Optimized Provisioning of SDN-enabled Virtual Networks in Geo-distributed Cloud Computing Datacenters

Khaled Alhazmi, Abdallah Shami, and Ahmed Refaey

Abstract: **Cloud computing provides on-demand IT services via large distributed datacenters over high-speed networks. Virtualization, a key cloud computing technology, allows service providers to offer computing services in cloud environments without platform compatibility discrepancies. The recent proliferation of cloud computing has rekindled interest in network virtualization. Thus, network virtualization is emerging as a polymorphic approach for the future Internet that will facilitate the use of shared resources. Virtual network provisioning is considered to be a main resource allocation challenge in any virtualized network environment. Software-defined networking (SDN) imparts flexibility to a network by removing the control layer from the data transfer layer of the network and moving it to the control plane. Network virtualization is further employed to share physical infrastructure to enable multiple service providers to access the network. Flexible access requires efficient management of network resources; the SDN control plane can be used for efficient management of virtual networks. In this study, we formulate virtual network provisioning in SDN-enabled, geographically distributed cloud computing datacenters as a mixed integer linear programming (MILP) problem. The formulation of the proposed optimized virtual network provisioning (OVNP) model is studied by means of simulations. The performance of the proposed approach is measured against enhanced network cloud provisioning (ENCP), our previous research, and other recognized research focused on the ratio of successfully provisioned requests and the efficiency of resource utilization. The results verify the effectiveness of the proposed approach.**

Index Terms: **Cloud computing, cloud datacenters, dynamic resource allocation, future Internet, network virtualization, software defined networking, virtualization, virtual software-defined networking (SDN) provisioning.**

Manuscript received October 13, 2016; approved for publication by YU Hua, Division III Editor, April 02, 2017.

This work was funded in part by King Abdulaziz City for Science and Technology through the Cultural Bureau of Saudi Arabia in Canada.

K. Alhazmi is with the Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada, and also with National Center for Computer Technology and Applied Math, Communication and Information Technology Research Institute, King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia, email: kalhazmi@uwo.ca, khazmi@kacst.edu.sa.

A. Shami is with the Department of Electrical and Computer Engineering, Western University, London, Canada. email: abdallah.shami@uwo.ca.

A. Refaey is with Manhattan College, NY, USA. email: ahmed.hussein@manhattan.edu.

K. Alhazmi is the corresponding author.

Digital object identifier: 10.1109/JCN.2017.000064

I. INTRODUCTION

CLOUD computing provides on-demand IT services via large distributed datacenters over high-speed networks. The benefits of cloud computing, which include on-demand availability, scalability, and the pay-per-use model, have prompted businesses to switch to the cloud in order to reduce the overall cost of computing. To operate in the cloud environment, users must be provided with robust performance by service providers: This can be achieved when the technologies used work efficiently and the system is managed optimally. Virtualization, which is the key technology in cloud computing, allows service providers to offer platform-compatible computing services in the cloud environment, which relieves the client from troubleshooting compatibility issues. In addition, virtualization enables service providers to improve the utilization of their server and storage capacities while providing them with the availability and flexibility required to service a large number of clients [1], [2].

Cloud computing, with its distributed datacenter approach, requires datacenters to be located as close to the client as possible in order to implement low-latency and real-time services. Service providers are required to manage and control their distributed datacenters; thus, the recent proliferation of cloud computing has rekindled interest in network virtualization. Indeed, network virtualization is emerging as a polymorphic approach for the future Internet that will facilitate the use of shared resources. It allows multiple networks to exist in a single substrate network. A virtual network consists of virtual nodes that are connected by links and must be provisioned on the physical network. Virtual network provisioning is considered to be the main resource allocation challenge in any virtualized network environment. Provisioning is the process of efficiently allocating the physical resources available to the virtual components available within the network. Network virtualization plays an important role in current cloud platforms because it is provided as a service [3]–[5].

The core networks in large-scale cloud datacenters must be flexible in order to meet changing requirements. The use of programmable networks has been proposed to facilitate a flexible networking environment. Software-defined networking (SDN) imparts flexibility to a network by removing the control layer from the data transfer layer of the network and moving it to the control plane. It also reduces CapEx and OpEx (up to $32 billion annually [6]), and increases the generated revenue of cloud service providers. The network is then managed by an entity called the SDN controller, which maintains an overall view of the network and allocates or configures networking resources dynami-

cally as per the system requirements [7]. The control plane defines the destination where the data is to be sent via a central SDN controller. This controller can configure and manage the flow of data packets through switches in the data plane. The data plane constitutes the system that actually routes the data traffic to the defined destination.

Network virtualization is used to share physical infrastructure in order to enable multiple service providers to access the network; this access requires efficient management of network resources. The SDN control plane promotes efficient management of virtual networks, thus, the SDN is an effective tool to facilitate the implementation of network virtualization. SDN utilizes software to manage, program, and virtualize the network, similar to the manner in which hypervisors virtualize a physical server setup. Moreover, virtualized SDN-based network architecture allows cloud service providers to implement versatile public cloud technology in a cost-effective manner. The open application platform interface will (1) enable end-users and service providers to obtain network resources on demand, (2) support need-based resource allocation, and (3) allow customers to control the procurement of resources. SDN, as an enabler of network virtualization, can expand the services provided by cloud service providers and offer an even higher level of innovation [8].

Cloud datacenters receive a large number of requests from clients for resource allocation and processing. In turn, service providers are required to efficiently allocate and schedule the requests in their distributed datacenters. Cloud service providers allow clients to reserve various types of virtual machines (VMs) along with connection requests. Each client requires a certain quality of service (QoS) to be maintained, regardless of the resources being shared with other clients. Therefore, the system must allocate resources to VMs dynamically so that the resources can be utilized effectively [9]. Such VM resources may be located in geographically distributed locations, and in order to acquire data exchange capabilities, the client rents the bandwidth required for inter-datacenter communications. In traditional approaches, fixed bandwidths are allotted to fulfill service-level guarantees [4]. However, these approaches do not utilize networking resources optimally. This problem can be effectively resolved by a central controller that has an overall view of the network and dynamically allocates network resources by considering QoS in conjunction with availability. SDN, when used in a network environment, can perform this task with its network controller, which is appropriately assigned to have an overall view of the network and its resources. The SDN controller can be programmed to meet system requirements, define suitable policies, and deliver efficient solutions. With these features, the SDN can allocate networking resources dynamically and proficiently as dictated by the demands of VM clients, while accounting for link and node resources [10].

In this study, we formulate virtual network provisioning in SDN-enabled geographically distributed cloud computing datacenters as a mixed integer linear programming (MILP) problem. We assume that the central management controller, which includes the cloud controller and SDN controller, has complete knowledge of both the physical network resources and the datacenters in terms of servers, computational resources, and link resources. With this comprehensive observation, the controller can dynamically and efficiently manage and optimize cloud resources–including network link resources and computational resources–while providing scalability and flexibility that satisfies the cloud client's requirements. Most cloud providers adopt the best-effort approach to fulfill the QoS requirements of cloud clients. The proposed MILP formulation, referred to as optimized virtual network provisioning (OVNP), allows the central controller to provision the aggregated cloud connections of the virtual SDN network to the underlying substrate network such that the number of cloud clients served is maximized and the allocated resources are minimized. For a virtual network to be provisioned, it must be active during a predefined time frame (determined dynamically). This provisioning adopts an online approach for handling requests from cloud clients. Moreover, the proposed OVNP model was compared to state-of-the-art algorithms that tackled the same problem.

The remainder of this paper is organized as follows: Section II provides a brief review of related studies; Section III presents the role of OVNP in SDN-based cloud environments; Section IV introduces the OVNP problem formulation and the proposed algorithm; Section V describes our simulation environment and the quantitative performance evaluation of the proposed approach; Section VII summarizes our findings and concludes the proceeding analysis; finally, Section VI summarizes the contributions of the present study.

II. RELATED WORKS

A. *Virtual Network Provisioning*

Efficient virtual network provisioning, which is the key to providing robust solutions to cloud computing clients, requires efficient network virtualization techniques. In the literature, network virtualization has been regarded as a key technology for achieving various provisioning goals. In addition, network virtualization has been considered the key component of future Internet technologies [11]–[13], as it allows different technologies to exist over a common physical infrastructure. Network virtualization is also advocated as a solution to the problem of Internet ossification, as it allows for controlled deployment of new architectures [14]–[17]. In [18], approaches for provisioning virtual networks have been compiled and the constraints on achieving an optimal solution have been listed. Specifically, "node and link resource constraints, limited substrate resources, the dynamic nature of the virtual network requests' (VNRs') arrival, and VNRs diverse topologies impose challenges on the process of virtual network mapping (VNM)" [18]. Dynamic provisioning of a virtual network requires an efficient approach for embedding virtual resources into the available physical resources [10]. Most researchers have not taken into account the simultaneous occurrence of one or more of these constraints; this has led to simplified and constricted solutions. Virtual network embedding is a difficult task owing to the use of diverse topologies, resource constraints, online requests, and admission control [19]. The complexity can be illustrated by the way that researchers have termed the problem as "NP-hard" [11]–[13], [20], [21].

The VNM process requires node mapping as well as link

mapping; this can be achieved in two stages or by using co-ordination between the two stages to improve efficiency. A set of virtual network embedding algorithms (ViNEYard) constitutes one such approach that co-ordinates node and link mapping stages [19]. The authors proposed an augmented graph by connecting a virtual node to each physical node. ViNEYard considers multicommodity flow to perform link mapping, and this method of link mapping is employed in this work. In [20], a substrate node with the required resources and bandwidth is selected using a node mapping algorithm. In [21], a technique called VHub approaches mapping as a mixed integer program to reduce the usage of physical resources and balance the load evenly. The virtual paths for the substrate are selected using a link-mapping algorithm. The author used path migration for distributing virtual links to different paths in order to maximize the utilization of the virtual network (VN) in the substrate network. Splitting of the virtual network provisioning request has been solved using techniques such as max-flow min-cut algorithms and linear programming [22]. In [11], the authors proposed integer linear programming formulation for simultaneous optimization of node and link mapping. Mapping of nodes and links in the same stage has also been proposed in [23]. The authors proposed a VNM algorithm based on subgraph isomorphism detection for mapping. In [17], the authors showed that the use of internal topologies for virtual mapping provides less efficient solutions and creates undesirable constraints. Virtual network requests were represented using traffic matrices, which were solved using mixed integer programming formulation. Another approach for virtual network embedding solutions was proposed in [24]. The traditional approach of imposing restrictions on the problem space or using heuristic algorithms was discarded, as these do not use the substrate resources efficiently. The authors proposed that the substrate network be re-designed so that simpler embedding algorithms can be used and the substrate resources be distributed more efficiently without imposing restrictions on the problem space. The virtual link is dispersed over multiple substrate paths, at which point path migration is employed to improve ink utilization as new requests are received.

B. Virtual Network Embedding in Cloud Computing

The end-to-end QoS in distributed cloud applications is a cause for concern among cloud clients. In most cases, cloud infrastructures and communication networks work independently of each other, making it difficult to guarantee QoS. Many approaches have been proposed to provide QoS in a cloud environment [5], [25]–[28]. A virtual-network-as-a-service (VNaaS) model was proposed by [5] to reduce the latency experienced in communication networks. The model maps the requirements to create virtual links with differentiated quality among datacenters that fulfill the QoS requirements. In [25], end-to-end availability and latency was guaranteed by modeling problems as linear optimization problems to control network and cloud resources, whereas mixed integer programming (MIP) and a heuristic methodology were adopted in [26] and [28] for the same purpose. Minimizing the rejection rate of requests by ensuring the availability of sufficient bandwidth, memory, and processing power was the key objective of the proposal presented by [27]. This strategy aims to enhance cloud revenue but

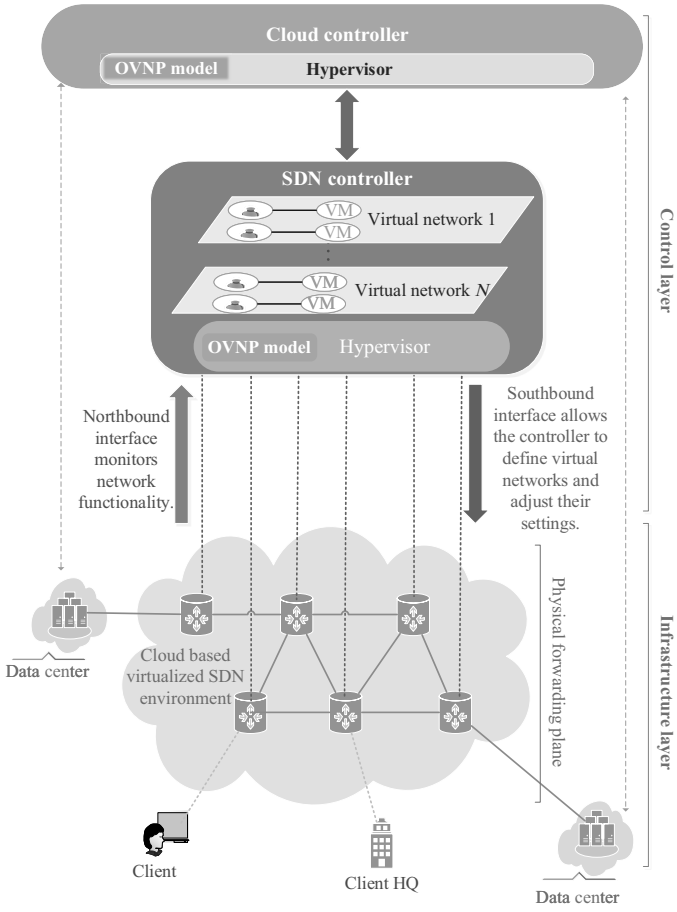

Fig. 1. OVNP in SDN-based cloud environment.

does not consider issues such as network latency. Most of the above mentioned studies have focused on solving the virtual network embedding problem and have not investigated the type of communication required by virtual networks. Communication-related issues have been discussed by [29] and [30].

C. SDN with the Concept of Network Virtualization

In [31], the SDN controller used an MILP formulation to determine the optimum virtual end-to-end paths, whereas [32] adapted an existing resource mapping algorithm for use with SDN in a networked cloud environment. Furthermore, the use of MILP for coordinated node and link mapping was proposed by [33]. Most of these schemes do not guarantee survivability in the event of network virtualization failures; to address this problem, [34] proposed the use of a VNM algorithm with coordinated primary and backup topology (VNM-PBT).

FlowVisor is a major facilitator of the SDN virtualization process. It causes segmentation of the SDN flow tables in the OpenFlow switches into different slices, and each slice instance can be managed and controlled by independent OpenFlow controllers [35], [36]. Autoslice is considered as another concept that envisions a virtualization layer to support implementation of the software defined network paradigm in separate slices. Conceptualized by Bozakov *et al*. in [37], this study is significant be-

cause it aims to streamline and automate the SDN virtualization process. The Autoslice control plane incorporates distributed hypervisor architecture that can handle multiple flow table requests from multiple clients concurrently. Niciras network virtualization platform (NVP) uses overlay networking to provide abstraction from the physical hardware. The overlay network offers each tenant an abstracted version of a single switch that connects all of its virtual machines [36], [38]. The rules that control the encapsulation and updating of data packets whenever a virtual machine moves is handled by a logically centralized controller.

III. ROLE OF OVNP IN SDN-BASED CLOUD ENVIRONMENT

In order to handle the substantial number of requests received by the central controller in large and distributed datacenters, techniques such as connection request aggregation may be adopted. These techniques require the use of proper aggregation techniques, request prioritization, window sizing, and so on. This issue was addressed in our previous works [9] and [18] by using an algorithm that selects a proper window size for VNM.

Although this paper addresses the problem of optimal provisioning of virtual networks in SDN-enabled geo-distributed cloud datacenters, it is vital to show its role in an SDN-enabled cloud environment. Such an environment makes it easier to virtualize the physical switches and routers, because components do not need to initiate their own instances of control plane software [36]. Fig. 1 highlights the different components and interfaces in an SDN-enabled cloud environment. The SDN control layer determines how data are to be routed to their destination [39]. The controller layer allows network operators to have centralized control over the network hardware utilities. There are three main SDN control plane interfaces, and each of them has its own specific settings. These interfaces facilitate a steady flow of communication between the different layers of the SDN architecture. The southbound interface lies between the SDN controller and the underlying data plane and facilitates communication between the SDN controller and the data forwarding plane. This interface allows the controller to define virtual networks and adjust their settings in a highly prioritized, need-based manner. The northbound interface, which is accessible to end users and customers, monitors network functionalities. Within the context of this work, the central cloud controller receives multiple cloud connection requests from multiple cloud users. Next, cloud connection requests are aggregated into virtual network requests, where they are received by the SDN controller and separate control logic is built for each virtual network request. The hypervisor within the cloud, as well as the SDN controller provided via the proposed OVNP model, will determine the optimal provisioning solution for the virtual network requests with respect to the underlying substrate network. Following which, the hypervisor of the SDN controller will transfer this solution into forwarding rules to be sent to the underlying physical forwarding plane through the southbound interface.

It is worth noting that several studies have focused on SDN controller placement, as it is one of the commonly known challenges in SDN [40]–[43]. Several heuristic techniques and optimization algorithms have been proposed to tackle SDN controller placement, owing to its hardness level (NP-hard). Any of these heuristics can be used to determine the controller placement, based on the desired design metric (e.g., latency, fault tolerance).

In the SDN environment, the traffic statistics collection is performed via the southbond interface which facilitate the communication between controller and SDN switches/network devices. OpenFlow is one mechanism of this communication. It has multiple features for monitoring and management. It provides the required interfaces to get traffic statistics from underlying SDN switches. Among these features, there are two messages of interest, namely PacketIn and FlowRemoved. These two messages are sent by the OpenFlow switch to the controller whenever the flow arrives and departs respectively. They are often used for push-based (passive) flow statistics collection approaches. In our work, a similar mechanism is employed, by reason of, when a set of cloud connection requests arrive, they are aggregated to form a VNR. This mimics the PacketIn feature available in OpenFlow switches. Furthermore, whenever a VNR departs, the substrate is updated and the information is also reported back to the SDN controller. This in turn resembles the FlowRemoved feature in OpenFlow. Several works have shown that the push-based flow statistics collection approaches (passive approaches) minimize the measurement cost for utilization monitoring. This is due to the fact that performance can be inferred based on the passive capturing and analysis of control messages between the switches and the SDN controller [44], [45].

Additionally, these works studied the tradeoff between the resources consumption on monitoring and the granularity of the statistics collection on both the time and address-space dimensions. To that end, we employed the windowing technique which limits the overhead that results from the aggregated incoming requests. Also, the aggregation of the requests into VNRs further reduces the effect of statistics collection since aggregation reduces the number of messages sent from the switches to the SDN controller [44]–[46]. Hence, the provisioning performance is not affected by the granularity of the flow statistics collection process.

IV. VIRTUAL NETWORK PROVISIONING MODEL AND PROBLEM FORMULATION

To solve the problem of virtual network provisioning in a geographically distributed cloud computing environment, this paper introduces an analytical formulation in which we model the problem as an MILP problem. This model can be used by cloud providers to efficiently allocate computational and networking resources, with the objective of serving the maximum number of cloud users with minimum physical cloud resources. We modeled the optimization problem–that is, maximizing the ratio of served cloud connection requests and minimizing the cost of provisioning these requests–as an MILP problem. This modification should, however, satisfy the requirements of different cloud clients' virtual connection requests. The problem definition and the various components involved in this model are described in the following section.

A. The Substrate Network

One of the inputs to the MILP is the undirected physical network denoted by $G_p(X, E)$, where X represents the set of physical nodes and E is the set of physical links. A physical node can be a datacenter C, which serves cloud connection requests as per the controller provisioning decision, or a client node S, which generates cloud connection requests (i.e., $X = \{C, S\}$). Each datacenter $c \in C$ has a number of servers. The total set of servers available in all data centers is denoted as Ser, and each server $ser \in Ser$ is associated with a finite computing capacity i. Note that a server is not considered a physical node within the network topology graph. The computing resources used in this work are CPU, memory, and storage. $T^i_{ser,c}$ represents the total available capacity i ($i = 1$ for CPU, $i = 2$ for memory, and $i = 3$ for storage) of server ser in datacenter $c \in C \subseteq X$. Similarly, each physical link $uv \in E$ has a limited bandwidth denoted by δ_{uv}.

B. Virtual Network Request

Most previous studies considered virtual network requests as predefined, and these virtual networks were generated randomly. However, in our work, the scenario is different; we consider real-world requests in the cloud environment. Cloud users reserve or rent VMs with different configurations for a certain period of time (duration) and request a connection with their VMs. In each connection request, the client defines the source (client node), duration, requested VM specifications (capacity units), required bandwidth, and allowed waiting time. Data exchange between the end users and their VMs will be performed through these cloud connection requests. A set of cloud connections that belong to different cloud clients are aggregated based on a preset aggregation factor. The aggregated cloud connection requests will be abstracted as a VNR. The VNR is modeled as an undirected graph $G_v(V, D, R, B)$, where V represents the set of virtual nodes, D denotes the set of cloud connection requests within the VN request, which are considered virtual links, R represents the set of requested capacities (e.g., CPU, memory, and storage) of the clients' VMs, and B denotes the set of requested bandwidths of the cloud connection requests. The virtual nodes V can be either client source nodes or their VMs.

C. Key Notations Used in This Work

Tables 1 and 2 present the key notations that are used in this work.

D. Decision Variables:

The decision variables as follows:

$$M_d = \begin{cases} 1, & \text{if cloud connection request } d \text{ is mapped;} \\ 0, & \text{otherwise.} \end{cases}$$

$$y^{d_e}_{ser,c} = \begin{cases} 1, & \text{if destination node } d_e = a \in \text{A is mapped} \\ & \quad \text{in server } ser \in Ser \text{ in physical datacenter} \\ & c \in C \subseteq X; \\ 0, & \text{otherwise.} \end{cases}$$

Table 1. Sets used in the model.

Sets	
X	Set of physical nodes
E	Set of physical links
S	Set of client nodes $S \subseteq X$
C	Set of datacenter nodes $C \subseteq X$
Ser	Set of servers in the all datacenters of cloud infrastructure
A	Set of requested virtual machines
V	Set of virtual nodes with $V = A \cup S$
D	Set of cloud connection requests
R	Set of capacities such as CPU, memory, storage{1,2,3}
B	Set of required bandwidth of the cloud connection requests
J	Set of time intervals $\{1 \leq j \leq Wmax\}$

Table 2. Parameters used in our model.

Parameters		
Notations	Meaning	Domains
s and e	Index the source and destination nodes of cloud connection request	>0
u and v	Index the nodes in the physical topology of the substrate network	>0
ser	Index the servers in the cloud infrastructure	>0
c	Index the cloud datacenter in the cloud infrastructure	>0
d	Cloud connection request $d \in D$, such that $d = (d_s, d_e)$	>0
d_s	Source node of cloud connection d	>0
d_e	Destination node of cloud connection d, which is equivalent to a required VM ($d_e \equiv a \in A$)	>0
b_d	Requested bandwidth for cloud connection request $d \in D$	>0
r^i_d	Requested capacity $\{i = 1, 2, 3\}$ (CPU, Memory, Storage), for cloud connection request d	>0
$T^i_{ser,c}$	Total available computational capacity i of server ser in datacenter $c \in C \subseteq X$	>0
δ_{uv}	Available bandwidth of physical link uv, where $u, v \in X$	>0
W^d_j	1 if cloud connection request d is active during interval j, 0 otherwise. It is a parameter that indicates whether a cloud connection request d is active during time window j, based on the knowledge of the request arrival time and boundary of the window size	>0
$Wmax$	Total number of windows	>0
L	Large number	>>0

f^d_{uv}: Requested bandwidth for cloud connection request d routed over physical link uv, 0 otherwise.

E. Mathematical Model:

E.1 Objective Function

The objective function (1) is a weighted sum of two main parts. The first part attempts to maximize the ratio of served connections M_d; hence, the ratio of virtual network acceptance is maximized. The second part aims to minimize the cost of provisioning cloud connection requests, and this is represented by the summation of the bandwidth reserved for cloud connection requests over all substrate edges. Therefore, to provision a virtual link, the second term of the objective function will provision that link to substrate links/paths with minimum resources; therefore, network physical resource usage/utilization is minimized.

Further, α and β are tuning parameters for setting the weight of parts in the objective function.

$$\text{Max} \quad \alpha \sum_{d \in D} M_d - \beta \sum_{u,v \in X} \sum_{d \in D} (f_{uv}^d + f_{vu}^d) \tag{1}$$

E.2 Constraints

- ***Bandwidth conservation***

$$\sum_{d \in D} (f_{uv}^d + f_{vu}^d) W_j^d \leq \delta(u,v) \qquad \forall u, v \in X, \forall j \in J \tag{2}$$

Constraint (2) deals with the bandwidth limit δ on a substrate link uv. The total bandwidth of cloud connection requests (within window j) allocated to a physical link should not exceed the available bandwidth of that physical link $\delta(u,v)$. Further, W_j^d is a parameter that indicates whether a cloud connection request d is active during time window j, based on the knowledge of the request arrival time and boundary of the window size.

- ***Provisioning of virtual links to physical links***

Constraints (3)–(5) deal with the provisioning of the virtual networks cloud connection requests to the substrate edges, which are revealed through flow conservation. Using these equations, the provisioning of cloud connection requests is determined and the provisioning of the nodes is carried out using the decision variable $y_{ser,c}^{d_e}$ in the link provisioning equations. Thus, node and link provisioning are carried out in a single stage.

$$\sum_{n \in X} f_{d_s n}^d - \sum_{n \in X} f_{n d_s}^d = b_d . M_d \qquad \forall d \in D \tag{3}$$

$$\sum_{n \in X, n \neq c} f_{cn}^d - \sum_{n \in X, n \neq c} f_{nc}^d = -b_d . \sum_{ser \in Ser} y_{ser,c}^{d_e} \qquad \forall d \in D, c \in C \subseteq X \tag{4}$$

$$\sum_{m} f_{nm}^d - \sum_{m} f_{mn}^d = 0 \qquad \forall d \in D, \forall m \in X, \forall n \in X \setminus \{d_s, c \in C\} \tag{5}$$

In constraint (3), the net flow to the source node d_s of the connection request, which represents the client node, must be equal to the requested bandwidth of cloud connection b_d given that this cloud connection request is provisioned, $M_d = 1$.

In constraint (4), if $y_{ser,c}^{d_e} = 1$, the destination node d_e of the connection request, which is the requested virtual machine of cloud connection request d, is provisioned to datacenter $c \in C \subseteq X$. Then, the net flow to the destination node at datacenter c must be equal to $-b_d$.

In constraint (5), the net flow to the intermediate nodes between the source (client node) and destination (datacenter) must be equal to zero.

- ***Domain constraints***

$$f_{uv}^d \geq 0 \qquad \forall u, v \in X, \forall d \in D \tag{6}$$

$$M_d \in \{0, 1\} \qquad \forall d \in D \tag{7}$$

Constraints (6) and (7) are domain constraints. They state that the flow decision variable should be greater than 0 while the connection request mapping decision variable should be binary.

- ***Node provisioning***

$$\sum_{c \in C \subseteq X} \sum_{ser \in Ser} y_{ser,c}^{d_e} = 1 \qquad \forall d \in D \tag{8}$$

Constraint (8) limits the provisioning of a VM to only one server in one of the available datacenters.

- ***Computational resources conservation***

$$y_{ser,c}^{d_e} . r_d^i \leq T_{ser,c}^i \qquad \forall d \in D, \forall i \in R, \forall c \in C \subseteq X, ser \in Ser \tag{9}$$

Constraint (9) deals with computational capacity during node provisioning. The virtual nodes must be provided with the required computational resources; thus, a VM or a destination node of a cloud connection request within the VN will be provisioned only to one server in one of the datacenters that has sufficient resources.

- ***Binary constraint***

$$\frac{1}{L} \sum_{c \in C \subseteq X} \sum_{ser \in Ser} y_{ser,c}^{d_e} \leq M_d \qquad \forall d \in D \tag{10}$$

$$\frac{1}{L} \sum_{u,v \in X} (f_{uv}^d + f_{vu}^d) \leq M_d \qquad \forall d \in D \tag{11}$$

Constraints (10) and (11) ensure that M_d is set to 1 whenever the destination virtual node within a VN is provisioned in one of the servers in any of the datacenters. Further, the virtual link is embedded into a substrate path.

F. Virtual Network Provisioning

This work adopts fixed and dynamic windowing techniques proposed in [18]. To the best of our knowledge, this is the first work that studies a proposed OVNP mathematical model that represents an actual distributed SDN-enabled cloud environment along with a dynamic windowing technique. Furthermore, the proposed approach is compared with the enhanced algorithm proposed in [18] as well as other methods in the literature.

F.1 OVNP Algorithm Input

The OVNP algorithm requires the following as input:

- The substrate network $G_p(X, E)$, which shows the topology,

characteristics, and utilization of the infrastructure. Here, the algorithm will have complete knowledge of the locations of the datacenters and cloud clients, available capacities of the servers, network topology, and state of the physical links in terms of the available bandwidth.

- The abstracted virtual network request $G_v(V, D, R, B)$. G_v defines the aggregated cloud connection requests (virtual links) to be provisioned simultaneously. Each cloud connection request (virtual link) of D has the following: virtual nodes V in terms of the source node (client node) and destination node (VM), requested computational capacities R of the destination virtual node, and networking resources B needed for the virtual link.

F.2 OVNP Algorithm Output

Given the OVNP formulation, the solver returns the following:

- The success of the cloud connection request (virtual link) provisions to the physical infrastructure using variable M_d.
- If a cloud connection request d of a virtual network G_v is provisioned, the solver indicates the server within a datacenter in which the destination node (virtual node) has been embedded, using variable $y^{d_e}_{ser,c}$. Such a server must have sufficient resources to accommodate that virtual node.
- The physical links and bandwidth required by the successful requests.

To successfully provision a virtual network, all of its virtual links and nodes must be provisioned to the physical infrastructure.

F.3 OVNP Algorithm Execution Process

Before the execution process of the algorithm can be delineated, it should be understood, as it is in [9] and [18], that cloud connection requests received from clients are accumulated in a fixed/dynamic time frame or window, and then processed together.

- Fixed window technique: This technique uses a predefined time window for a collection of virtual network requests. The analysis of cloud connection requests is carried out, and the requests sent by the same user are given the highest priority to be aggregated. The aggregated requests formulate VNs, and these VNs are then provisioned to the substrate network. The cloud clients can define maximum tardiness along with a connection request, and if the system is unable to serve the connection within this period, it is considered rejected. All unprovisioned requests are also rejected.
- Dynamic window technique: In this technique, requests are divided into a set of windows with different sizes using the maximum independent set algorithm. The algorithm takes the details of a predefined size set of cloud connection requests–such as arrival time, lifetime, and maximum waiting time–as input and constructs a binary table (called the intersection table) for connections. This table represents how these requests are overlapped over time. From this table, an interval graph is constructed. In this graph, each connection request is taken as a node; subsequently, the links connected to this node represent the cloud connection requests that intersect with this connection over time. At this point, the maximum independent set (MIS) algorithm is executed for the generated interval graph. The algorithm generates the maximum set of nodes (cloud connection requests) using the interval graph as input; during this process, no two nodes in the set are connected by a link. This set of nodes is called the maximum independent set of this particular interval graph. The boundaries of the dynamic time windows are determined using the requested connections' start times in each set. Note that this is done only once, independent of the total number of cloud connection requests received over time.

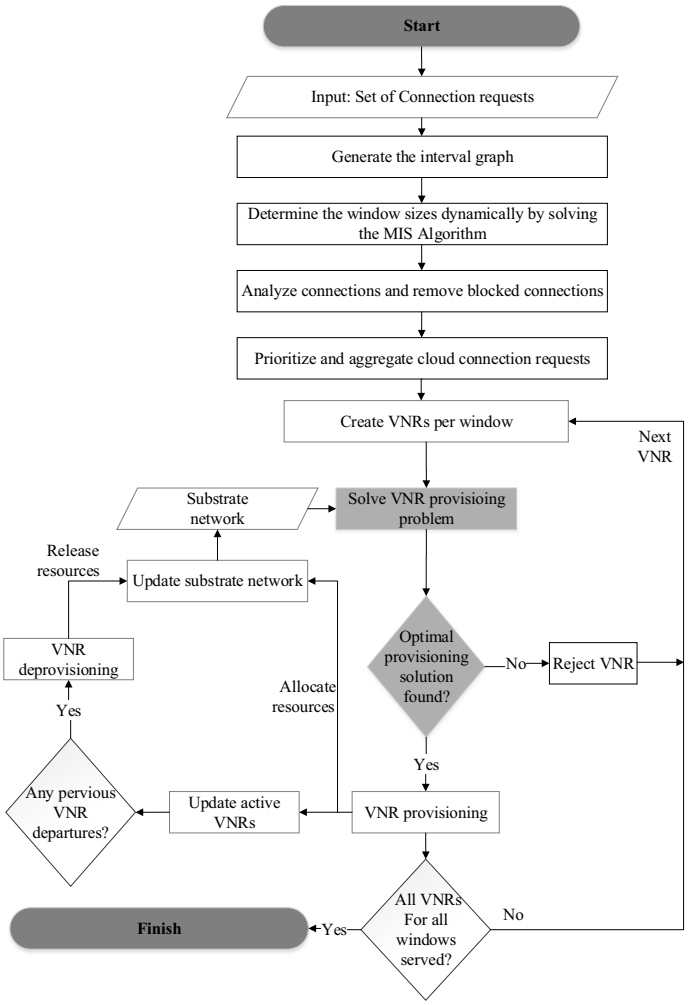

Fig. 2. Optimized virtual network provisioning process.

In the next step, the connections expiring before the end of the window assigned to them are set aside. The remaining connections in each window are stored according to their arrival time. Cloud connection requests within each dynamic window are aggregated into VNRs, taking into consideration that requests sent by the same user are given priority to be grouped together into one VNR.

Given the substrate network and the list of VNRs within a window the proposed OVNP model is solved for each VNR in order to assign the optimal provisioning of virtual nodes and links simultaneously to the substrate resources. Node and link provisioning is carried out in a single stage while checking the

current substrate network status. If the solution is not feasible, the VNR will be rejected, and the procedure will continue to the next VNR.

However, if a feasible solution is found, the central controller will provision it to the physical resources and update the substrate network status in terms of the computational resources and link bandwidth. Moreover, the set of active VNRs will be updated whenever a successful provisioning occurs, hence the number of served cloud connections is incremented based on the VNR's size. Furthermore, the algorithm checks if there are any departing VNRs. Whenever a VNR departs, it is de-provisioned, the resources will be released, and the network status will be updated accordingly. The algorithm will continue until all VNRs for all windows are served. The complete process is shown in Fig. 2.

F.4 Proposed Enhanced Network Cloud Provisioning

The network cloud provisioning (NCP) technique was proposed in [18]. The NCP process is divided into node and link provisioning. This work proposes an enhanced network cloud provisioning (ENCP) approach, which is designed to facilitate the process of node and link provisioning to enhance the coordination between the two stages. This approach increases the load balance and ensures wide distribution of load; the result is less congestion. The delay and request provisioning costs can be reduced by ensuring that clients' cloud requests are served and provisioned in the closest datacenter. VM provisioning in datacenter servers, which is represented by virtual nodes in virtual networks, is based on server resource availability. Selection of the correct datacenter/substrate nodes for provisioning is very important for node and link provisioning.

For node provisioning, an enhanced heuristic of the NCP, known as the region and load distribution based (RLDB) technique, was evaluated. In this technique, we divide the substrate network topology into multiple regions; each region has its own datacenter or set of servers. Whenever a virtual network request arrives at the central controller for provisioning, the central controller must optimize the virtual nodes within the physical network. It is important to note that substrate node selection is based on multiple factors. The average distance between the physical nodes is calculated and then arranged in descending order. Nodes with the highest average distance are preferred. For a virtual network request, the controller will select the nodes based on the following: highest average distance, source node region of the virtual network request (cloud client region/location), and the remaining capacities of the physical nodes. The algorithm will select the best of these three factors. The selected nodes must be within or close to the client region, must be far from previously selected provisioned nodes, and must have the maximum remaining capacities. The same heuristic illustrated in our previous work in [9] was utilized in the link provisioning stage.

As for this algorithm, the overall complexity can be schematized as follows:

1. Constructing the interval graph requires $O(m^2)$. This can be reduced to $O(m)$ for certain special connection set cases, where m is a predefined small size set of requests.
2. Finding the optimal dynamic window size by applying the maximum independent algorithm could require $O(m^8)$. However, this process will be performed only once for the previously defined set of connections as it is part of the preprocessing stage, and is independent of the size of any future connection requests set. It is worth mentioning that the complexity of the maximum independent set algorithm can be reduced to $O(m^5)$ for some special cases of graph topology. This is a potential improvement that we can develop in future works. Moreover, the produced dynamic window sizes can be used for different sets of connections regardless of their size, because the same window sizes can be repeated over time.
3. Constructing virtual SDN networks from the connections requires $O(n \log n)$ operations because it involves sorting of the requests received within each window, where n is the total number of cloud connection requests with $n >> m$.
4. The node provisioning process using the RLDB technique requires $O(n.NumPaths)$ operations. Here, $NumPaths$ is the number of paths between nodes (any two nodes u and v), and $Nodes$ is the number of nodes in the substrate graph.
5. The link provisioning process requires $O(n.Nodes)$ operations. Looking at the system overall, the dominant factor is the calculation of the maximum independent set, which is of $O(m^8)$ operations. However, because this is done offline and only once, it does not factor into the complexity calculation of the provisioning algorithm.

The provisioning process of our algorithm requires $O(n.Nodes + n.NumPaths)$ only. This accounts for the total complexity of our provisioning method. This is a polynomial running time and, hence, is tractable and acceptable. Note that this is also comparable to similar methods including [19], which requires $O((|Es`|(1 + |Ev|))^{3.5}L^2 lnLlnlnL)$ to serve a single virtual network or $O(n.(|Es`|(1 + |Ev|))^{3.5}L^2 lnLlnlnL)$ to serve all connections. Here, $Es`$ is the set of augmented substrate graph edges, Ev is the set of virtual links, and L is the set of physical links.

V. PERFORMANCE EVALUATION

This section describes the simulation environment and presents the evaluation results. A C++ based discrete event simulator was used to evaluate the proposed model. The simulator consists of the required modules to implement the functionalities illustrated in Fig. 2. These include modules for preprocessing, provisioning and producing the final results, and incorporating all the algorithms used in our experiments. For the optimal solution for the problem, the open-source GNU linear programming kit (GLPK) was used in the discrete event simulator. As GLPK supports the AMPL modeling language, the model and its data were compatible with the AMPL language. The efficiency of the proposed OVNP model was studied using simulation and its effectiveness was evaluated by comparisons with ENCP, NCP in [18], and other well-known approaches in the literature, such as the greedy multi-commodity flow problem (G-MCF) [24] and ViNEYard [19]. Different metrics were used in the evaluation process. To ensure the accuracy of the developed simulator, we reproduced the results of widely recognized approaches that have been applied in the literature; namely, [19]

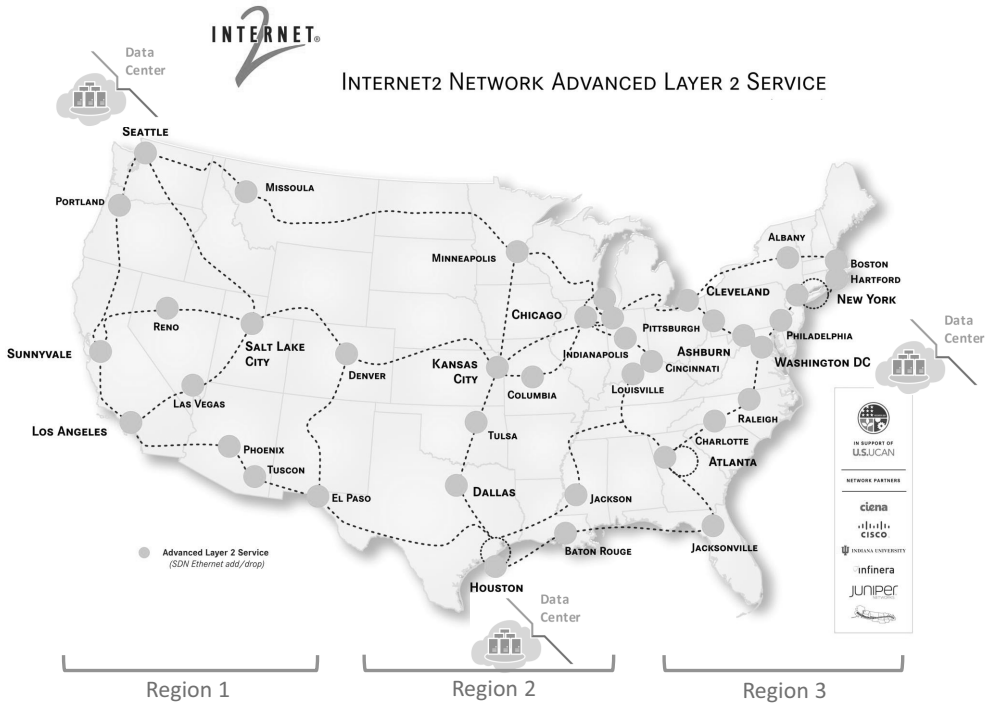

Fig. 3. Internet2 OS3E network topology.

and [24]. The reproduced results using our simulator matched the results reported in [19] and [24] and are shown in Fig. 4. The detailed analysis of this quantitative comparison is shown in the "Analysis of results" Section. Discrete event simulation techniques were applied, as well as appropriate simulation methodologies [47].

A. Simulation Setup

The physical network topology used in this work to evaluate the proposed model along with the heuristics is a well-known real-world network topology called Internet2 open science, scholarship and services exchange (OS3E) (`http://www.internet2.edu/network/ose/`), which is representative of a medium-scale infrastructure provider topology as shown in Fig. 3. OS3E was chosen because of its popularity in the research and education community; it is frequently used to support advanced global scientific research [40], [42], [43]. This network consists of a 34-node SDN (physical nodes can either be a datacenter or a client node), and 42 physical links. Internet2 is an OpenFlow enabled nationwide backbone. The topology is divided into three regions (east, middle, west) in a similar manner as the topology utilized by Amazon AWS, with one datacenter in each region (Seattle, Houston, and Washington, D.C.) considered to be one physical node. The considered regions are widely separated. Each region consists of one availability zone because it is assumed that the datacenters do not fail and thus ensure high availability.

Each datacenter has 400 servers; thus, $1,200$ different servers were applied in this environment. Substrate nodes were connected with 42 physical links. Each of the remaining 31 physical nodes can either generate cloud connection requests itself or route requests generated elsewhere. For each pair of nodes, three different paths were arranged. Further, $2,000$ different VMs were incorporated in the input data. The underlying assumption is that a maximum of $2,000$ VMs can be provisioned at any moment in time, owing to the arrival rate and service time of the requests.

Similarly, the simulation setup matches commonly used environments in the literature [14], [18], [19], [24], [26], [28]. The substrate link capacity was set to 200 bandwidth units (Mb/s). Clients' cloud connection requests arrive according to a Poisson process with a rate of 1–5 in increments of 0.5 per 100 time units, and each has an average lifetime of $1,000$ time units following an exponential distribution. Each experiment was run using $30,000$ cloud connection requests, and each one had a permitted waiting time of half the lifetime. Whenever a client requests a connection to a virtual machine, the client must determine its location (source node number), virtual machine specifications, connection start time, duration, requested bandwidth, and allocated waiting time. For the purpose of this work, and for alignment with the above-mentioned references, the source nodes are normally distributed in the range [0, 31], and the destination nodes representing the virtual machine numbers are uniformly distributed in the interval [1, 2000]. Specifically, 2,000 different VM instances were used in the simulation environment. It is worth mentioning that the number of VMs reflects the VM specification profiles provided by the service providers, which gives the cloud user more flexibility in choosing the desired profile according to his needs. Multiple VMs of the same profile can be provisioned for different cloud users. The capacities of the requested VMs were uniformly distributed. CPU requirements are in the range [0, 20], and memory and storage requirements are in the range [0, 20]. A uniform distribution within the range [0, 50] is set for the requested bandwidth. Fol-

lowing the simulation configuration used in [2], [9], and [26], the available resources per server are CPU, memory, and storage, and their capacities are uniformly distributed in the interval [50, 100] of their respective units.

The processor capacity represents the processing aptitude of a server in the million instructions per second (MIPs) gauge. This processing power measurement unit has been used in previous research (e.g., in [48] and [49]). Moreover, some cloud providers (such as Amazon [50], Azure [51]), as well as various researchers (such as [52]) use the number of cores offered in a machine to measure the computing power. However, in this work we used MIPs for this purpose because this measurement is more precise and suitable for optimization problems. It provides the ability to model tasks/requests with a higher level of granularity, instead of simply allowing the task to request the entire processing core. Memory (in MB), storage capacity (in GB), and bandwidth (in Mb/s) are also considered along with the processing power (MIPs).

It is assumed that any of the previous works can be used to determine the location of the SDN controller depending on the desired performance metric such as latency or fault tolerance [40]–[43]. Considering these metrics are outside the scope of this work, the placement of the controller was not considered. However, the model was built in a generic manner such that the objective metrics (ratio of served connection, computational and network resources utilization, etc.) were not significantly affected by the controller's placement.

B. Analysis of Results

We compared and evaluated the performance of the approaches proposed in [18]– i.e., NCP fixed window–as well as other approaches in the literature (namely, ViNEYard [19] and G-MCF [24]), against the optimal solution obtained in this study by the OVNP model and ENCP using the dynamic window technique. This comparison was made using different real-world performance metrics such as ratio of served connections (which represents the admission rate of the proposed model), provisioning average revenue, and the utilization of substrate resources.

B.1 Ratio of Served Connections

The first metric is the ratio of served connections. This metric represents the effectiveness of the proposed provisioning model. Reducing the number of rejected cloud service requests is the ultimate goal for any service provider. For different arrival rates, we studied the ratio of served cloud connection requests, as shown in Fig. 4. The proposed optimal model outperforms the other approaches at low and high arrival rates. The optimal solution obtained by the OVNP model exhibits stable and efficient provisioning, even with a loaded substrate network. The single-stage node and link provisioning model provide excellent performance. With a crowded network, the OVNP model tended to have an approximately 57% higher ratio of served connections.

As the number of served connections increases, the overall throughput is positively influenced. In terms of the service rate and other QoS metrics, we consider only cases where the virtual network request requirements are completely satisfied. This implies that a request is either served with the exact requirements for the lifetime specified, or the request is blocked. There is no fluctuation or degradation in the service, because the resources allocated to a request are not released until the connection lifetime is over.

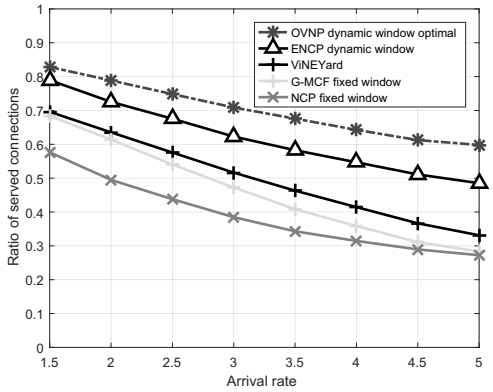

Fig. 4. Ratio of served connections in OS3E network topology for $30,000$ cloud connections.

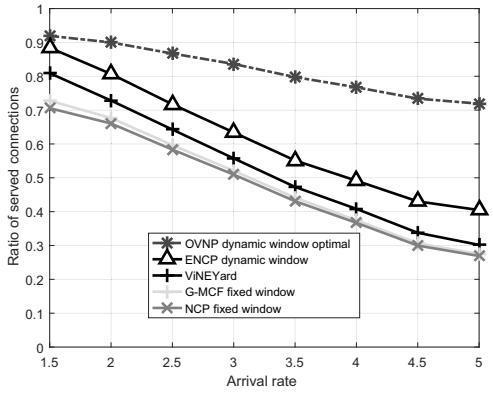

Fig. 5. Ratio of served connections in NSFNET network topology for $3,000$ cloud connections [9].

B.2 Scalability

To evaluate the scalability of the proposed model, all the considered algorithms were simulated using the same setup adopted in [9], which consisted of 14 nodes, 3,000 cloud connection requests, and 200 VMs. Figs. 4 and 5 show the ratio of served connections for the simulation setup presented in section V-A and the one considered in [9], respectively. The number of cloud connection requests was increased from 3,000 to 30,000. This number can be increased further by expanding the time frame considered. However, this will not affect performance, owing to the online nature of the problem. This is evident through the observation that similar trends appear at both simulation scales. Even with the larger, more realistic scale considered in this study, the proposed OVNP algorithm outperforms other works presented in the literature. This shows the scalability of the simulation environment – the results remained consistent even with the larger number of physical nodes, connection requests, and VMs.

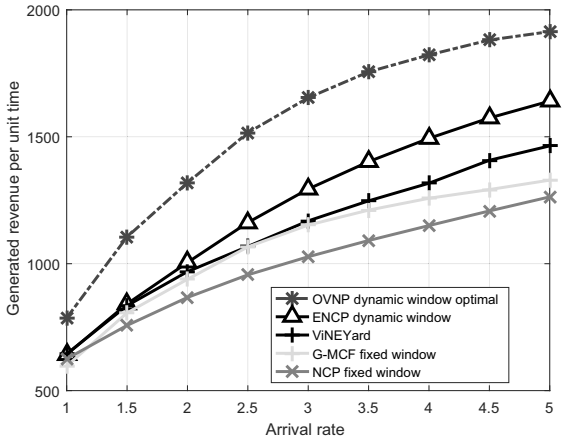

Fig. 6. Average revenue per unit time for OS3E network topology.

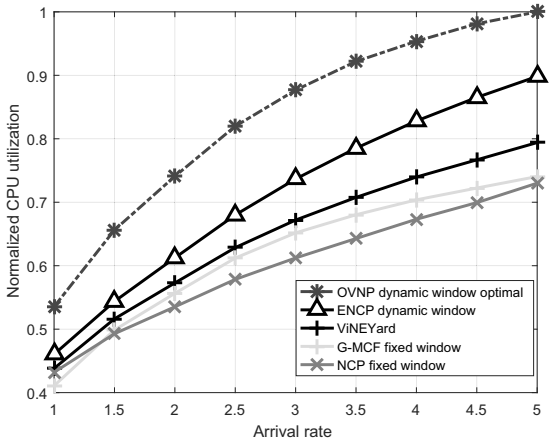

Fig. 7. Normalized CPU utilization.

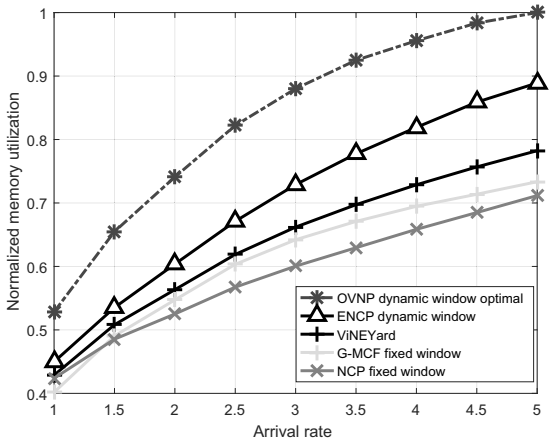

Fig. 8. Normalized memory utilization.

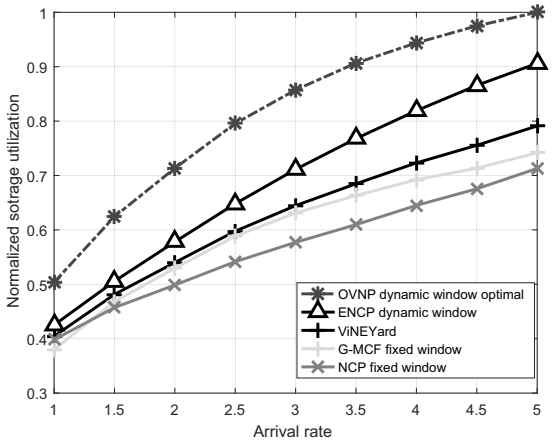

Fig. 9. Normalized storage utilization.

B.3 Generated Profit per Unit Time

The second metric – namely, generated profit per time unit – is of great importance to cloud service providers. The revenue is defined similarly to the previous work in [14], [18], and [24]. Serving a high number of client requests does not necessarily lead to high revenue; rather, high revenue results from serving a higher number of requests with fewer requested resources and a shorter duration. This can be seen in Figs. 4 and 6 when we compare the NCP fixed window and G-MCF fixed window. The G-MCF fixed window tends to serve slightly more cloud requests at low and high arrival rates than the NCP fixed window; however, the generated revenue for NCP at low arrival rates is higher than that for G-MCF (as shown for arrival rate $= 1$). In contrast, our optimal OVNP model has a higher ratio of served connections and higher revenue at low and high arrival rates. The proposed optimal algorithm outperforms the other approaches in terms of generated profit per time unit.

B.4 Resource Utilization

The third metric is resource utilization. We considered CPU, memory, storage, and link bandwidth for this purpose, as shown in Figs. 7–10. These figures compare the resource utilization of the proposed optimal OVNP model with that of the four methods presented above. Note that the OVNP model has the best computational resource utilization, including optimal CPU power, memory, and storage.

Fig. 10 shows the average link utilization for the different algorithms. Several observations can be made. First, G-MCF has high link utilization, which is not desirable as this can lead to more rejected requests arriving in future timeframes. This is verified by the low acceptance ratio of the G-MCF algorithm, showing that the link provisioning process is not efficient. Second, all algorithms have an average link utilization of approximately 52%. This further proves the effectiveness of the proposed OVNP model, as it served a significantly higher ratio of connections while having similar link utilization to other algorithms proposed in the literature. Note, the links connected to datacenters had higher link utilization levels than other links in the network.

B.5 Average Hop Count per Virtual Link

Fig. 11 shows the average hop count per virtual link for the different algorithms. The figure shows that OVNP outperforms the other algorithms as it has the lowest average hop count. This leads to lower latency as well as lower cost because the virtual link is mapped on a smaller number of physical links. This is the result of the node mapping being optimal.

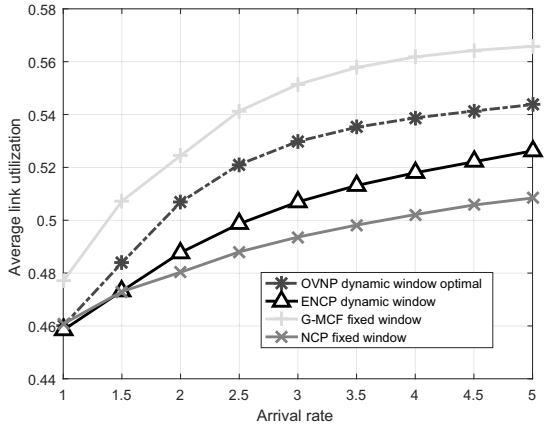

Fig. 10. Average link utilization.

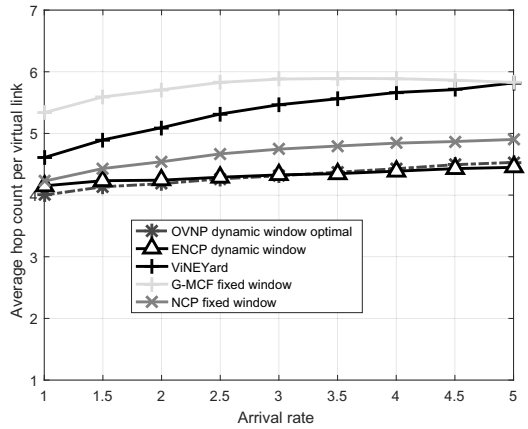

Fig. 11. Average hop count per virtual link.

VI. SUMMARY OF CONTRIBUTION

The models described in the related work section deal with the problem of virtual network provisioning; however, all of the previous approaches, available in the literature, relaxed one or more of the challenges or environment constraints imposed on the virtual network provisioning process. These flexible approaches were taken to reduce the complexity/search space of the problem. The search space dimensions are three-fold, namely the bandwidth, computational resources, and the time dimension. The effect of the constraints studied falls within these three dimensions. The constraints considered for achieving an optimal solution are node and link requirement constraints, the online nature of request arrivals, diversity of virtual network topologies, and limited physical computational and network resources, imposing challenges on the provisioning process. These constraints better model real life scenarios and network characteristics.

- The node and link requirement constraints directly affect the virtualization process. This is because joint provisioning of both the node and links simultaneously can improve the provisioning efficiency. In contrast, performing the node and link provisioning in two stages can reduce the efficiency since the virtual network would lose some of its flexibility when it is provisioned.
- The online nature of requests complicates the virtualization design as it increases the size of the search space. Moreover, the unpredictable nature of the statistics of incoming VNRs forces the system to solve the provisioning problem continuously. On the other hand, if the VNRs are assumed to be known in advance (i.e., offline), the search space is reduced since the controller knows all the VNRs that need to be provisioned a priori and hence can more efficiently map them onto the substrate network.
- The diversity of virtual network topology constraint provides the controller with more options to provision the VNRs. If a specific virtual network topology is assumed, this can reduce the utilization efficiency of the available resources since the topology is not flexible. However, this work assumes a flow metrics-based approach which can better "compress" the VNRs onto the substrate. This leads to improved admission control performance.
- The limited physical computational and network resource constraints limit the number of VNRs that are provisioned onto the substrate network. If the computational resources were infinite, more virtual nodes can be mapped onto the physical nodes. Similarly, if the network resource (i.e., the bandwidth) is infinite, a greater number of virtual links can be mapped to the physical link. Hence, the performance would be exaggerated with such assumptions (infinite resources).

  Each one of the previous models, which relaxed one or more of these constraints, has been analyzed in the related work section. The relaxation was in attempt to reduce the search space size and hence the complexity of solving the problem. However, this study addressed all of these challenges and environment constraints, and introduced the most complete problem configuration.

To the best of our knowledge, this is the first work that tackles all these constrains in totality. Moreover, it is the first to address these problems via the dynamic windowing methodology as in [18], along with mathematical modeling that accounts for one-stage provisioning of SDN-enabled virtual networks in geo-distributed cloud computing datacenters.

The contributions of this paper can be summarized as follows.

- Comprehensively formulating the problem of virtual network provisioning in SDN-enabled geographically distributed cloud datacenters as a MILP problem. This is done with the objective of maximizing the ratio of the served cloud client connection requests while simultaneously minimizing the resources used during the provisioning process. In our work, SDN-enabled virtual networks comprise aggregated cloud connection requests.
- Formulating the problem of virtual machine provisioning to cloud datacenters and cloud connection request provisioning to substrate network in a single stage.
- Studying the time complexity of the proposed ENCP heuristic and comparing it with that of previous works.
- Comparing the proposed OVNP model with well-known approaches in the literature (as well as the ENCP model) via simulation, and verifying that our approach demonstrates the best performance capabilities by using various real-world metrics.

VII. CONCLUSION

We modeled the optimization problem of maximizing the ratio of served cloud connection requests while minimizing the cost of provisioning these requests in SDN-enabled geo-distributed cloud computing datacenters as a MILP problem. To solve the problem optimally, the open-source GNU linear programming kit (GLPK) was employed in a discrete event simulator. Node and link provisioning was solved in a single stage. We evaluated the proposed model against four other approaches; numerical results showed that our OVNP model achieved a higher ratio of served connections, higher profits, higher computational resource utilization, and lower average hop count per virtual link.

REFERENCES

[1] A. Fox *et al.*, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, p. 13, 2009.

[2] M. A. Sharkh, A. Ouda, and A. Shami, "A resource scheduling model for cloud computing data centers," in *Proc. IWCMC*, July 2013, pp. 213–218.

[3] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.,*, vol. 47, no. 7, pp. 20–26, July 2009.

[4] F. Hao, T. Lakshman, S. Mukherjee, and H. Song, "Enhancing dynamic cloud-based services using network virtualization," in *Proc. ACM VISA*, 2009, pp. 37–44.

[5] B. Wanis, N. Samaan, and A. Karmouch, "Efficient modeling and demand allocation for differentiated cloud virtual-network as-a service offerings," *IEEE Trans. Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[6] O. N. O. S. (ONOS), "Driving sdn adoption in service provider networks," 2014. [Online]. Available: http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-Service-Provider-SDN-final.pdf

[7] S. Jamalian and H. Rajaei, "Data-intensive hpc tasks scheduling with sdn to enable hpc-as-a-service," in *Proc. IEEE CLOUD*, June 2015, pp. 596–603.

[8] C. Chappell, "Unlocking network value: Service innovation in the era of sdn," *HeavyReading, white paper*, 2013.

[9] K. Alhazmi, M. Abusharkh, D. Ban, and A. Shami, "A map of the clouds: Virtual network mapping in cloud computing data centers," in *Proc. IEEE CCECE*, May 2014, pp. 1–6.

[10] C. Papagianni, G. Androulidakis, and S. Papavassiliou, "Virtual topology mapping in sdn-enabled clouds," in *Proc. IEEE NCCA*, Feb. 2014, pp. 62–67.

[11] M. Melo *et al.*, "Virtual network mapping–an optimization problem," *Mobile Networks and Management*, Springer, 2012, pp. 187–200.

[12] M. Melo, J. Carapinha, S. Sargento, U. Killat, and A. Timm-Giel, "A re-optimization approach for virtual network embedding," *Mobile Networks and Management*, Springer, 2013, pp. 271–283.

[13] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Trans. Network and Service Management*, vol. 10, no. 4, pp. 356–368, Dec. 2013.

[14] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[15] R. Esteves, L. Granville, and R. Boutaba, "On the management of virtual networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 80–88, July 2013.

[16] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[17] C. Wang and T. Wolf, "Virtual network mapping with traffic matrices," in *Proc. ACM/IEEE ANCS*, Oct. 2011, pp. 225–226.

[18] K. Alhazmi, M. Sharkh, and A. Shami, "Drawing the cloud map: Virtual network provisioning in distributed cloud computing data centers," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2016.

[19] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Networking*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[20] W.-H. Hsu, Y.-P. Shieh, C.-H. Wang, and S.-C. Yeh, "Virtual network mapping through path splitting and migration," in *Proc. WAINA*, Mar. 2012, pp. 1095–1100.

[21] S. Shanbhag, A. R. Kandoor, C. Wang, R. Mettu, and T. Wolf, "Vhub: Single-stage virtual network mapping through hub location," *Computer Networks*, vol. 77, pp. 169–180, 2015.

[22] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011–1023, 2011.

[23] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM VISA*, 2009.

[24] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[25] I. Barla, D. Schupke, M. Hoffmann, and G. Carle, "Optimal design of virtual networks for resilient cloud services," in *Proc. DRCN*, Mar. 2013, pp. 218–225.

[26] C. Papagianni *et al.*, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Computers*, vol. 62, no. 6, pp. 1060–1071, June 2013.

[27] I. Fajjari, N. Aitsaadi, M. Pióro, and G. Pujolle, "A new virtual network static embedding strategy within the clouds private backbone network," *Computer Networks*, vol. 62, pp. 69–88, 2014.

[28] G. Sun, V. Anand, H.-F. Yu, D. Liao, and L. Li, "Optimal provisioning for elastic service oriented virtual network request in cloud computing," in *Proc. IEEE GLOBECOM*, Dec. 2012, pp. 2517–2522.

[29] S. Ayoubi, C. Assi, K. Shaban, and L. Narayanan, "Minted: Multicast virtual network embedding in cloud data centers with delay constraints," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1291–1305, Apr. 2015.

[30] M. Bui, T. Wang, B. Jaumard, D. Medhi, and C. Develder, "Time-varying resilient virtual network mapping for multi-location cloud data centers," in *Proc. ICTON*, July 2014, pp. 1–8.

[31] R. Trivisonno *et al.*, "Virtual links mapping in future sdn-enabled networks," in *Proc. IEEE SDN4FNS*, Nov. 2013, pp. 1–5.

[32] C. Papagianni, G. Androulidakis, and S. Papavassiliou, "Virtual topology mapping in sdn-enabled clouds," in *Proc. IEEE NCCA*, Feb. 2014, pp. 62–67.

[33] R. Guerzoni *et al.*, "A novel approach to virtual networks embedding for sdn management and orchestration," in *Proc. IEEE NOMS*, May 2014, pp. 1–7.

[34] Z. Wang, J. Wu, Y. Wang, N. Qi, and J. Lan, "Survivable virtual network mapping using optimal backup topology in virtualized sdn," *Communications, China*, vol. 11, no. 2, pp. 26–37, Feb. 2014.

[35] R. Sherwood *et al.*, "Can the production network be the testbed?" in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.

[36] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[37] Z. Bozakov and P. Papadimitriou, "Autoslice: Automated and scalable slicing for software-defined networks," in *Proc. ACM CoNEXT*, New York, USA, 2012, pp. 3–4.

[38] Nicira, "Its time to virtualize the network," 2012, [Online]. Available: http://nicira.com/en/network-virtualization-platform

[39] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

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

[41] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Let.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.

[42] D. Hock *et al.*, "Pareto-optimal resilient controller placement in sdn-based core networks," in *Proc. ITC*, Sept. 2013, pp. 1–9.

[43] M. Soliman, B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Source routed forwarding with software defined control, considerations and implications," in *Proc. ACM CoNEXT*, New York, USA, 2012, pp. 43–44.

[44] A. Yassine, H. Rahimi, and S. Shirmohammadi, "Software defined network traffic measurement: Current trends and challenges," *IEEE Instrum. Meas. Mag.*, vol. 18, no. 2, pp. 42–50, Apr. 2015.

[45] C. Yu *et al.*, "Flowsense: Monitoring network utilization with zero measurement cost," in *Proc. PAM*, Hong Kong, China, 2013, pp. 31–41.

[46] L. Nacshon, R. Puzis, and P. Zilberman, "Floware: Balanced flow monitoring in software defined networks," *CoRR*, vol. abs/1608.03307, 2016. [Online]. Available: http://arxiv.org/abs/1608.03307

[47] J. Banks, J. Carson, and B. Nelson, *DM Nicol, Discrete-Event System Simulation*, Prentice hall Englewood Cliffs, NJ, USA, 2000.

[48] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," 2009.

[49] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid com-

puting," *Concurrency and computation: Practice and experience*, vol. 14, no. 13–15, pp. 1175–1220, 2002.

[50] S. Ostermann *et al.* "A performance analysis of EC2 cloud computing services for scientific computing," in *Proc. IEEE CLOUD*, Springer, Berlin, Heidelberg, 2010, vol. 34, pp. 115–131.

[51] D. Chappell, "Introducing the Azure services platform," *White paper,* , vol. 1364, no. 11, Oct. 2008.

[52] M. D. De Assunçao, A. Di Costanzo, and R. Buyya, "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters," in *Proc. ACM HPDC*, 2009, pp. 141–150.

Khaled Alhazmi received his B.Sc. degree in Computer Engineering from King Saud University in 2007. Since then he has worked as a Senior R&D Researcher, Project Manager, and Senior Engineer at King Abullaziz City for Science and Technology-KACST. He received his ME.Sc. degree in Electrical and Computer Engineering from Western University in 2014. Since September 2014, he has been pursuing his Ph.D. studies at Western University, London, Ontario, Canada. Since September 2012, he has been with optimized communication and computations (OC2Lab) group at Western University. His current research interests are in the areas of cloud computing, virtualization, cloud computing optimization management and service provisioning, software defined networking, and software engineering.

Abdallah Shami received his B.E. degree in Electrical and Computer Engineering from the Lebanese University in 1997 and his Ph.D. degree in Electrical Engineering from the Graduate School and University Center, City University of New York, in September 2002. In September 2002, he joined the Department of Electrical Engineering at Lakehead University, Thunder Bay, Ontario, Canada as an Assistant Professor. Since July 2004, he has been with Western University, where he is currently a Professor in the Department of Electrical and Computer Engineering. His current research interests are in the areas of network optimization, cloud computing, and wireless networks.

Ahmed Refaey Hussein, Ph.D., P.Eng., SMIEEE, is an Assistant Professor at Manhattan College as well as an adjunct research professor at The University of Western Ontario. Previously, his positions included: A Sr. Systems Architect, R&D group, Mircom Group of Companies since 2013–2016; and as a Postdoctoral Fellow at ECE department, The University of Western Ontario since 2012–2013; and Professional Researcher at the LRTS lab, Laval University in the field of wireless communications since 2007–2011. Prior to joining Laval University, he was a System/ Core Network Engineer leading a team of junior engineers and technicians in the telecom field in the three large companies of Fujitsu, Vodafone and Alcatel-Lucent. He received his B.Sc. and M.Sc. degrees from Alexandria University, Egypt in 2003 and 2005,respectively; and Ph.D. degree from Laval University, Quebec, Canada in 2011.