Programming model may suffer from scalability issues [5]. With a large number of IaaS requests, an ILP model takes on a large number of variables and constraints. This is potentially a significant drawback to solving the ILP model optimally in a reasonable time, as shown in the following section.

1) DRAWBACKS OF ILP FORMULATION

i. As, for a given IaaS request $G_n(A_n, E_n)$, a virtual link $e \in E_n$ can be assigned to: $\mid W_s \mid \times \mid W_s \mid \times \mid \Pi_{uv}^e \mid$ possible provisioning light-path solutions.

ii. Thus, $\mid E_n \mid$ virtual links of N IaaS requests can be assigned to: $\left(\mid W_s \mid \times \mid W_s \mid \times \mid \Pi_{uv}^e \mid\right)^N$ possible provisioning solutions, which can be approximated by the exponential number: $\simeq o(c^N)$.

As mentioned previously, link provisioning is a NP-hard problem, equivalent to a multi-way separator problem [6]. To address this complexity, we propose in following a large scale approach based on the Column Generation technique [5].

2) COLUMN GENERATION FORMULATION

Using the Column Generation (CG) technique means reformulating the link provisioning problem in terms of Independent provisioning Configurations (IECs) [31].

An IEC configuration defines the provisioning solution of at least one IaaS request. It is represented by the set of ECDC nodes used to handle the resource requirements (CPU, memory, GPU and storage) and all the links/light-paths with the same wavelength that connect these nodes. An IEC indexed by c defines a provisioning solution of one or more IaaS requests. We call the set of all possible IECs C. In order to reduce the number of configurations, we restrict ourselves to maximal IECs. An IEC is maximal only if we cannot increase the number of IaaS requests served without increasing the cost of resources. An IEC configuration $c \in C$ is defined by the vector $(a_n^c)_{n \in N}$ such that: $a_n^c = 1$, if the IEC c serves IaaS request n and 0 otherwise. We also define the decision variable $\lambda_c = 1$, if the IEC c is used in provisioning solution and 0 otherwise.

The link provisioning problem can be formulated with respect to the variables (λ_c), $c \in C$. Under the new formulation, the problem then chooses a maximum of W IECs, where W is the number of available wavelengths per optical fiber link $l \in L_s$. The resulting configuration corresponds to the so-called master problem in the Column Generation technique [31], while each configuration design corresponds to the so-called pricing problem. We denote by cost$_c$ the cost of an IEC c. It corresponds to the costs of link bandwidth and OEO conversion ROADM ports (used to groom IaaSs bandwidth requirement on the same wavelength) used by IEC c. It is defined according to Equation (1) as follows:

$$Cost_c = \sum_{u \in W_s} M_u^c \cdot c_u^d + \sum_{l \in L_s} B_l^c \cdot c_l^b \tag{13}$$

where, c_u^d is the unit cost of using a ROADM in ECDC node $u \in W_s$ and c_l^b is the bandwidth unit cost of link $l \in L_s$. M_u^c and B_l^c are respectively the total number of ROADMs used in ECDC node $u \in W_s$ and the total bandwidth used in optical link $l \in L_s$ by IEC configuration $c \in C$.

The Column Generation technique [5] means that the link provisioning problem is decomposed into a master problem (which includes constraints related to the availability of substrate resources) and a pricing problem (which includes the constraints related to the link provisioning of IaaS requests).

The problem becomes one of generating an additional column to the constraint matrix of the master problem. In other words, generating an IEC that reduces the current value of the master objective function. We call this approach Link Provisioning using CG (Link-CG). The mathematical formulations of the master problem and the pricing problem are provided below.

A-Master Problem: Master problem is formulated as an Integer Linear Program *ILP*(M).

a-Objective Function: The link provisioning objective is to reduce the cost of used light-paths while maintaining IaaS request's virtual links QoS requirements.

$$\min \sum_{c \in C} (Cost_c \cdot \lambda_c) \tag{14}$$

b-Constraints:

$$\sum_{c \in C} \lambda_c \cdot a_n^c \geq 1; \quad u \in W_s;\ n \in N. \quad (\beta_u) \tag{15}$$

$$\sum_{c \in C} \lambda_c \cdot M_u^c \leq N_r(u); \quad u \in W_s \quad (\mu_u) \tag{16}$$

$$\sum_{c \in C} \lambda_c \leq W \quad (\alpha_0) \tag{17}$$

$$\lambda_c \in \{0, 1\}; \quad c \in C \tag{18}$$

Equation (15) ensures that an IaaS request is embedded to only one IEC configuration. Equation (16) guarantees the respect of available number of ROADMs in any ECDC node. Equation (17) ensures that the number of IEC configurations selected in the link provisioning solution do not exceed the number of available wavelengths W in the optical links, as each IEC will be assigned a distinct wavelength.

B-Pricing Problem: The pricing problem is used in generating additional columns for the constraint matrix of the master problem. It deals with the constraints related to provisioning IaaS virtual links into routing light-paths. Consider β, μ and α as dual variables for the master problem constraints (15), (16) and (17) respectively. Then, the reduced cost of variable λ_c can be written as:

$$\overline{Cost_c} = Cost_c + \alpha_o + \sum_{u \in W_s} \mu_u \cdot M_u^c - \sum_{n \in N} a_n^c \cdot \beta_u \tag{19}$$

In order to linearize the expression of the reduced cost and to express the constraints of the pricing problem, we define the binary decision variables z_n, y_u and x_e^π, where $z_n = 1$, if IaaS request n is served by the IEC configuration c and 0 otherwise. Similarly, $y_u = 1$, if a ROADM is installed in the node $u \in W_s$ and 0 otherwise. $x_e^\pi = 1$, if virtual link $e \in E_n$ is assigned to the light-path $\pi \in \Pi_{u*v*}^e$ and 0 otherwise. Where $u*$ and $v*$ are the selected ECDC nodes in the node provisioning phase for source node s and destination node d of a virtual link $e \in E_n$.

Then, we derive the following relations between the variables of the pricing problem and the coefficients of the master problem.

$$a_n^c = \sum_{n \in N} z_n; \quad n \in N,\ c \in C. \tag{20}$$

$$M_u^c = 2y_u, \quad u \in W_s,\ c \in C. \tag{21}$$

$$B_l^c = \sum_{n \in N} \sum_{e \in E_n} \sum_{\pi \in \Pi_{u*v*}^e} \delta_l^\pi \cdot b_e \quad l \in L_s,\ c \in C. \tag{22}$$

The pricing objective function and constraints can then be expressed as follows.

a-Objective Function:

$$\overline{cost} = \sum_{u \in W_s} 2 \cdot y_u \cdot c_r + \sum_{l \in L_s} c_l \cdot \sum_{n \in N} \sum_{e \in E_n} \sum_{\pi \in \Pi^e_{u*v*}} \delta^\pi_l \cdot b_e + \alpha_o + \sum_{u \in W_s} \mu_u \cdot 2 \cdot y_u - \sum_{n \in N} z_n \cdot \beta_u \qquad (23)$$

b-Constraints:

1) All virtual links $e \in E_n$ of an IaaS request $n \in N$ have to be accepted, otherwise the request will be rejected. This is defined as:

$$z_n = \sum_{\pi \in \Pi^e_{u*,v*}} x^\pi_e; \quad e \in E_n; \ n \in N \qquad (24)$$

Equation (24) means that only one light-path is selected for each virtual link to guarantee the stringent QoS requirements of IaaS requests, where sending data on different light-paths may result in additional delay.

2) Sum of bandwidth of light-path π using a given optical link $l \in L_s$ should not exceed the wavelength capacity B_l.

$$\sum_{n \in N} \sum_{e \in E_n} \sum_{\pi \in \Pi^e_{u*v*}} x^\pi_e \cdot \delta^l_\pi \cdot b_e \le B_l; \quad l \in L_s \qquad (25)$$

where parameter $\delta^l_\pi = 1$ if light-path π uses link l and 0 otherwise. Π^e_{u*v*} is the set of light-paths that can be used for provisioning virtual link $e \in E_n$ of an IaaS request n where their lengths do not exceed the maximum allowed number of hops d_e. Candidate light-paths for each virtual link are calculated using a k-shortest path algorithm [47].

3) Grooming factor g is defined as the total amount of bandwidth that can be pushed into a given wavelength. For the sake of simulation, the grooming factor is kept constant. OTU-1 bandwidth capacity is used for each IaaS request and each substrate link has a bandwidth capacity defined by OTU-3 standards. This makes the grooming factor 16. However, the grooming factor can be adjusted using any value g by adding the following equation (26) to the modeling.

$$\sum_{n \in N} z_n \le g \qquad (26)$$

3) SOLVING COLUMN GENERATION FORMULATION

We now discuss how to solve the Link-CG model developed in previous sections and how to obtain an integer link provisioning solution.

- We denote by $LP(M)$ the continuous relaxation of the master problem $ILP(M)$ obtained by exchanging the integrality constraint (18) by $\lambda_c \in [0, 1]$ for any $c \in C$.
- We solve $LP(M)$ using any linear programming solver. Next, we check the optimality of the solution with respect to the original link provisioning problem. To do so, we need to check the existence of a variable λ_c with a negative reduced cost by solving the pricing problem.

If we succeed in finding a new column (IEC) with a negative reduced cost, we add the resulting IEC c to the current set of IECs by adding the corresponding variable λ_c to the master problem. We repeat until no column can be found with a negative reduced cost. This step is described in the following section.

Column Generation Procedure():

1. We initialize $LP(M)$ by a subset of dummy configurations, that is, a set of artificial IECs with a large cost.
2. We solve the linear relaxation $LP(M)$ of the master problem optimally using CPLEX solver and then go to Step 3.
3. We solve the pricing problem optimally as follows.
   - We first solve the pricing problem using a greedy heuristic that we developed based on k-shortest path [47].
   - If, using this heuristic, we succeed in finding a new column with a negative reduced cost, we go to Step 4.
   - Otherwise we solve exactly the pricing problem using the CPLEX solver and then go to Step 4.

4. If a column with a negative reduced cost has been found, we add this column to the current master problem, and we repeat Steps 2 and 3. Otherwise, the master problem is optimally solved.

The optimal solution of $LP(M)$ only provides a lower bound on the optimal integer solution of $ILP(M)$. To derive an integer value once the $LP(M)$ has been optimally solved, we therefore use the Branch & Bound technique described below, starting with the optimal solution of the linear relaxation $LP(M)$ generated using *Column Generation Procedure()*.

Branch & Bound Technique: In order for the process to remain scalable, instead of defining a branch-and-price procedure, we propose to remove the relaxation on variable λ_c and we proceed with a classic branch-and-bound procedure using any linear solver (such as CPLEX) on selected columns (IECs) to solve the new Integer Linear Program.

V. OPTIMIZATION OF SDN CONTROLLER LOCATION IN METRO OPTICAL NETWORK

The second question that we want to answer in this paper is: Did the SDN controller's location play any role in the performance of the proposed IaaS provisioning approach? For the answer, we propose different strategies to select the optimal location of SDN controller, and then evaluate the performance of the Proposed IaaS Provisioning solution with respect to that optimal location. Depending on the network requirement the Optimal location for an SDN controller can be determined. Some of the objective functions used in the current study is discussed in section V-A. The constraints associated with the objective functions are presented in Section V-A.2.

A. STRATEGIES FOR OPTIMAL LOCATION OF SDN CONTROLLER

SDN controller location problem is defined as determining optimal location of controller in a network based on the defined objective and constraints. To do so, we define a binary decision variable $y_d = 1$ if the SDN controller is located in ECDC node d and 0 otherwise. We denote by $W_d \subset W_s$ the set of all possible controller locations.

1) OBJECTIVE FUNCTIONS

The objective functions to be considered for optimizing controller-to-node communication costs are as follows.

(a) *Minimize the distance between SDN controller to switching nodes:* If $D_d^u(a,o)$ gives a geodetic distance between two points d and u on the surface of the earth, a represents the latitude and o represents the longitude. Actual fiber distance can also be used for the purpose of calculation. The distance is calculated as follows.

$$\min \sum_{d \in W_d} \sum_{u \in W_s/\{d\}} y_d \cdot D_d^u(a,o) \tag{27}$$

where,

$$D_d^u(a,o) = R_{earth}\sqrt{(\Delta(a))^2 + (cos(\phi_m) \cdot \Delta(o))^2} \tag{28}$$

where $\Delta(a)$ is a function that calculates the latitude difference between controller location and switching nodes. ϕ_m is the mean latitude of location d and u and R_{earth} is the earth radius.

(b) *Average minimum response time and distance:* The shortest distance cannot usually provide the least latent paths; the location therefore needs to be optimized so that it does not affect the QoS of data plane to control plane.

$$\min \sum_{d \in W_d} y_d \cdot \left(\frac{\sum_{u \in W_s/\{d\}} R_{du}}{N-1} + \sum_{u \in W_s/\{d\}} D_d^u(a,o) \right) \tag{29}$$

where R_{du} is the response time for path $\pi_{du} \in \gamma_{du} \subseteq \Pi_{du}$. γ_{du} are the set of paths that satisfy the QoS of control plane to data plane. $N = |W_s|$ is the total number of networked ECDCs nodes.

(c) *Most reliable and average minimum response time for SDN controller to switching nodes:* The strategy to select the controller location that minimizes its response time is not always optimal and does not guarantee resilience to link or node failures. To circumvent this limitation, a location that can reduce response time and provide resilience during failure is selected as follows.

$$\sum_{d \in W_d} \frac{y_d}{F^d} \cdot \left(\sum_{u \in W_s/\{d\}} L_{max}(\Pi_{du}) \right) \tag{30}$$

where L_{Max} and F^d are the average of maximum response time of the selected paths and average Shared Risk Link Group (SRLG) respectively. The SRLG factor is determined by finding the number of independent paths from the SDN controller to each switching node. An example of the SRLG calculation is shown in Fig.5 for a controller located in node D to all the other nodes i.e., A, B, C and E. For example D to A can have two independent paths: $D \rightarrow C \rightarrow A$ and $D \rightarrow B \rightarrow A$. Given the total number of possible paths between controller and nodes as k, SRLG can be defined as:

$$F^d = \frac{k}{N-1}$$
$$L_{max} = \frac{\sum_{u \in W_s/\{d\}} R_{du}}{N-1}$$

(d) *Switching node with the most-used flow table:* As demonstrated in paper [15], it is important to consider the flow table space of the switch when designing and planning an SDN-enabled network. The switches with full flow tables are more likely to contact the SDN controller in order to update their flow tables. Placing the controller near the switches prominent in the network can be defined as:

$$\max \sum_{d \in W_d} I_d \cdot y_d \tag{31}$$

where I_d is the node weight factor of the switch. This is calculated by adapting the Katz centrality factor [39]. The node weight in the Katz formulation is modified to represent the flow table size. The greater the value of I_c, the higher the number of flows in the switching table node.

(e) *Switching node with the most-used flow table and the average minimum response time:* When selecting a location based on the flow table memory of a switch, a high response time is possible. This is illustrated in Fig. 5 where, since it is an edge switch, location E is selected if only flow table use is considered. This can lead to a sub-optimal location. The solution is to consider the response time of the SDN controller to nodes as follows.

$$\min \sum_{d \in W_d} \frac{\sum_{u \in W_s/\{d\}} y_d \cdot L_{max}(\Pi_{du})}{I_d} \tag{32}$$

2) CONSTRAINTS

1) At least one location has to be selected to host the controller. Thus the cardinality of W_d should be greater than 1, given as:

$$|W_d| >= 1 \tag{33}$$

2) *Response Time:* This is defined as the time taken for an event generated at the data plane (switch) to receive an update from the control plane (SDN controller). This constraint depends on the type of controller used,

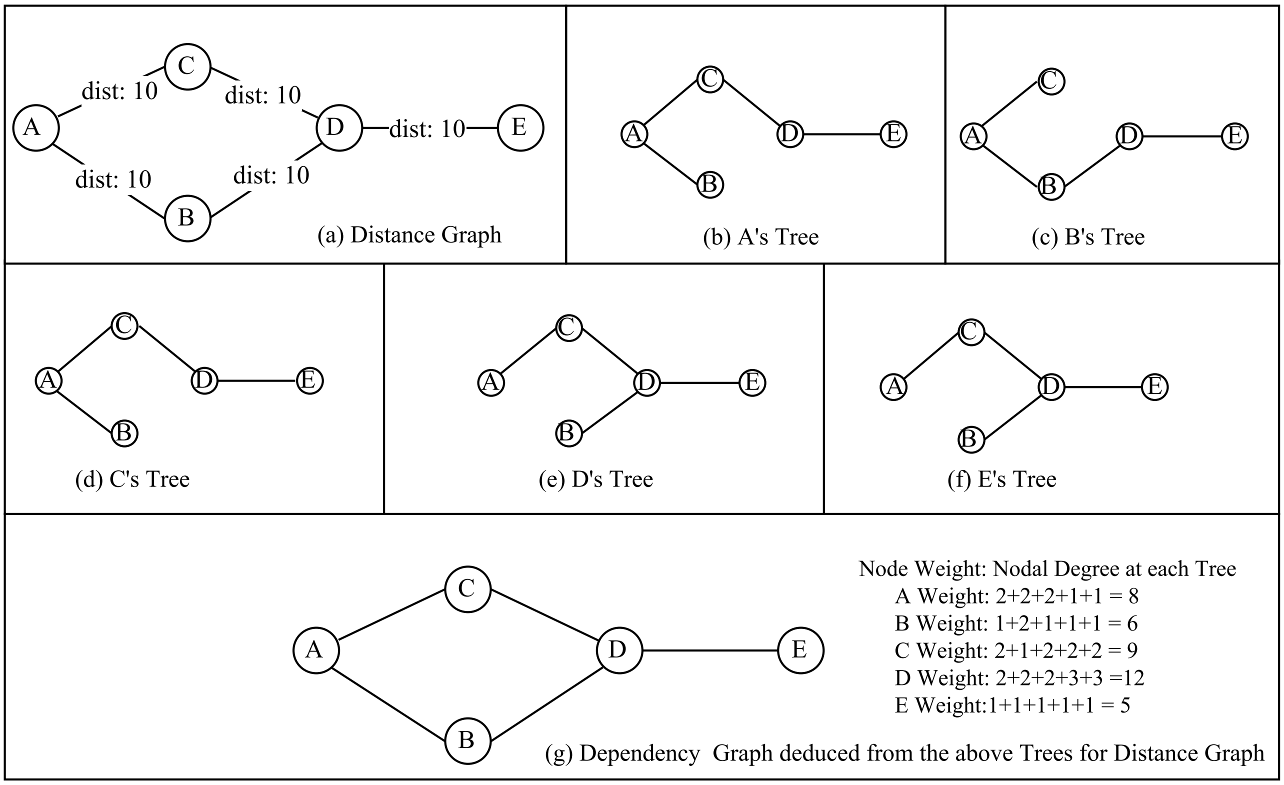

FIGURE 5. **Calculation of performance parameters for a network.**

the link bandwidth, the switching latency and the signal conversion latency.

$$y_d \cdot R_{du} \leq R^u_{max}, \quad \forall u \in W_s/\{d\}, \ \forall d \in W_d \quad (34)$$

where R_{du} is the response time from controller location node d to switching node u using the shortest path. R^u_{max} is the maximum tolerable response time for the switch u.

3) *Maximum Hops:* The hop constraint is vital in MON, because a significant amount of time is taken by signals during the O-E-O conversions required to upload new customer data. This is defined as follows.

$$y_d \cdot \frac{\sum_{\pi \in \Pi_{du}} N(\pi)}{\mid \Pi_{du} \mid} \leq N_{max}(d,u), \quad \forall u \in W_s/\{d\}, \quad \forall d \in W_d \quad (35)$$

where $\mid \Pi_{du} \mid$ gives the number of available paths from SDN controller d to ECDC node u. $N_{max}(d,u)$ is the maximum number of intermediate nodes between the SDN controller node and switch node location u to maintain QoS. $N(\pi)$ is the total number of intermediate nodes in path π.

4) *Contingency:* This is defined as a failure or delay in communication between SDN controller and switch, which can occur due to the shortage of network resources. Selecting paths and nodes with sufficient resources can minimize this effect and ensure that the network is stable even during the worst scenarios. The contingencies are more common during the traffic bursts that lead to bottlenecks in network links and nodes. Contingencies in the network can be reduced by allocating resources, such as sufficient bandwidth, from SDN controller to switching nodes and processing power to the switch node selected for the controller location. These resources are defined as follows.

- *Available Bandwidth:* The higher the bandwidth, the less chance of delay during peak hours. The communication path between the SDN controller and a switching node should satisfy the minimum bandwidth requirement. The available bandwidth in the selected path link should be more than it is required for the switching node to communicate with the SDN controller.

$$\sum_{d \in W_d} \sum_{u \in W_s/\{d\}} \sum_{\pi \in \Pi_{du}} y_d \cdot \delta^l_\pi \cdot b_{du} \leq B_l; \quad \forall l \in L_s. \quad (36)$$

- *Resource Requirement at the SDN Controller Node:* The resources needed for optimal performance of the controller are CPU and memory. Resources available at the ECDC switching node should be more than those needed for seamless performance of the SDN controller node, as defined in Eq. (37) and Eq. (38).

$$p_c \cdot y_d \leq R_d; \quad \forall d \in W_d \tag{37}$$

$$r_c \cdot y_d \leq P_d; \quad \forall d \in W_d \tag{38}$$

where p_c and r_c are respectively the CPU and memory needed for the optimal operation of the SDN controller. P_d and R_u are the maximum CPU and memory available that the node $d \in D$ can spare during sporadic burst events.

5) Limit the total number of live SDN controllers at any given time to 1.

$$\sum_{d \in W_d} y_d = 1 \tag{39}$$

VI. PERFORMANCE EVALUATION

A. SIMULATION ENVIRONMENT

To assess the efficiency of the proposed IaaSP-SDN framework, we conducted experiments on the topology from taken topology Zoo [43]. Fig. 6 and 7 are some of the example of typologies used for the study. The yellow highlights represent the optimal location determined based on the mathematical modeling. Further discussion is done in the later section. We used a synthetic IaaS topology generator inspired from the GT-ITM tool proposed in [44]. In each IaaS request, the number of virtual nodes is randomly determined and uniformly distributed between 2 and 20. The minimum degree of connectivity degree is set to two links and the bandwidth requirement of virtual links are set according to *ODU* standards. Factors associated with Optical networking components such as tuning a laser, optical losses are relaxed and assumed to be negligible for the current evaluation.

The available CPU, memory, GPU and storage capacities of ECDC nodes are real numbers, uniformly distributed between 2 and 10 units. The available bandwidth capacity of the optical links is measured in terms of the number of available DWDM wavelengths $W = 16$.

The IaaS requests were fed to both GMPLS and IaaSP-SDN. The advantage of the SDN network is that we can merge routing and signaling application with resource manager through the use of SDN Controller's API. Thus, the SDN controller is responsible for both managing the Cloud computing ECDC resources and routing and signaling the devices. GMPLS performs only routing and signaling, using end nodes. The resource allocation runs separately.

For GMPLS, we used a GMPLS Lightwave Agile Switching Simulator (GLASS) [48]. For IaaSP-SDN, we used an ONOS controller and Mininet, with LINC OE (Link is Not Closed Optical Extension) switches. We assume that all the nodes in a network are capable of grooming traffic i.e., OTN Switches.

The whole system was run on a computer with i7 dual processor, 16 GB RAM and 128 GB flash memory. IBM CPLEX [49] was used for solving the ILP and MILP models used in IaaS provisioning and in the calculation of the optimal controller location.

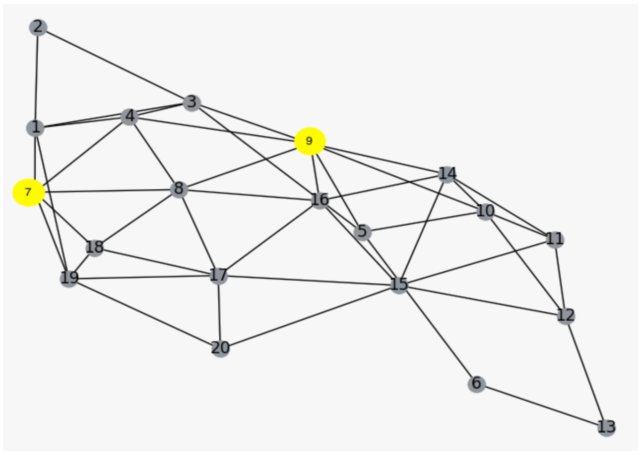

FIGURE 6. Example of topology for IaaS provisioning and controller.

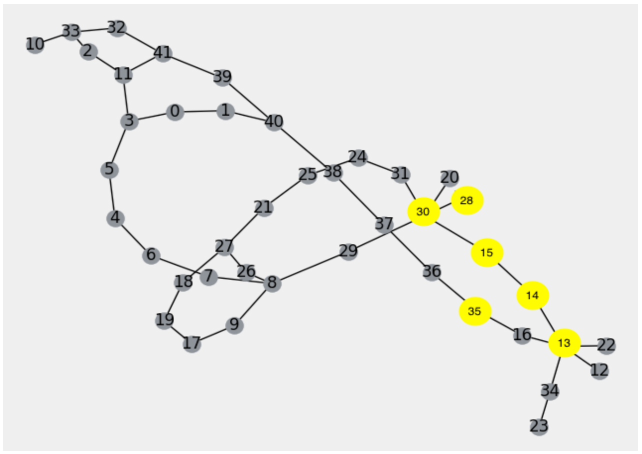

FIGURE 7. Topology used for the discussion of optimal controller location.

TABLE 3. Topology characteristics.

Attribute	Topology in Fig. 7	Topology in Fig. 6
Total Fibers	46	45
Total Nodes	41	20
Avg. Nodal Degree	2.19	4.5
Max Nodal Degree	5	7

TABLE 4. Average IaaS requirement.

Attribute	Value
CPU	46
GPU	41
RAM	2.19
CPU	5

B. RESULT ANALYSIS

1) IaaSP-SDN VS GMPLS MON

Comparison of SDN with GMPLS has always been of great interest among the research community as well as the industry [50], [51]. As pointed out in [50] the main difference between GMPLS and SDN is ''Whereas GMPLS offers distributed control, GMPLS/PCE is commonly regarded as having centralized path computation but still distributed provisioning/configuration; while OpenFlow centralized all the network control''. Lack of in-depth quantitative analysis comparing the performance of distributed architecture i.e., GMPLS to a completely centralized architecture i.e., OpenFlow based solution instilled in us to carry over this study. We believe to bring some of the performance differences between those two solutions. The performance of both IaaSP-SDN and GMPLS approaches depends on the number of switches and flows configured. The GMPLS performance was found to be highly unstable. Fig 8(a) compares the GMPLS and the IaaSP-SDN results for path setup time, for paths of different lengths, during a link failure event. Length is defined by the number of intermediate O-E-O conversions in a path i.e., OTN links. An O-E-O conversion is required to add or drop IaaS request signals. It is clear from the results that the GMPLS network, on average, took a round trip time nine times longer to converge in a worst case scenario. IaaSP-SDN took little more than five times the average round-trip time to re-establish the path. GMPLS takes more time because it does not keep track of the node and link resources; when a link fails, it has to re-execute the recovery. The RSVP-TE protocol has to locate the links with the required resources after the link failure has occurred. With IaaSP-SDN, the databases keep track of all the resources. As soon as a link failure is detected, the process of recovery begins. This saves time in collecting the resources available in the network.

The acceptance ratio of IaaS requests is directly related to the implementation of the control plane (Fig. 8(b)). A request accepted in IaaSP-SDN was dropped in GMPLS, for two main reasons. First, when an incoming request arrives, the RSVP-TE protocol of GMPLS goes through the available resources in the selected path. If that path cannot meet the request, RSVP searches the next path. Brute force and the lack of global view keep the incoming IaaS requests on hold for longer. As a result, the IaaS Resource Manager determines that it is best to drop those requests and to execute the next request in line. Second, the dropping occurs because GMPLS is not completely aware of the QoS requirements and considers that the path has too many intermediate O-E-O conversions. As a result, the IaaS resource manager assumes that the network does not fulfill the QoS requirements and drops the request. This lack of global view clearly makes GMPLS computationally expensive and non-optimal solution.

It is of vital importance that the control plane is scalable, especially with MONs, where nodes and links are being constantly added. The bootstrap time for GMPLS and IaaSP-SDN is shown in Fig. 8(c). The GMPLS performance is stable and reliable in smaller topology with fewer nodes and links. However, as the number of nodes and links increases, the time taken to search the network also increases. In the SDN framework, the time is relatively stable.

The throughput was calculated by sending TCP packets between the end nodes. The results in Fig. 8 (e) clearly show that GMPLS was highly unstable compared to IaaSP-SDN. The reason lies with the protocols used for traffic engineering. Routing and Wavelength Assignment (RWA) uses the Best-fit and OSPF algorithms in order to reduce the number of wavelengths used. Best-fit approach leads to increased congestion of links and instability of the control plane when processing the incoming packets. IaaSP-SDN not only considers factors such as link cost and the number of hops, but it also uses the First-fit algorithm for wavelength assignment. It also uses a large-scale optimization tool based on the Column Generation technique that is able to consider the specific requirement for IaaS requests. This ensures that the links are not overly crowded. The simulation results clearly show a constant throughput for the IaaSP-SDN framework.

Jitter values are shown in Fig. 8 (f). The graph clearly shows that the points are strongly correlated for both GMPLS and IaaSP-SDN when there is no link or node failure. The maximum jitter is because of the simulation of link failure. The maximum jitter of IaaSP-SDN is clearly more than the maximum jitter of GMPLS because only a single SDN controller is used. When a link failure occurs, the packet has to reach the controller, which in turn updates the required nodes. In the GMPLS network, the jitter is not very high: because the topology was small, the system was able to converge faster. However, jitter varied greatly with link failure. Jitter points in the graph clearly show a change from tightly coupled to moderately coupled, a change that indicates instability. This is because the GMPLS controller overloads the neighboring link for faster recovery.

2) EFFECT OF CONTROLLER PLACEMENT ON THE PERFORMANCE OF IaaSP-SDN PROVISIONING

In this section, we evaluate the performances of the IaaSP-SDN framework with respect to the SDN location selection strategies proposed in Section V-A.1. For the purpose of result discussion topology shown in Fig. 7 is used. Table 5 gives a summary of optimal SDN controller locations. Many proposals in the literature argue that a single controller is sufficient in a small-scale network [20], [29]. The proposals therefore

TABLE 5. Controller location nodes selected.

Objective Function	Scenario	NodeID
Average Minimum Distance	Case 1	13
Average Minimum Distance and Response Time	Case 2	30
Reliability and Reduced Response Time	Case 3	28
Based on Flow Table Usage	Case 4	28
Flow Table and Minimum Response time	Case 5	30

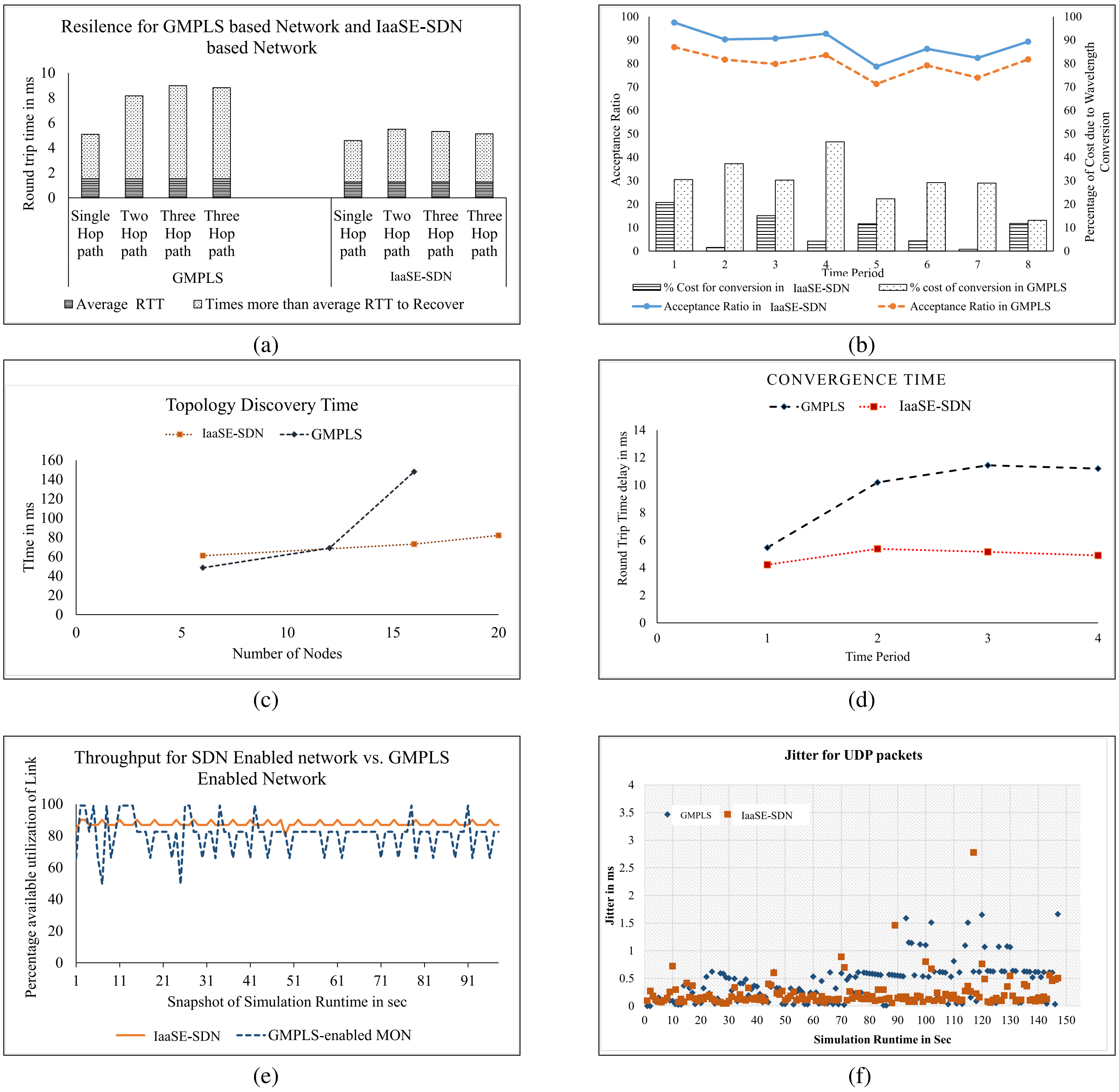

FIGURE 8. **Evaluation of GMPLS and SDN-based approach for IaaSP-SDN. (a): Path re-calculation time after a link failure event is triggered. (b): Acceptance ratio for GMPLS vs SDN controller. (c): Bootstrap time for discovering the nodes between IaaSP-SDN vs GMPLS. (d): Convergence Time from Failure Recovery (e): Throughput of TCP packets in IaaSP-SDN vs GMPLS. (f): Jitter variation in IaaSP-SDN vs GMPLS.**

select only one SDN controller location. The cost is defined based on the controller's use of the network for management and control purposes. Algorithm 1 shows implementation of node selection for the placement of SDN controller.

Fig. 9 (*a*) - Fig. 9 (*d*) provide an overview of the topology and show how different metrics in the network vary with each node. The results are arranged in increasing order of latency followed by distance, where applicable. Initially, all nodes are assumed to be potential controller locations. The worst response time of each path is recorded during peak hours. It is clear that the latency of some nodes that are relatively distant from each other is less than those closer together, for example nodes 3 and 11. Therefore using distance, as in [24] or the number of hops as a metric to find the optimal SDN controller location is not advisable. Fig. 9 (*d*) shows the cost vs. the Shared Risk Link Group (SRLG) factor value of different nodes. Intuitively, it can be said that the cost of having more disjoint paths from the controller to a network element can lead to higher cost. But, in a few cases, the cost is high while the SRLG is comparatively low. One such example is node 10. Optimizing a controller location based on a single metric may therefore lead to poor network performance, especially with MON, where the traffic and the type of access network vary.

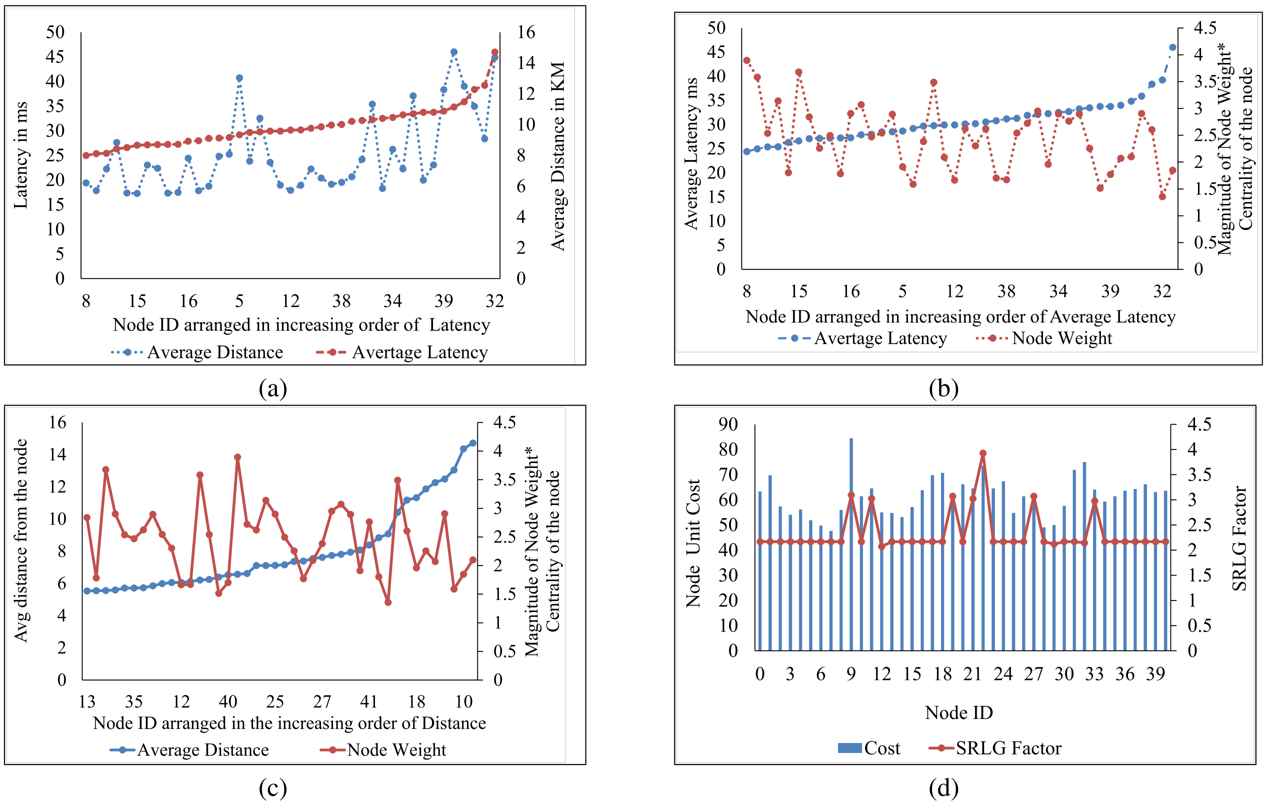

FIGURE 9. Initial state of node parameters. (a): Variation of average distance vs. latency. (b): Variation of response time vs. node weight. (c): Variation of average distance vs. Node weight. (d): Cost vs. SRLG factor.

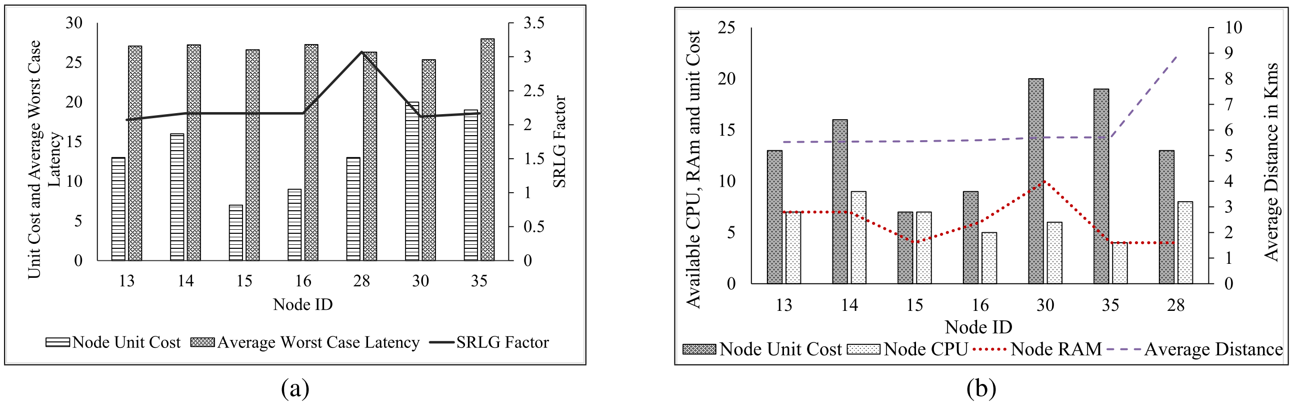

FIGURE 10. Nodes selected when the model is relaxed for Eq.33 by making it greater than 1. (a): Cost vs. Latency vs. SRLG. (b): Resource vs. average distance.

The nodes obtained for different strategies from the controller placement model are shown in Fig. 10 (a) and Fig. 10 (b). The location selected for the SDN controller is one that can reliably satisfy the networking QoS requirements. In topology show in Fig. 7, some of the potential candidates determined as controller location are 13, 14, 15, 16, 30, 35, 28. The results fluctuate depending on the MON topology and the available ECDC resources. The latency of all the potential candidates nodes is almost the same, although the distance is highly variable. The node resources play a crucial role, having to host multiple virtual controllers for the incoming IaaS requests. The controller should therefore have sufficient computing resources. It is interesting to note the high SRLG factor of node 28 (Fig. 10 (b)) that confirms that the location is very reliable even though the average distance from node 28 to all other nodes is high. It proves that optimization based on selecting the location only from the reliability factor might lead to non-optimal results during peak traffic hours.

Different strategies can have the same optimal SDN controller location. Cases 2 and 5 have the same location, as do cases 3 and 4. We therefore focus only on three strategies,

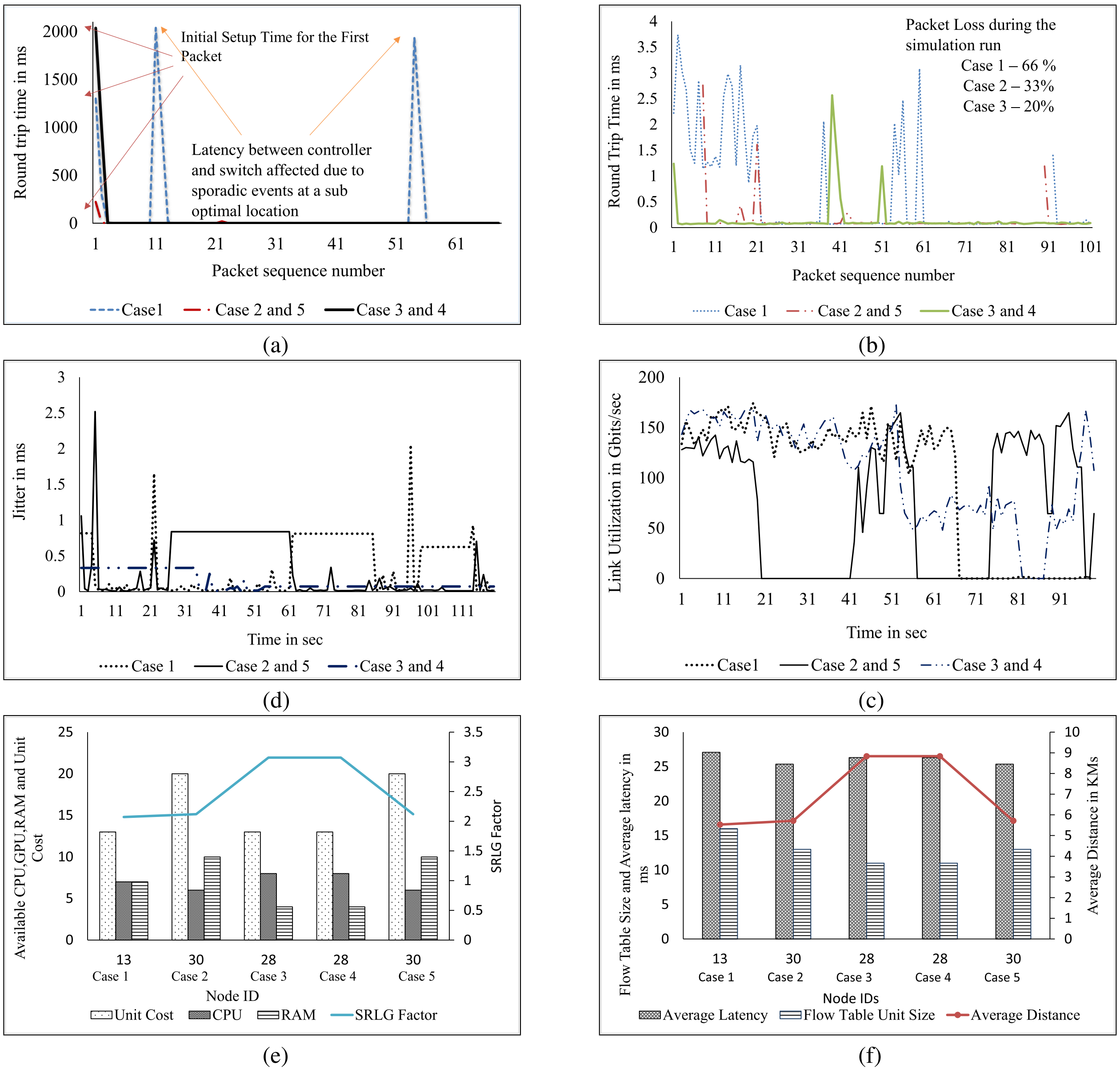

FIGURE 11. **Optimal location depending on selected objective function as summarized in Table 5. (a): Reactive forwarding of packets. (b): Reactive recovery time. (c): Jitter for UDP Traffic. (d): Bandwidth for TCP traffic. (e): Computing resource availability vs. SRLG. (f): Variation of average latency, flow table vs. average distance.**

Cases 1, 2 and 3. Case 1 selects the controller location by minimizing the distance from control plane to data plane. Case 2 optimizes distance and worst-case response time. Case 3 selects the controller location by minimizing latency and maximizing reliability.

Figures 11 shows QoS performances with different SDN controller locations. These results are obtained under the stressful conditions of link failures and sporadic traffic bursts. "Sporadic" in this article means bursts of huge (MTU 1500) packets and inter-packet times of 0.1s. Fig. 11 (*a*) evaluates the reactive forwarding of the accepted IaaS requests. The initial high value shows the time taken by the node-to-SDN-controller communication plus the SDN controller response time. Optimizing based only on distance (case 1) has an adverse effect if the intermediate nodes are overloaded. Fig. 11 (b) shows the recovery time with respect to reactive routing, where re-routing is performed when a link failure occurs. This is a highly CPU-intense task that incurs unnecessary overhead.

Fig. 11 (*c*) and Fig. 11 (*d*) evaluate the performance of TCP bandwidth and UDP jitter respectively. TCP is used for reliable communication and UDP is used for multimedia communication. It is important that both types of applications perform well in the network and were therefore evaluated under stressful conditions. Case 3 had the least downtime, proving that it is the most reliable controller location. Case 2 had

Algorithm 1 Selection Process of SDN Controller Location

```
Input  : Graph $G = (U, W_s, L_s)$
/* Cardinality of U and V is given by
   $n_1, n_2$; E defines Edge Set              */
begin
    foreach $IaaS$ in $G_n$ do
        if Constraints defined in Eqs: (33) -(39)
        satisfied then
            Find the total Cost of SDN controller based
            on the objective functions defined in Eq: 27 -
            32
        end
    end
    Return Optimal set of $W_d$ for all the given objective
    function
end
Output: $T$ mapped to G
```

low downtime and was able to converge faster, but with a constant drop in the network bandwidth. Case 1, where optimization was based on distance, had the highest downtime. Initially, the network was stable during the sporadic events, but when the simulation included both link failures and sporadic events, the system failed to converge within acceptable limits. Jitter was stable in all cases, except for a few sporadic bursts. Case 3 had the least jitter. From Fig. 11, it is clear that Case 3 was the most suitable SDN controller location for multimedia communication. Figures 11 (*e*) and 11 (*f*) compare different ECDC resources available at the controller location node, along with the average worst-case metrics. It shows that Case 3 has higher available resources and the highest shared risk, factors that plays a vital role in the performance of the SDN controller.

This evaluation shows that the performance of the SDN controller is highly dependent on the selected location attributes such as computational power, node to controller latency, and SRLG. A comprehensive evaluation of placement strategies is needed before the SDN controller is installed at a given location. This applies especially with MON topology, where a single SDN controller can fulfill the network requirement but a non-optimal location can lead to poor network performance. This may also give CSPs the impression that more SDN controllers are needed for efficient performance.

VII. CONCLUSION

This paper proposed a resource allocation framework of interconnected ECDC infrastructure to accommodate IaaS requests with stringent QoS requirement. The framework uses the global view made possible by the use of an SDN controller to manage the MON topology that interconnects ECDC sites. Provisioning IaaS requests into the infrastructure was also performed in a GMPLS test-bed. The results of performance comparisons revealed that GMPLS performed better than SDN when the topology had few network elements. However, as the network grows and multimedia communication forms the major part of the traffic, the distributive nature of GMPLS made it impractical.

The physical location of the SDN controller must be selected with care. We proposed a number of strategies to determine the optimal location. Each strategy showed distinct performances, making it difficult to choose a single methodology. We have shown that there is a fine trade-off between the various QoS metrics, proving that each MON topology must be considered separately with respect to its IaaS traffic pattern.

The proposed strategies for SDN controller placement focus on a single location. A MON using multiple standby SDN controllers to ensure resiliency in the control plane is not considered. In future work, we would like to extend this current proposal to multiple controllers, and to use a proactive approach for determining the optimal locations that can sustain traffic bursts and network failures.

COMPETING INTERESTS

The authors declare that they have no competing interests.

REFERENCES

[1] J. Reich, S. L. Woodward, A. N. Patel, and B. G. Bathula, "Leveraging SDN to streamline metro network operations," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 109–115, Oct. 2016.

[2] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 854–862.

[3] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 59–69, May 2011.

[4] C. Chapman, W. Emmerich, F. G. Márquez, S. Clayman, and A. Galis, "Software architecture definition for on-demand cloud provisioning," *Cluster Comput.*, vol. 15, no. 2, pp. 79–100, 2012.

[5] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Oper. Res.*, vol. 53, no. 6, pp. 1007–1023, 2005.

[6] D. Anderson. (2002). *Theoretical Approaches to Node Assignment*. [Online]. Available: http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps

[7] P. Fraternali, G. Rossi, and F. Sánchez-Figueroa, "Rich internet applications," *IEEE Internet Comput.*, vol. 14, no. 3, pp. 9–12, May/Jun. 2010.

[8] R. Nejabati, S. Peng, B. Gou, M. Channegowda, and D. Simeonidou, "Toward a completely softwareized optical network [invited]," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 12, pp. B222–B231, Dec. 2015.

[9] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 668–681, Mar. 2015.

[10] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[11] N. M. Mosharaf, K. Chowdhury, and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[12] F. Gu, M. Peng, S. Khan, A. Rayes, and N. Ghani, "Virtual network reconfiguration in optical substrate networks," in *Proc. Opt. Fiber Commun. Conf. Expo. Nat. Fiber Optic Eng. Conf. (OFC/NFOEC)*, Anaheim, CA, USA, 2013, pp.1–3.

[13] M. Jiachen, X. Liu, K. Wang, L. Liu, and H. Gu, "Virtual optical network embedding with SDN architecture based on memetic algorithm," in *Proc. 4th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Harbin, China, 2015, pp. 1050–1053.

[14] M. Capelle, S. Abdellatif, M.-J. Huguet, and P. Berthou, "Online virtual links resource allocation in software-defined networks," in *Proc. IFIP Netw. Conf.*, Toulouse, France, May 2015, pp. 1–9.

[15] L. R. Bays, L. P. Gaspary, R. Ahmed, and R. Boutaba, "Virtual network embedding in software-defined networks," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Istanbul, Turkey, Apr. 2016, pp. 10–18.

[16] P. Han, L. Guo, and Y. Liu, "Virtual network embedding in SDN/NFV based fiber-wireless access network," in *Proc. Int. Conf. Softw. Netw.*, Jeju, South Korea, 2016, pp. 1–5.

[17] X. Wen, Y. Han, H. Yuan, X. Zhou, and Z. Xu, "An efficient resource embedding algorithm in software defined virtualized data center," in *Proc. IEEE Global Commun. Conf.*, San Diego, CA, USA, Dec. 2015, pp. 1–7.

[18] R. Guerzoni *et al.*, "A novel approach to virtual networks embedding for SDN management and orchestration," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Krakow, Poland, May 2014, pp. 1–7.

[19] C. Develder, J. Buysse, B. Dhoedt, and B. Jaumard, "Joint dimensioning of server and network infrastructure for resilient optical grids/clouds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1591–1606, Oct. 2014.

[20] H. Brandon, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, New York, NY, USA, 2012, pp. 7–12.

[21] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–6.

[22] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards SDN," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 363–368.

[23] M. T. I. ul Huque, G. Jourjon, and V. Gramoli, "Revisiting the controller placement problem," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Clearwater Beach, FL, USA, Oct. 2015, pp. 450–453.

[24] K. S. Sahoo, B. Sahoo, R. Dash, and N. Jena, "Optimal controller selection in software defined network using a greedy-SA algorithm," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, New Delhi, India, Mar. 2016, pp. 2342–2346.

[25] A. M. Sallahi and M. St-Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 274–277, Feb. 2017.

[26] T. Y. Cheng, M. Wang, and X. Jia, "QoS-guaranteed controller placement in SDN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[27] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *Proc. 8th Int. Workshop Resilient Netw. Design Modeling (RNDM)*, Halmstad, Sweden, Sep. 2016, pp. 253–259.

[28] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards controller placement for robust software-defined networks," in *Proc. IEEE 34th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nanjing, China, Dec. 2015, pp. 1–8.

[29] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 4–17, Mar. 2015.

[30] F. Zaman, A. Jarray, and A. Karmouch, "QoS aware virtual network embedding in SDN-based metro optical network," in *Advances in Network-Based Information Systems*, L. Barolli, T. Enokido, and M. Takizawa, Eds. 2017.

[31] A. Jarray, J. Salazar, A. Karmouch, J. Elias, and A. Mehaoua, "Column generation based-approach for IaaS aware networked edge data-centers," in *Proc. IEEE Globecom Workshops*, Austin, TX, USA, Dec. 2014, pp. 93–98.

[32] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.

[33] A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Dynamic restoration with GMPLS and SDN control plane in elastic optical networks [Invited]," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 2, pp. A174–A182, Feb. 2015.

[34] S. Das, G. Parulkar, and N. McKeown, "Why OpenFlow/SDN can succeed where GMPLS failed," in *Proc. Eur. Conf. Exhibit. Opt. Commun.*, 2012, Paper Tu.1.D.1.

[35] E. Aguiar *et al.*, "A real-time video quality estimator for emerging wireless multimedia systems," in *Wireless Netw.*, vol. 20, no. 7, pp. 1759–1776, 2014.

[36] R. L. Gomes, L. Bittencourt, E. Madeira, E. Cerqueira, and M. Gerla, "Management of virtual network resources for multimedia applications," *Multimedia Syst.*, vol. 23, no. 4, pp. 405–419, Jul. 2017.

[37] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, Apr. 2014.

[38] D. Kim, M.-I. Kim, M.-K. Noh, B.-Y. Park, G.-S. Yu, and S.-H. Kim, "Multimedia network service environment based on distributed virtual network management," in *Embedded and Multimedia Computing Technology and Service*. 2012, pp. 73–80.

[39] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[40] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, Chicago, IL, USA, 2014, pp. 1–6.

[41] GitHub. (2017). *FlowForwarding/LINC-Switch*. Accessed: Jan. 23, 2017. [Online]. Available: https://github.com/FlowForwarding/LINC-Switch

[42] E. Mannie, *Generalized Multiprotocol Label Switching Architecture(GMPLS)*, document RFC 3945, IETF, 2004.

[43] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[44] E. W. Zegura, K. L. Calvert, and B. Bhattacharjee, "How to model internetwork," in *Proc. IEEE INFOCOM*, Mar. 1996, pp. 594–602.

[45] Y. Kim *et al.*, "GLASS (GMPLS lightwave agile switching simulator)—A scalable discrete event network simulator for GMPLS-based optical Internet," NIST, Gaithersburg, MD, USA, White Paper, 2002.

[46] A. Jarray and A. Karmouch, "Periodical auctioning for QoS aware virtual network embedding," in *Proc. IEEE 20th Int. Workshop Qual. Service (IWQoS)*, Coimbra, Portugal, Jun. 2012, pp. 1–4.

[47] D. Eppstein, "Finding the K shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1999.

[48] Ssfnet.org. (2017). *Scalable Simulation Framework*. Accessed: Jan. 23, 2017. [Online]. Available: http://www.ssfnet.org/homePage.html

[49] (Sep. 2017). *IBM ILOG CPLEX 12.3 User Manual*. [Online]. Available: ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/refcppcplex.pdf

[50] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2738–2786, 4th Quart., 2016.

[51] L. Liu, T. Tsuritani, and I. Morita, "From GMPLS to PCE/GMPLS to OpenFlow: How much benefit can we get from the technical evolution of control plane in optical networks?" in *Proc. 14th Int. Conf. Transparent Opt. Netw. (ICTON)*, Coventry, U.K., Jul. 2012, pp. 1–4.

[52] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 783–791.

FAISAL A. ZAMAN received the M.A.Sc. degree in software-defined metro optical network, in 2017. He is currently a Research Assistant with the Imagine Lab, University of Ottawa. His research interests include SDN, openflow in optical network, and applied combinatorial optimization in networking.

ABDALLAH JARRAY received the M.S. and Ph.D. degrees in computer science from the University of Montreal, Montreal, QC, Canada, in 2005 and 2010, respectively. He is currently an Adjunct professor in electrical and computer engineering with the University of Ottawa, Canada. His research interests include network virtualization, software-defined networking, cloud computing, wireless body area networks, femtocell networks, and optical networks. His areas of expertise include, but are not limited to, optimization technique for network design problem, integer linear programming, decomposition approaches, column generation, metaheuristics, and game theory.

AHMED KARMOUCH received the M.S. and Ph.D. degrees in computer science from the University of Paul Sabatier, Toulouse, France, in 1976 and 1979, respectively. He is currently a Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He was the Industrial Research Chair with the Ottawa Carleton Research Institute and the Natural Sciences and Engineering Research Council. He has been the Director of the Ottawa Carleton Institute for Electrical and Computer Engineering. He is involved in several projects with industry and government laboratories in Canada and Europe. His current research interests include mobile computing, autonomic overlay networks, software-defined networks, mobile cloud computing, and network virtualization. He organized several conferences and workshops and edited several books, and served as a Guest Editor for the *IEEE Communications Magazine*, *Computer Communications*, and others.