Journal of Network and Computer Applications 162 (2020) 102639

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Multi-domain virtual network embedding with dynamic flow migration in software-defined networks

Dipanwita Thakur [a,*], Manas Khatua [b]

[a] Department of Computer Science and Engineering, Banasthali Vidyapith, Rajasthan, India
[b] Department of Computer Science and Engineering, IIT Guwahati, Assam, India

ARTICLE INFO

Keywords:
Virtual network embedding
Software defined networking
Multi-domain SDN
Cellular learning automata

ABSTRACT

Software Defined Networking (SDN) decouples the control and data planes of a network, and employs a central controller to provide effective use of network resources and ease of service provision. Virtualization in SDN empowers the virtual SDN networks to share common physical network infrastructures, and, thus, make them capable for versatile service provisioning. In such setting, SDN hypervisor supports multiple virtual SDN network (vSDN) in which each vSDN has its own controller. To create virtual networks and achieving optimal sharing of physical network resources, virtual network embedding (VNE) algorithms are designed. Single domain VNE is a well-studied problem in NV literature. However, in most of the real life scenarios, VNs are provisioned across heterogeneous administrative domains belonging to different Infrastructure Providers (InPs).

In this paper, we propose a VNE algorithm, namely *vSDN-CLA*, using Irregular Cellular Learning Automata (ICLA) for multi-domain SDN networks. We consider two aspects – optimal mapping of virtual nodes and links into multi-domain substrate network, and optimal placement of SDN controller with respect to throughput and end-to-end delay. We extend the vSDN-CLA by considering dynamic flow migration to achieve resource optimal routing. We evaluate the proposed schemes using Mininet. We observe that the proposed schemes outperform the existing benchmark schemes with respect to throughput, end-to-end delay, and virtual network request acceptance ratio under both the single and multi-domain environments.

1. Introduction

The key idea behind SDN is decoupling network control and management functionalities from data forwarding operations to enable a centralized control platform for supporting network programmability of data center and network functionalities (Kreutz et al., 2015; Yang et al., 2015, 2016). In SDN environment, network applications can communicate to network switches and reconfigure network resources to fulfill their needs. However, control and data plane isolation is not sufficient to satisfy the evolving service requirements. NV is being researched with the rapid expansion of cloud computing diverse services and high-bitrate services to solve the ossification of current network architecture (Luo et al., 2014). Further, in many case, services are heterogeneous in nature. It is therefore challenging to satisfy the requirements of every service using single physical network architecture.

On the other hand, multiple physical networks together create a heterogeneous network architecture which in turn increases the complexity of network installation and maintenance. In such immensely complex environments the probability of control data such as delay, packet loss rate, and transmission rate loss is very high, which leads to violation of QoS requirements. To overcome this trade-off, a heterogeneous network architecture created on a multi-domain physical network is essential. In this regard, NV plays important role by providing overlay on multiple physical network. Therefore, virtualization of SDN networks combines the benefit of both SDN and NV (Blenk et al., 2016). During the creation of VNs, it is required to map the VNs onto underlying multiple physical networks. This process of mapping is called multi-domain VNE (Fischer et al., 2013).

1.1. Motivation

The existing virtualization technologies (e.g. hypervisor-based and container-based virtualization) support multi-domain virtual network to create and share common SDN infrastructure. However, multi-domain VNE is not a trivial task. One of the main goals of VNE is

* Corresponding author.
E-mail addresses: tdipanwita@banasthali.ac.in (D. Thakur), manaskhatua@iitg.ac.in (M. Khatua).

https://doi.org/10.1016/j.jnca.2020.102639
Received 4 September 2019; Received in revised form 9 February 2020; Accepted 20 March 2020
Available online 13 April 2020

to satisfy maximum number of virtual network requests (VNRs). The arrival time and resource requirements of the VNRs are unknown at inception. The VNE algorithm produces efficient mapping solution for the VNRs within an acceptable time period. For the following reasons, setting up a multi-domain VN is more challenging than one on a single domain.

- Researchers solved various single domain VNE problems using Integer Linear Programming (ILP) (Gong et al., 2016; Demicri and Ammar, 2014). It assumes complete knowledge of physical resources and the underlying substrate topology, which is difficult to perceive in multi-domain architecture. Multi-domain substrate network is provided by separate InPs as each domain may have different administrator. InPs are unwilling to disclose any domain topology and resource information to VN providers (Chowdhury et al., 2009; Yu et al., 2008; Zhu and Ammar, 2006). A multi-domain VNE problem can be considered as a single large domain VNE problem if we have the optimal complete view of all the different domains. So, multi-domain VNE problem is computationally difficult to solve due to NP-completeness of mapping (Samuel et al., 2013). Though, previously very few researchers addressed the multi-domain VNE problem (Xiao et al., 2017; Li et al., 2017; Zhou et al., 2014). However, none of the previous works addressed the dynamic placement of the controller in inter-domain as well as intra-domain environments and also the flow-migration issue.
- On the other hand, due to business privacy policy, one domain operator is totally unaware of another domain to protect the domain topology and resource information. So optimal and complete view of all the domains is not at all possible and costly in real scenario.
- Due to the limitation of the processing power of a single controller such as latency resulting from distant network devices and an enormous quantity of overhead due to messaging between controller and physical devices, single controller solution is not scalable for large scale multi-domain networks (Wibowo et al., 2017). Therefore, there is a need for a multi-controller distributed control plane so that each controller is responsible for a single domain of the physical network. Therefore, controller-to-controller synchronization is required to get the complete view of the whole network. Mapping in between the control plane and the data plane must be automated, so that the dynamic changes in the network will not deteriorate the overall performance of the network.

To address the aforementioned challenges, multi-domain VNE is required with proper placement of the controller.

1.2. Contribution

In this paper, we provide a multi-domain VNE algorithm, namely vSDN-CLA, based on ICLA. In vSDN-CLA, a slice of substrate network is created which is mapped into a heterogeneous multi-domain network through the interactions with both the global and/or local environments. Multi-Domain VNE problem can also be solved using non-learning algorithms. However the ICLA is useful for multi-domain VNE design as it supports changeable node neighbors i.e. neighbors of a node can not be fixed forever, arbitrary node placement and ability to observe the performance of nodes during packet forwarding (Aghababa et al., 2012). Further, the ICLA based model has the capability to perform efficient and robust complicated estimation using adaptive decision making strategy which operates on unknown random environments. Moreover, the natural system model in which huge collections of simple objects those are interacting with each other are specially designed using ICLA. As VNE is a multi-objective optimization problem in dynamically changing network, we observe that the learning automaton (LA) ascertains the action with least penalty likelihood and ultimately selects optimal action recurrently.

The vSDN-CLA scheme runs in two phases – (i) co-ordinated mapping of node and link (ii) controller placement. In the first step, it establishes ICLA to map the virtual nodes and links to physical nodes and links, respectively, based upon the action performed by the LA of each and every physical node. In the latter stage, it realizes the optimal position of the controller nodes based upon the action such as accepted or rejected, performed by the LA in each vSDN. In this process, each node learns about optimal action using local rules and global rules, and, in turn, the controller-to-switch delay is minimized. After the completion of the VNE, the nodes in VNs send packets through the established path. If the path is clogged with traffic volume, the VNs are not able to find an alternate path as the traffic flow rules can not be changed dynamically. Therefore, it demands some kind of network programmability for modifying the traffic flow rules. Hence, we extend the vSDN-CLA scheme by proposing a dynamic flow migration algorithm, namely FM-vSDN, to accommodate new flows in the flow table of a node. In this method, source to destination flow migration is performed at runtime without affecting the ongoing flows. It is pertinent to mention that this paper is an extended version of our prior work (Thakur and Khatua, 2018). In the present version, we have added (i) flow migration, (ii) theoretical analysis of the solution, (iii) new set of experimental results exploring the average utilization of node and link resource, and average acceptance ratio of the virtual network requests. In brief, we summarize our contributions as follows:

- We formulate the multi-domain VNE problem as the combination of two distinct problems – VNE on common substrate network and optimal placement of the controller. We design an algorithm, namely 'vSDN-CLA', for solving the multi-domain VNE problem.
- We further design a dynamic flow migration algorithm, namely 'FM-vSDN', to achieve resource optimal routing in multi-domain SDN networks.
- We perform simulation on Mininet emulator to analyse the performances of 'vSDN-CLA' and 'FM-vSDN' schemes, and compare their performances with benchmark schemes.

2. Related work

In this section, we outline some research advancements in the area of VNE (Fischer et al., 2013) on SDN (Kreutz et al., 2015; Blenk et al., 2016). A few heuristic VNE algorithm such as (Sun et al., 2013; Rahman and Boutaba, 2013) were proposed earlier in traditional network. A novel cooperative virtual network embedding algorithm for network virtualization has been proposed by Feng et al. (2018), that coordinated the advantages of the centralized and distributed algorithms. They have used the "Bloom Filter" concept to synchronize the mapping information within the substrate network to reduce the massive communication overhead as flooding. They have provided the static solution without considering varying and dynamic virtual network requests. Haeri et al. (Haeri and Trajkovi, 2017), implemented the virtual node embedding algorithm using "Markov decision process" framework and the node mapping policies for the proposed approach is formalized by the "Monte Carlo Tree search algorithm". They also solved the virtual link mapping problem using multi-commodity flow and shortest path strategies. However, they did not consider the re-embedding concept of the virtual network. However, all the prior works mentioned above did not consider the SDN environment. On the other hand, researchers are continuously trying to take advantage of SDN in different aspect of networking including VNE. A SUDOI architecture was proposed to enhance multiple layer resource integration in intra-data-center and inter-data-center optical interconnection (Yang et al., 2016). Another virtual optical network embedding algorithm in elastic optical networks was proposed by (Luo et al., 2014). To enhance the resource utilization, multi-stratum resource integration method is proposed by (Yang et al., 2015). The proposed method was based on network function virtualization in software defined elastic data center optical interconnect. A VNE algorithm in SDN environment is proposed in (Feng et al., 2014) by considering the price of each resource, fair allocation of bandwidth,

and flow table for multiple control applications to address the above mentioned problem. However, the evaluation of this work is done using the private dataset provided by an organization and compared with a random scheduling model. A SDN-based virtualized network has been considered for flow-migration based resource management in (Mijumbi et al., 2014). The authors enhanced the Floodlight SDN controller with a resource manager and a database. The authors evaluated the proposal based on the acceptance ratio, average link and switch resource utilization, and virtual network mapping cost. However, they did not consider the problem of controller selection in SDN-based network. Tuncer et al. (2015) proposed a SDN-based framework for fixed backbone network which provides support for both static and dynamic resource management applications. The authors considered the selection of SDN controller to manage resources. However, they did not considered the network virtualization for better resource management. Zhou et al. (2014) proposed a multi-domain virtual network embedding strategy for SDN to improve the performance with respect to scalability. However, the aforementioned work on virtual network mapping in SDN neither considered the controller selection problem nor flow entry resource allocation in node mapping.

Gong et al. (2016) proposed a novel heuristic, online virtual SDN mapping algorithm using ILP, namely Co-vSDN, to minimize the embedding cost and controller-to-switch delay. Considering the selection of SDN controller, Demicri and Ammar (2014) designed virtual link and node mapping algorithm, namely DM-vSDN, to balance the node of the substrate network, and to minimize the controller to switch delay. However, the proposed scheme did not provide any coordination in between controller selection and node mapping followed by link mapping. Zhong et al. (2016) proposed FlowVisor-based virtual network embedding algorithm in OpenFlow networks in cost-efficient way. They have considered the controller placement problem to minimize controller-to-switch delay. Zhu et al. (2017) emphasized on the delay constrained virtual SDN embedding problem, considering placement of the controller. They believed that both the controller placement and the virtual network embedding are strictly restricted by controller-to-switch delay constrained. In their analysis they predicted that as the delay increased the acceptance ratio of the requested virtual network decreased.

Few existing works (e.g. (Heller et al., 2012; Hu et al., 2014; Jiménez et al., 2014; Rath et al., 2014; Lange et al., 2015; Zhu et al., 2017; Killi and Rao, 2018)) addressed the controller placement issue. However, the objective of those work were to control the substrate network instead of VNs and thus they did not consider the computing resources. On the other hand, in virtualized SDN environment, each VN has its own controller. Therefore, it is important to administer computing resources. Recently, Mostafaei et al. (2018) proposed a controller placement algorithm based on learning automation known as LACP. This algorithm is based on clustering of network and LA. However, the authors did not consider the states or actions of the neighbor nodes to update the state and action of a specific LA. Further, they did not consider flow migration in VNE to reconfigure the vSDN. Moreover, the above mentioned works were not designed for the multi-domain substrate network.

A few flow migration methods such as (Ghorbani et al., 2014; Mijumbi et al., 2014) and (Chen et al., 2019) were proposed in SDN environment but not in multi-domain SDN. Few works such as (Xiao et al., 2017), (Li et al., 2017), (Guo et al., 2015), (Dietrich et al., 2013) and (Zhou et al., 2014) proposed multi-domain VNE for SDN without addressing the issues of controller placement and flow migration.

According to the best of our knowledge, VNE, controller placement and flow-migration together in multi-domain SDN environment is a novel work. All the state-of-the-art algorithms cover at most any of the two. These gaps open up the scope for studying multi-domain VNE in SDN, while considering both the controller placement and flow migration issues.

3. Preliminaries

Before proceeding to the detailed discussion of problem description and proposed schemes, we discuss about pre-requisite knowledge on ICLA in this section.

3.1. Learning automata

In a random unknown environment, an adaptive decision-making method is used, known as Learning Automata (LA) (Thathachar and Sastry, 2002). The structure of an LA consists of four tuples: $LA = \{\alpha, \beta, p, T\}$, where.

- $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the action set of LA with r number of actions,
- $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the input set of reward and penalty,
- $p = \{p_1, p_2, \ldots, p_r\}$ represents the action probability set,
- $p(t+1) = T[\alpha(t), \beta(t), p(t)]$ represents the learning algorithm.

Let at time t, α_i is the action chosen by the automata. The action probability p is updated based on the following recurrence equations after receiving the environment's reinforcement signal $\beta(t)$:

$$\left.\begin{aligned} p_i(t+1) &= p_i(t) + a(1-p_i(t)) \\ p_j(t+1) &= p_j(t) + a(1-p_j(t)) \quad \forall j, j \neq i \end{aligned}\right\} \tag{1}$$

for the favourable responses (Thathachar and Sastry, 2002), and

$$\left.\begin{aligned} p_i(t+1) &= (1-b)p_i(t) \\ p_i(t+1) &= \frac{b}{(r-1)} + (1-b)p_j(t) \quad \forall j, j \neq i \end{aligned}\right\} \tag{2}$$

for the unfavorable responses (Thathachar and Sastry, 2002). The parameters a and b represent the reward and penalty values, respectively. Based on the values of a and b, we get different types of LA. If $a = b$, the LA is known as linear reward-penalty (L_{R-P}). If $a >> b$, the LA is known as linear reward- ϵ penalty ($L_{R\epsilon P}$). If $b = 0$, the LA is known as linear reward-inaction (L_{R-I}).

3.2. Cellular Automata

Cellular Automata (CA) is a numerical model to represent a large system comprising of a collection of cells arranged in a grid. The operation of the CA is based on the local rules of the neighbor cells. The CA is capable to converge to a fixed point with interesting properties, which are quantized by an objective function.

3.3. Cellular learning automata

A Cellular Learning Automaton (CLA) (Esnaashari and Meybodi, 2015) is the combination of CA and LA. For many decentralized phenomena, CLA is a powerful mathematical model. The distributed computation capability of CA and learning capability of LA in unknown environment are inherited by CLA, which are useful to solve problems relating to complex networks and systems.

3.4. Irregular Cellular Learning Automata

An irregular cellular automaton (Esnaashari and Meybodi, 2015) with the combination of LA generates the ICLA. ICLA can be represented by an undirected graph in which every vertex is appeared to be a cell, and each cell is accomplished with a LA, and each edge produce an adjacency relation between two cells (two LAs) as shown in Fig. 1. The LA associated with a specific cell takes its action to determine its

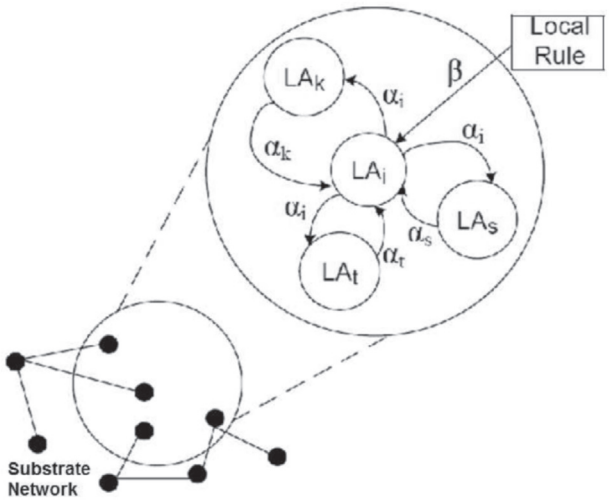

Fig. 1. Irregular cellular learning automata.

state on the basis of action probability vector. The reinforcement signal of the LA existing in a specific cell is decided by the rule of the CLA and the action elected by the neighboring LAs of that specific cell. The adjacent LAs of any specific LA construct the local environment of that cell. During the evolution of the ICLA, the probability vector of the adjacent LAs varies. So the local environment of a cell is changeable.

4. Problem description

Multi-domain networking brings additional challenges to SDN virtualization as only one controller is no longer valid in multi-domain environment due to single-point failure. The distributed controllers in multi-domain SDN improve the overall performance and scalability. Otherwise, the communication overhead increases in the large network as hop count in between the controller and data plane devices increase. To overcome such challenges, the virtualization layer of multi-domain SDN must provide an efficient approach that masks the fact that the actual network infrastructure consists of multiple heterogeneous domains.

In order to encounter such challenges, the virtualization layer of multi-domain SDN must provide a higher-level abstraction that conceals the fact that the underlying infrastructure actually consists of multiple domains with heterogeneous implementation. In the literature, most of the existing VNE approaches are restricted on a single domain maintained by an InP. It assumes complete knowledge of physical resources and the underlying substrate topology, which is difficult to perceive in multi-domain environment as there could be multiple InPs. The requested resource requirements for a vSDN may not be fulfilled by a single InP. Further, it is necessary to achieve better quality-of-service (QoS) while creating a VN.

On the other hand, multiple domains can provide their SDN infrastructure for creating a single logical network through virtualization as shown in Fig. 2. Multiple domains can effectively increase the response time and reduce the loss of data as VNE can be done from multiple domains. In Fig. 2, we have three domains – domain 1, 2 and 3 – consisting of different network switches. C1, C2 and C3 are the corresponding controller nodes for the domains. VN1 and VN2 are two virtual network requests. Multiple VN requests can share the same substrate resources such as CPU capacity, flow table capacity, memory capacity and bandwidth. Therefore, it is challenging to design a multi-domain virtual network embedding algorithm for software defined networks.

In Fig. 2, ACFGRK substrate path is used by VN1, and GRJ substrate path is used by VN2. Therefore, the flow table space of the switches G and R are shared by the VN1 and VN2. In this situation, the bandwidth of the shared link (e.g. G → R) is also divided to accommodate

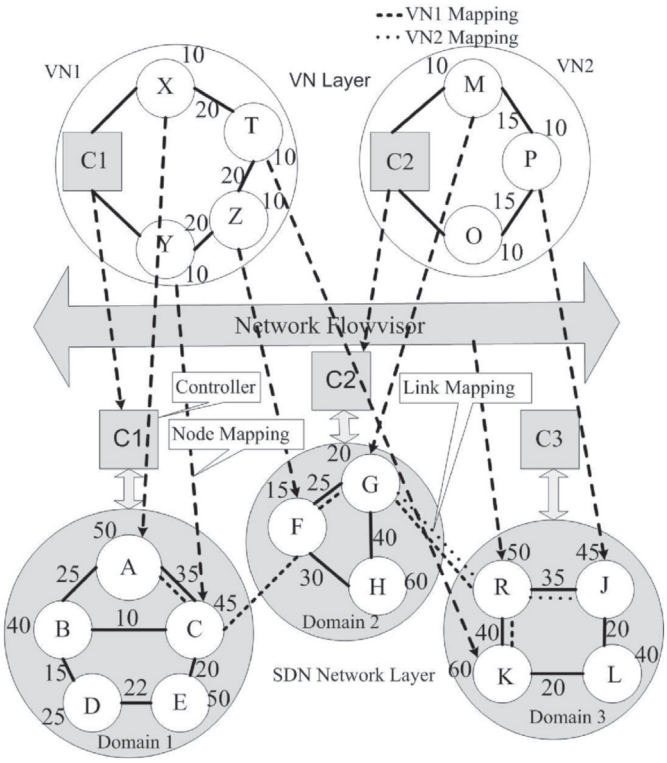

Fig. 2. Multi-domain virtual network embedding in SDN

the virtual link request. The resulting mapping reveals a problem that is distinct on SDN. To be specific, the source node in a VN contains the flow rules corresponding to destination nodes, but, in the mapped substrate network, we need extra rules for each of the flows in any intermediate switches like G and R to ensure that the flows end up with intended switches such as K or J in Fig. 2. If the coming flows are split to take different paths at node R, then, another rule is needed to choose different paths at R. Increases of number of rules in turn increases the cost of the VN. Consequently, the substrate node resources are diminished in quantity which leads subsequent VN requests to be postponed. Finally, it decrease the *acceptance ratio* of the VN requests and increase the *embedding cost*. Therefore, there is need of dynamic flow migration for dynamically managing the node and link resources to provide resource optimal routing.

5. Formal description of the problem

At the outset we mention that the keywords ‘vertex’, ‘node’, and ‘cell’ are used to represent the similar entity but in different context. Similarly, for the keywords ‘edge’ and ‘link’ too. In this section, we describe the formal description of vSDN-CLA followed by FM-vSDN in details. The substrate network allocates the required resources from a single or multi-domain substrate network to create the vSDN using vSDN-CLA algorithm. vSDN-CLA algorithm performs two major tasks – virtual node and link mapping, and placement of controller. The mapping algorithm considers two types of physical resource constraints:

- *Capacity Constraint*: Each physical node of a substrate SDN network is having some CPU capacity, flow table space and memory capacity. Further, a physical node is shared among different vSDN. Therefore, in node mapping, the physical node capacity (CPU, memory, and flow-table space) must be greater than the requested virtual node capacity.
- *Bandwidth Constraint*: Bandwidth consumption increases the overall cost of VNE because one virtual path can be assigned to multiple physical links, creating a connected path, which in turn occu-

pies more physical resources during virtualization. Moreover, every vSDN selects its own controller which again leads to more bandwidth requirement. Therefore, in link mapping, the bandwidth of substrate link must be higher than the desired virtual link bandwidth.

5.1. Node and link mapping

In this work, each SDN-based substrate network is defined as an isomorphic, asynchronous ICLA, $L = \{G^s\langle V^s, E^s\rangle, \phi, A, F\}$, where.

- $G^s\langle V^s, E^s\rangle$ is an undirected graph of nodes and links which represent each domain of substrate network.
- V^s is the finite set of substrate nodes. Each node is associated with a variable structure LA in the set A. Substrate nodes perform the task of both switches and SDN controller. E^s is the finite set of substrate links.
- ϕ is the finite set of states denoted by ϕ_i associated with each node i based upon the action probability vector $p = \{p_1, p_2, \ldots, p_r\}$ where r is the number of actions. In this work, we have only two states – either a substrate node is available to map to the virtual node or it is not available. Therefore, the set of states = {available, not available}.
- $A = \{LA_1, LA_2, - - -, LA_n\}$ is the set of n LAs where n is the total number of nodes in the substrate network. $LA_i = \{\alpha_i, \beta_i, p, T_i\}$, where α_i represents the action set, β_i represents the input set of reward and penalty for the node i, and T_i represents the learning algorithm of the LA_i. Here, set of actions = {accepted, postponed}, set of inputs = {give reward, impose penalty} and algorithm = ⟨Linear Reward-Inaction (L_{R-I}) ⟩. The substrate network environment is dynamically inconsistent. In such network, the node capacity and link bandwidth changes. It is assumed that these changes are known to the environment that sends reward/penalty signal to LA based upon link and node capacity change.
- $F : \phi_i \rightarrow \beta$ is the function of local rule in node i where β_i is the set of rewarded or penalized values of node i. Here, set of local rule = ⟨ all constraint checking ⟩.

In the ICLA, each LA A_i chooses its action $\alpha_i \in \alpha$ after receiving a reinforcement signal $\beta_i \in \beta$. The local rule $F : \phi_i \rightarrow \beta$ is applied to produce the reinforcement signal. At last, on the basis of the reinforcement signal and the chosen action by the cell, each LA updates its action probability vector. This process continues until the desired action is obtained.

It is pertinent to mention that the superscript 's' indicates substrate network and 'v' indicates virtual network. Each node (i.e. switch) $i \in V^s$ is associated with a certain flow table capacity S_i^s which defines the maximum number of flow information a switch can store in its flow table. At a particular time t, the flow table size of node i is denoted by $S_i^s(t)$. Each link $e_{ij} \in E^s$ of the substrate SDN connects the substrate nodes i and j, and it has bandwidth B_{ij}^s. Each node i is also associated with some content addressable memory capacity M_i^s and CPU capacity CPU_i^s. In virtualization, the substrate nodes can be shared among multiple vSDNs according to the requested amount of resources for the vSDN network.

In this network configuration, a requested VN is defined by an undirected graph as $G^v\langle V^v, E^v\rangle$ with a unique request id denoted by *Req_id*. Each VNR consists of node and link capacity requirements including a specific network topology. When a VNR arrives, SDN hypervisor executes the embedding strategy to serve the request. The vSDN node mapping and link mapping functions are denoted as $MP^V : V^v \rightarrow V^s$ and $MP^E : E^v \rightarrow E^s$ based upon the probabilistic decision function Q. The function Q considers the node and link mapping constraints which are as follows:

- *Memory Capacity*: Requested virtual node's memory capacity is always less than or equal to the substrate node's memory capacity.

$$M_l^v \leq M_j^s; l \in V^v, j \in V^s \tag{3}$$

- *CPU Capacity*: Requested virtual node's CPU capacity is always less than or equal to the substrate node's capacity.

$$CPU_l^v \leq CPU_j^s; l \in V^v, j \in V^s \tag{4}$$

- *Flow-Table Capacity*: At a particular time t, the flow table size of a virtual node is less than or equal to the flow-switching capacity of the corresponding substrate node.

$$S_l^v(t) \leq S_j^s; l \in V^v, j \in V^s \tag{5}$$

- *Link Bandwidth*: Requested bandwidth in the virtual link should be less than or equal to the substrate link bandwidth.

$$B_{kl}^v \leq B_{ij}^s; e_{kl} \in E^v, e_{ij} \in E^s \tag{6}$$

5.2. Controller placement

In SDN network, placement of the controller is an important task as the hypervisor slices the substrate network into multiple parts. After the successful mapping of virtual nodes and links, it is required to follow a separate step to select the controller for each VN, according to user requirements. The aim of the controller placement problem is to choose an optimal location of the controller among all the possible substrate nodes. The placement is defined by a mapping function $MP^V : V^v \rightarrow V^s(c)$, where $V^s(c)$ in the substrate node is chosen as a controller node, and it must be attached with one of the nodes of the substrate network. The selection of the controller should be optimal in such a way that the controller-to-switch delay is minimized. In a vSDN request, each virtual control link $e_{cv} \in E_c$ is mapped to a substrate link between the controller node and the substrate node that hosts virtual node V^v.

Let $D(e_{ij})$ be the transmission delay of the substrate link $e_{ij} \in E^s$. The average controller-to-switch delay (D) for vSDN request is computed as

$$D = \sum_{e_{ij} \in E^s} D(e_{ij}) / |E_c| \tag{7}$$

where $|E_c|$ is the total numbers of virtual control link. The transmission delay (d) between each virtual node and its controller should be less than the maximum permissible delay (D_p) defined by the user.

$$d(V^s(c), MP^V(V^v)) \leq D_p \tag{8}$$

5.3. Flow migration

After the successful execution of VNE and controller placement, it is necessary to migrate few flows to achieve resource optimal routing, as every switch contains a flow table with limited size. During flow migration, we have to consider the following three actions performed by the switches to minimize the cost of the VNs.

1. *Flow Rule Addition*: The selected controller node adds flow rules to the concerned nodes for the virtual links. For example, after the mapping of VN1 on substrate nodes A, C, F and K with controller C1, the controller C1 adds rules to nodes A, C, F and K, as shown in Fig. 2.
2. *Flow Rule Modification*: Substrate network is fragmented when arrival and departure of VNs happens because all the mapped virtual nodes are not from same substrate network. Flow rule of a VN is changed due to the departure of another VN. For example, the substrate path ACFGRK is changed to ACFK if the availability of the resources is satisfied by ACFK, as shown in Fig. 2. So substrate path

ACFGRK is migrated to ACFK, and the flow rule in C1 is modified.

3. *Flow Rule Deletion*: The controller deletes the flow rules in two cases. (i) Some flow rules become useless due to the migration of virtual links. These useless flow rules need to be deleted. For example, the flow rule for G and R is deleted from C1 if ACFGRK substrate path is migrated to ACFK, as shown in Fig. 2. (ii) Whenever a VN departs, the corresponding rules need to be deleted.

The flow migration can be direct or indirect. In case of direct flow migration, source node and destination nodes are directly connected. On the other hand, in indirect flow migration, source node is not directly connected with the destination node. Some intermediate nodes exist in between the source and the destination nodes in the substrate network. It may be possible that the intermediate nodes are not involved in the VN. These intermediate nodes can be shared by any other VN depending upon its status. In such a way node utilization and link utilization increased.

6. Algorithm design

In this section, we describe the proposed algorithms vSDN-CLA followed by FM-vSDN in details.

6.1. Node and link mapping algorithm

The proposed VNE algorithm, shown in Algorithm (1), works for both the intra-domain and inter-domain VNE. If a VNR is fully mapped with a single-domain substrate network, this is called intra-domain VNE. In inter-domain VNE, different parts of the VNRs are mapped in different substrate network. Each of the nodes of every substrate network is represented as a cell, and each cell is assigned with an LA. The Algorithm (1), is locally and independently run at each cell of the ICLA. Initially, each node assigns a default probability vector with each value equals 0.5, All the substrate nodes are available with full capacity. Each node maintains an array containing three information – current capacity of the node, number of neighbors and the current action taken by the node. For each of the requested virtual node, the resource constraints are compared for each available substrate node. If the requesting resources are not present, the vSDN requests are postponed. If more than one node is available to fulfill the requested capacity constraint, the node with minimum capacity is mapped with the virtual node. For the assigned node, if the residual capacity is 0, then the state of the node is changed to "not available". When the VN expires, the allocated substrate resources are released. Furthermore, if the reconfiguration of the VN is required, the substrate network releases previously assigned resources and then allocates new resources based on the existing assignment. After arriving a VNR, the substrate node sends a reinforcement signal to all its neighbors using the local rules. $F : \phi_i \rightarrow \beta$ be the function of local rules in each node, which are represented by Equations (3)–(6) to update the state of ϕ and to map the vSDN request to the substrate network. Based upon the action α_i of the LA, the random environment behaves. The random environment takes those actions as input and reacts either by providing a reward or charging a penalty. Each node send a packet called "STATUS" to its neighbor nodes containing the action information of the node. The reinforcement signal is computed according to the following local rules:

- If the selected action of LA_i was "accepted" and if either all neighbors' selected actions are "accepted" or any neighbor's selected action is "postponed", then node i rewards its selected action "accepted". Otherwise, node i penalizes its selected action.
- If the selected action of LA_i was "postponed" and if any neighbor's selected action is "postponed", then node i penalizes its selected action "postponed". Otherwise, node i rewards its selected action.

Algorithm 1 ICLA-based Multi-domain Node and Link Mapping Algorithm in virtual SDN

Inputs: $(G^s\langle V^s, E^s\rangle, \phi, A, F)$, $(Req\_id, G^v\langle V^v, E^v\rangle)$
Output: Node and Link Mapping
1: Each domain is represented as ICLA
2: Initially, action probability of each node equals 0.5
3: **for** Each available node i of each domain **do**
4: Get the resource capacities of node i
5: Select an action α_i based upon decision function $Q : \phi \rightarrow \alpha$ (using Equations (3)–(6))
6: Send "STATUS" packet to the neighbors
7: **if** $action_i$ is 'Accepted' **then**
8: **if** all neighbors' action are 'Accepted' OR any neighbor's action is 'Postponed' **then**
9: the selected action is rewarded
10: **else**
11: the selected action is penalized
12: **end if**
13: **else if** $action_i$ is 'Postponed' **then**
14: **if** any neighbor's action is 'Postponed' **then**
15: the selected action is penalized
16: **else**
17: the selected action is rewarded
18: **end if**
19: **end if**
20: **end for**
21: **if** All the rewarded nodes are in same domain **then**
22: **for** each link e_{lk} between the rewarded nodes **do**
23: **if** more than one path are found that can satisfy the requested link bandwidth **then**
24: map link e_{lk} with minimum bandwidth path
25: **end if**
26: **end for**
27: **else if** All the rewarded nodes are in different domain **then**
28: Send "DINFO" packet to multiple domain network
29: **if** different domains received a packet with same *Req_id* **then**
30: the rewarded nodes for same *Req_id* are connected with a virtual link.
31: **else**
32: link mapping fails
33: **end if**
34: **end if**
35: **for** Each available node i of each domain **do**
36: Update the state, ϕ_i.
37: Update the action probability vector, p.
38: **end for**

Here, $\beta = \{reward, penalty\}$ is the environment input to the LA_i. For each link connected with two rewarded nodes at the either ends, if more than one substrate path exists which can satisfy the requested bandwidth demand then substrate path with minimum bandwidth is mapped as a virtual link. To achieve inter-domain VNE, a communication protocol must be established in between different domains. Each domain send a packet called "DINFO" packet to all domains. The "DINFO" contains meta-virtual network request, G^v_m with the unique VNR ID, and *Req_id*. If different domains receive a packet with same *Req_id*, then the rewarded nodes with same VNR ID are connected with a virtual link. Finally, it updates the state of a node according to the recent action and also updates the action probability vector on the basis of the reinforcement signal and its selected action.

6.2. Controller placement algorithm

Algorithm 2 ICLA-based Controller Selection Algorithm

Inputs: VNE result, Permissible Delay (D_p)
Output: Placement of the Controller Node
1: **for** each VNR **do**
2: Create a set Z with all available substrate nodes. Initially all the available substrate nodes $\in Z$.
3: **for** Each node i in Z **do**
4: Calculate the delay (d) between node i and all other mapped nodes (using Equation (7)).
5: **end for**
6: **for** $j = 1 \rightarrow |V^v|$ **do**
7: Find all the available substrate nodes satisfying $d < D_p$, and put them in a set S^c
8: **end for**
9: **for** $j = 1 \rightarrow |S^c|$ **do**
10: Obtain the summation of hops in between ${S^c}_j$ and mapped virtual nodes.
11: **end for**
12: Select the node i with minimum hop count ${S^c}_i$ as controller node
13: **if** There exist more than one substrate nodes with minimum hops **then**
14: One of the controller nodes is selected based on the action probability vector of the nodes.
15: **end if**
16: **end for**
17: Update the state of the selected node i based on the action α_i.
18: Update the action probability vector of the selected node i based on previous state of the node and the reinforcement signal $\beta_i \in \beta$.

The Algorithm (2) shows the steps for controller node selection/mapping. After the successful execution of VNE, use the rewarded substrate nodes to select the nodes satisfying the controller requirements, and put these nodes into a controller candidate set S^c. Then, find out the hop count between each member of S^c and the virtual nodes, and, then, computes the sum of all the received hop counts. The node having minimum hop count will be selected as a controller node. If there is more than one node with minimum hop count value, then, according to the action probability vector of those nodes, the controller node will be selected. Finally, it updates the state of a node according to the recent action, and updates the action probability vector on the basis of the reinforcement signal and its selected action.

6.3. Flow migration algorithm

The Algorithm (3) shows the flow-migration procedure to achieve the resource optimal routing. After completion of the node and link mapping with optimal placement of the controller, the rewarded nodes are only responsible for the flow migration. If the rewarded nodes has a substrate path in between then the selected controller add the flow rules to those rewarded nodes. This algorithm also takes care of the departed VN by deleting the flow rules for that VN. If the substrate path contains any penalized node in between then this algorithm add the flow rules to that penalized node only. This algorithm reduced the overall complexity of the large network because addition of flow-rules is only done for current active VNs.

Algorithm 3 Flow Migration Algorithm

Inputs: Network node and link mapping with optimal placement of controller
Output: Flow Migration Action
1: **for** Each existing mapping of nodes and links **do**
2: **if** There is a mapped substrate path in between two rewarded nodes **then**
3: Add flow rules to the rewarded nodes.
4: **end if**
5: **if** There is a mapped substrate path in between rewarded node through penalized node **then**
6: **if** Any Previous VN departs **then**
7: Modify the Flow rules of Virtual link of existing VN.
8: Delete the Flow rules of the penalized node.
9: **end if**
10: **else**
11: Add flow rules to the rewarded and penalized nodes.
12: **end if**
13: **end for**
14: **for** Each expired mapping of nodes and links **do**
15: Delete the Flow rules of the each node of the corresponding mapping of nodes and links.
16: **end for**
17: Update the state of mapped nodes according to the action α_i.
18: Update the action probability vector of mapped nodes according to the previous state of node and the reinforcement signal.

7. Theoretical analysis

In this section, we formulate few lemmas to understand the proposed scheme within some limits of critical bounding assumptions.

Lemma 7.1. *Assume that the optimal VNE solution is accomplished by a inter-domain routing strategy and each domain contains a connected graph with V^s number of nodes, which can be represented as spanning tree with a root node called the controller node. Each domain contains l number of levels, and overall degree of each node is n. Inspired by lemma (2) of (Esnaashari and Meybodi, 2008), we propose that, overall every node in this directing tree communicates*

$$k(1 - n^{\log_n^{V^s}-1})(n^2 - n^{\log_n^{V^s}})/(1-n)^2 \tag{9}$$

packets of the other nodes, where k is the number of domain networks.

Proof. The root of the spanning tree is the controller node and other nodes are the switches. Assuming that overall degree of each node is n, so at level i, the average number of relayed packets is (γ_i) and the average number of packets communicated in each domain network (δ_i) is defined as follows:

$$n^{i+1} + n^{i+2} + - - - - + n^l = \gamma_i \tag{10}$$

as root node is the controller node and it is level 0, we are not considering this level. The above equation is in geometric progression. So, the above equation is represented as

$$n^{i+1} + n^{i+2} + - - - - + n^l = \gamma_i = n^{i+1}(1 - n^l)/(1-n) \tag{11}$$

Therefore average number of packet relay

$$\sum_{i=1}^{l}\gamma_i = \sum_{i=1}^{l}(n^{i+1}(1-n^l)/(1-n)) = (1-n^l)(n^2-n^{i+1})/(1-n)^2 \quad (12)$$

Therefore, the average number of packets relayed by k number of domain network is given below

$$k(1-n^l)(n^2-n^{i+1})/(1-n)^2 \quad (13)$$

Using

$$l = \log_n{}^{V^s} - 1$$

in the above equation, we get

$$k(1-n^{\log_n^{V^s}-1})(n^2-n^{\log_n^{V^s}})/(1-n)^2 \quad (14)$$

Hence the lemma.

Lemma 7.2. *Let LA_i be the LA residing in a substrate node, V_i^s with two available actions α_0 and α_1 respectively. LA_i executes the proposed 'vSDN-CLA' algorithm. Every node refreshes its activity likelihood vector in the wake of accepting the "STATUS" packet from every one of its neighbors nodes. Assume that the suitable action of LA_i is α_0 and the maximum time interval between two subsequent transmission of "STATUS" packets in 'vSDN-CLA' algorithm is t_0, respectively. Inspired by lemma (3) of (Ghaderi et al., 2014), we propose that the expected learning time of 'vSDN-CLA' algorithm in node V_i^s is computed as $\frac{1-a}{a}t_0$, where a is the reward parameter of the learning automata.*

Proof. We assume that, at time step $t = t'$, $p_0(t') = 1$ and $p_1(t') = 1$ with the maximum learning time of the LA. According to the 'vSDN-CLA' algorithm, each LA_i updates its action probability after receiving the "STATUS" packet after maximum duration of t_0 time. Hence, the learning time at each time step $t(L_i(t))$ is at most equal to t_0.

According to Equation (1) in Section 3.1, we can write

$$p_1(t'+1) = (1-a)p_1(t') = 1-a$$

$$p_1(t'+2) = (1-a)p_1(t'+1) = (1-a)^2$$

and so on. Similarly,

$$p_1(t'+t'') = (1-a)p_1(t'+t''+1) = (1-a)^{t''} = \epsilon$$

$$\lim_{t'''\to\infty} p_1(t'+t''+t''') = \lim_{t'''\to\infty}(1-a)^{t''+t'''} = 0$$

Based on the mathematical expectation formula, the expected learning time of $LA_i(E[L_i])$ is computed as follows:

$$E[L_i] = \sum_{t=1}^{\infty} L_i(t)p_1(t) \quad (15)$$

As the maximum value of $L_i(t) = t_0$, the above equation can be written as

$$E[L_i] = t_0\sum_{t=1}^{\infty}p_1(t) = t_0\sum_{t=1}^{\infty}(1-a)^t \quad (16)$$

As R.H.S term of the above equation is in geometric progression and $(1-a)^{\infty} = 0$, the above equation is written as

$$E[L_i] = t_0\frac{(1-a)}{a} \quad (17)$$

Hence the lemma.

Lemma 7.3. *Consider that LA_i be the LA dwelling in a substrate node, V_i^s with two available actions α_0 and α_1 respectively. LA_i executes the proposed 'vSDN-CLA' algorithm. Assume that the suitable action of LA_i is α_0. Also, assume that the learning process is stopped when $p_1 < \epsilon$. Then the maximum learning time of 'vSDN-CLA' in node V_i^s is $t_0\frac{\log\epsilon}{\log(1-a)}$, where a is the reward parameter of the learning automata and t_0 is the maximum time interim within two succeeding transmission of "STATUS" packets in 'vSDN-CLA' algorithm. Assume that we are considering the longest running time of LA_i.*

Proof. According to Equation (1) in Section 3.1, at time step $t = t' + t''$, we get

$$p_1(t) = (1-a)^{t''} = \epsilon$$

Therefore, we get

$$t'' = \frac{\log\epsilon}{\log(1-a)} \quad (18)$$

Therefor, the maximum learning time of 'vSDN-CLA' algorithm in node V_i^s is $\max((L_i))$.

$$max(L_i) = t''t_0 = t_0\frac{\log\epsilon}{\log(1-a)} \quad (19)$$

Hence the lemma.

Lemma 7.4. *After the termination of the Controller selection phase of 'vSDN-CLA' algorithm, any substrate node is chosen as a controller node except the embedded substrate nodes for a vSDN. So $MP^V(V^s) \cap V^s(c) = \phi$, where $MP^V(V^s)$ are the mapped substrate nodes and $V^s(c)$ is the selected controller node.*

Proof. After the completion of the initial stage of the 'vSDN-CLA' algorithm, the controller placement algorithm is started taking the VNE and Permissible delay, D as inputs. The nodes which are participated in a VNE for a VNR, changed its state to 'not available'. So state of all $MP^V(V^s)$ is 'not available'. We assume that only the 'available' state nodes are participated to further steps. The delay is calculated in between the nodes participated in the VNE and rest of the substrate nodes with 'available' state for a VNR. Therefore, only those nodes can be selected as a controller node, which is not in $MP^V(V^s)$. So controller selection algorithm, is selected the controller node except the embedded substrate nodes. Therefore, $MP^V(V^s) \cap V^s(c) = \phi$

8. Performance Evaluation

The proposed VNE algorithm vSDN-CLA and the flow migration algorithm FM-vSDN are evaluated exhaustively for both the intra-domain and inter-domain VNE using mininet (De Oliveira et al., 2014).

8.1. Evaluation metrics

We have used the following metrics to evaluate the performance of the proposed schemes and compare the performances with the benchmark protocols. The metrics are: (i) controller-to-switch delay (D), as shown in Equation (7), (ii) acceptance ratio of the vSDN requests, which is the ratio between the accepted requests and the total requests arrived, (iii) average end-to-end delay, (iv) average throughput, (v) average link resource utilization, (vi) average node resource utilization, and (vii) embedding revenue and cost ratio (R/C). We measure the end-to-end delay using 'ping' command, and the throughput by executing file transfer between each pair of nodes via TCP using the 'Iperf' tool. We show the R/C ratio computation method below. The revenue of embedding a VN, G^v, at time t refers to the total amount of resources it requested, which is defined as (Yu et al., 2008), (Chowdhury et al., 2012):

$$R_t(G^v) = \sum_{i\in V^v}(M_i^v + CPU_i^v + S_i^v) + \sum_{ij\in E^v}B_{ij}^v \quad (20)$$

The cost of embedding a VN, G^v, at time t is the total substrate resources allocated to the virtual network, which is defined as:

$$C_t(G^v) = \sum_{i\in V^v}(M_i^v + CPU_i^v + S_i^v) + \sum_{ij\in E^v}(\delta_{ji}B_{ij}^v) \quad (21)$$

where δ_{ji} defines the intermediate Boolean variable which indicates a value equals 1 if some amount of bandwidth from link e_{ij}^s is assigned to a

virtual link of virtual node i; otherwise the value is 0. So, the long-term average revenue is defined as (Yu et al., 2008):

$$R = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R_t(G^v)}{T} \tag{22}$$

Similarly, the long-term average cost is defined as:

$$C = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} C_t(G^v)}{T} \tag{23}$$

Finally, the long-term revenue and cost ratio is defined as:

$$\frac{R}{C} = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R_t(G^v)}{\sum_{t=0}^{T} C_t(G^v)} \tag{24}$$

8.2. Experimental setup

We consider tree topology, as it is popularly used in data center network and cloud computing environment, with different size and shape for intra-domain and inter-domain VNE. We evaluate the proposed schemes with various network properties. We consider that the substrate network consists of 100 nodes and 530 links. Each node is considered as a cell with the initial probability equals 0.5. For each substrate node, the capacity of the content addressable memory, capacity of CPU, and flow table capacity varies from 50 to 100 unit following the uniform distribution. For each substrate link, the transmission delay varies from 5 ms to 20 ms. The available bandwidth capacity of each of the substrate link varies from 50 to 100 unit in uniformly distributed manner. We consider the vSDN requests of different sizes. The vSDN requests for 2 to 10 nodes is represented as 'small', for 10 to 20 nodes is represented as 'medium', and for 20 to 50 nodes is represented as 'large'. We use Poisson process for determining the arrival rate of vSDN requests, as the interarrival times are independent and exponentially distributed. The average arrival rate of vSDN requests is 5 per 100 time units. Each vSDN request has an exponential life span with the average of 1000 time units. We assume that each flow rule requires 1 unit of switching memory. The node and link utilization simulation is performed for 500 VNRs. The above mentioned configuration is taken for both the intra-domain VNE and inter-domain VNE with 4 domains. All the required parameters are tabulated in Table 1.

The above mentioned parameters are used in the state-of-the-art algorithms to achieve QoS requirements in the study of VNE with controller placement and flow migration in multi-domain SDN environment. We use FlowVisor (Blenk et al., 2016) as a SDN hypervisor which separates the substrate network in different isolated virtual networks. We take different controllers for each and every virtual SDN network, and we set up NOX (Blenk et al., 2016) as a virtual SDN controller. The coordinated virtual node and link mapping algorithm, as shown in Algorithm (1), always checks the availability of the node and link resources using Equations (3)–(6). In the controller selection algorithm, as shown in Algorithm (2), we use predefined values for 'permissible delay'. We heuristically determine the values after extensive simulation using above mentioned network setting. In case of large number of substrate nodes, the permissible delay is considered as 50 ms.

The above mentioned topology is good enough where scalability is concerned due to its hierarchical nature. It can accommodate any number of nodes in its hierarchical chain. Our algorithm handles both the automated mapping and the placement of the distributed controller dynamically. In case of a very large network we have to only increase the domain with its own controller. The proposed algorithm will run simultaneously in each and every node to handle the controller placement problem. So distributed controller solution improves the performance and scalability issues.

Finally, we compare the received result of the proposed scheme vSDN-CLA with the benchmark schemes CO-vSDN (Gong et al., 2016) and DM-vSDN (Demicri and Ammar, 2014), which are the baseline strategy, as these algorithms have considered VNE with the placement of the controller and the results of FM-vSDN with the benchmark scheme SDN-VN (Mijumbi et al., 2014), which considered the flow-migration in SDN-based virtualized network. Not only that, we have also compared our controller placement algorithm with a very recent paper where the authors proposed a controller placement algorithm based on learning automata known as LACP. For the controller-to-switch delay comparison, we run the LACP algorithm (Mostafaei et al., 2018) in our experimental setup.

8.3. Simulation result of vSDN-CLA

At the outset, we evaluate the performance of the proposed scheme vSDN-CLA in terms of average controller-to-switch delay, acceptance ratio of VN requests, R/C ratio, average end-to-end delay, throughput, average node resource utilization and average link resource utilization and, then, compare the results with that in the benchmark schemes. We consider the tree topology with the 'Large' number of nodes to perform the experiments in different simulation time range 5s–50s with 95% confidence level unless and otherwise it is specified explicitly. The received results corresponding to intra-domain and inter-domain VNE are mentioned as 'vSDN-CLA-intra' and 'vSDN-CLA-inter', respectively.

In the experiments corresponding to Fig. 3, we observe that, in different simulation time, the average controller-to-switch delay is always less in vSDN-CLA compared to that in the CO-vSDN (Gong et al., 2016), DM-vSDN (Demicri and Ammar, 2014) and LACP (Mostafaei et al., 2018). This is because the controller selection is performed with minimum hop count after the successful execution of VNE.

In Fig. 4, we measure the average delay under different substrate network size for both the intra-domain and inter-domain VNE. This experiment once again shows that the average controller-to-switch

Table 1
Simulation parameters.

Parameters	Value
Number of nodes in the substrate network	100
Number of links in the substrate network	530
Initial available computing resources on substrate nodes	50–100 units
Transmission delay for each substrate link	5–20 ms
Bandwidth capacity of each substrate link	50–100 units
vSDN request represented as small	2-10 nodes
vSDN request represented as medium	10-20 nodes
vSDN request represented as large	20-50 nodes
Average arrival rate of vSDN requests	5 per 100 unit time
Average life span of each vSDN request	1000 time units
Switching memory of each flow	1 unit
Number of VNRs on which node and link utilization performed	500
Number of domains	4
Initial probability of each node	0.5

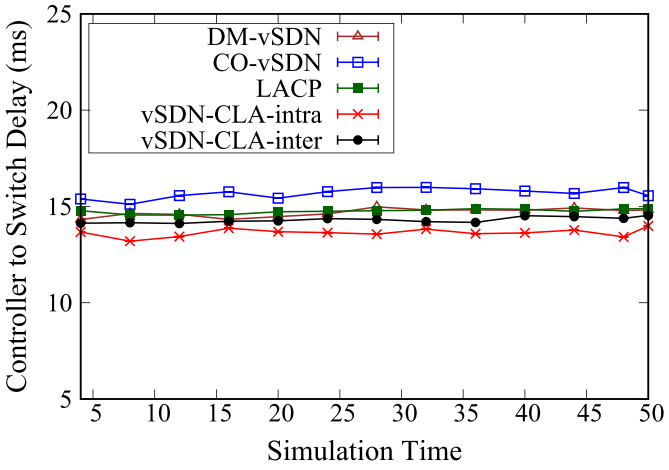

Fig. 3. Average controller-to-switch delay.

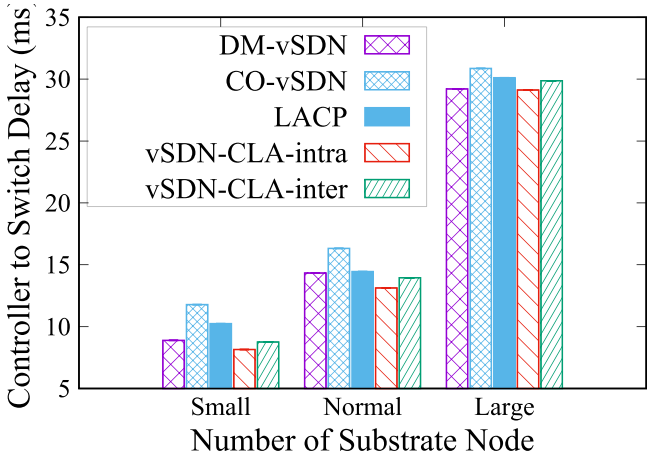

Fig. 4. Average controller-to-switch delay under different set of vSDN.

delays remain lesser than that in the benchmark schemes in each and every case.

Fig. 5 shows the comparison of acceptance ratios corresponding to the proposed and the benchmark schemes. We observed that the average acceptance ratio in vSDN-CLA is 0.935, which is higher than that in the other two benchmark schemes.

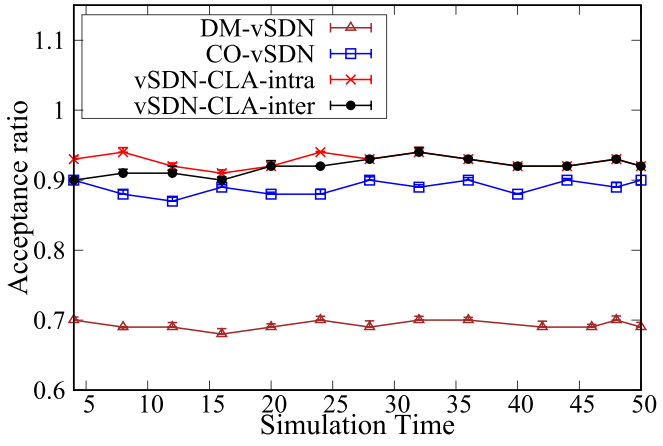

Fig. 5. Acceptance ratio.

In Fig. 6, it is shown that the R/C ratio of vSDN-CLA for both the intra-domain and inter-domain VNE are higher than that in CO-vSDN and DM-vSDN. In vSDN-CLA, the node mapping is done using local rules of the network. Those rules helps the VNE to achieve the optimal mapping which in turn reduces the cost of link mapping, and, thus, increases the value of R/C ratio. The higher R/C ratio signifies the higher resource utilization of the substrate network which leads to higher acceptance ratio of the vSDN requests with minimum cost.

Figs. 7 and 8 shows the R/C ratio and acceptance ratio in different size of vSDN requests using tree topology. The figures show that, in all the three algorithms, the R/C ratio and the acceptance ratio decrease with the increase of the number of vSDN requests. The reason behind this is the requirement of more resources once the vSDN requests increase. Due to the limitation of the bandwidth of the substrate link, the virtual links are to be assigned with longer substrate path. Thus, it consumes more resources which ultimately decrease the R/C ratio. More resource consumption may lead to 'postpone' the vSDN request, which in turn leads to the lower acceptance ratio.

To evaluate the performance of the schemes in terms of average end-to-end delay and throughput, we consider different substrate network size. We emulate the operation of an SDN with multiple vSDN network placed upon it. In this experiment, 20 different vSDN networks are embedded on 6 substrate networks sliced by the FlowVisor. We take the substrate nodes in between 50 and 100, and the virtual nodes in between 5 and 20. The received results are plotted in Figs. 9 and 10, respectively. Both the figures show that the proposed scheme outper-

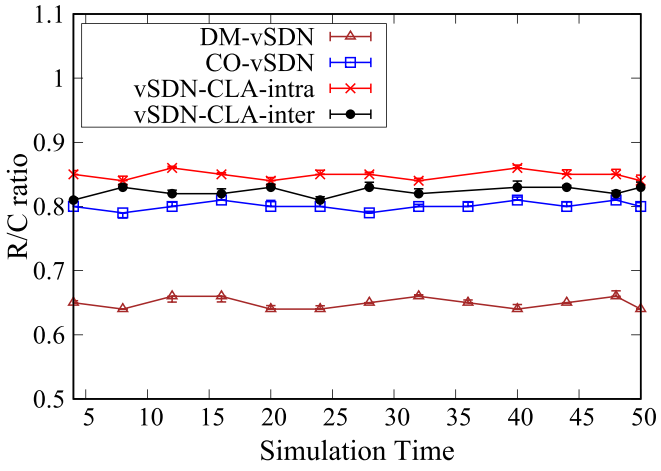

Fig. 6. Long-term revenue to cost ratio.

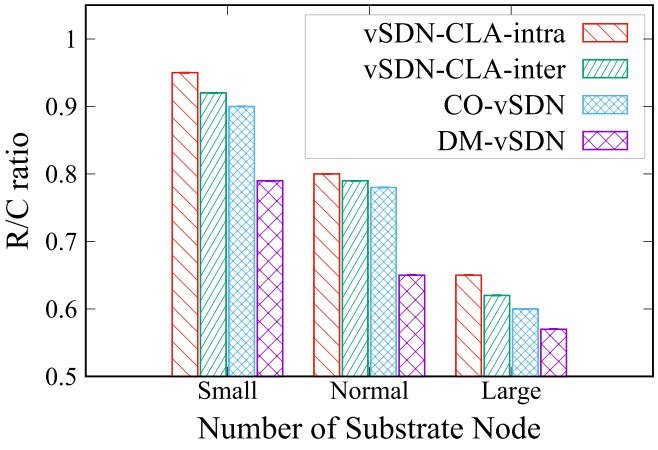

Fig. 7. Revenue-Cost ratio under different set of vSDN.

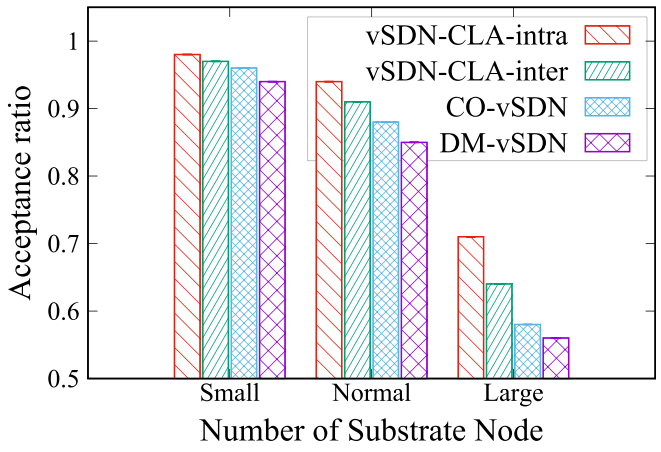

Fig. 8. Acceptance ratio under different set of vSDN.

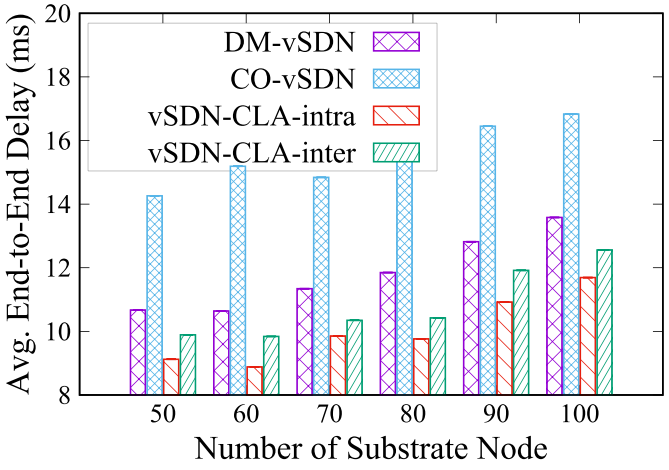

Fig. 9. Average end-to-end delay.

forms the benchmark schemes. In the proposed scheme, the end-to-end delay is minimized due to the use of minimum hop count for performing the VNE, and, this in turn, increases the throughput.

Summary: We observe that the proposed scheme vSDN-CLA outperforms the benchmark schemes with respect to all the mentioned metrics. For example, considering a substrate network of 100 nodes, the proposed scheme achieved 5.60% and 10.03% lesser controller-to-switch delay, 23.72% and 10.55% higher throughput, 13.19% and 30.60% lesser end-to-end delay, 14.03% and 8.33% higher R/C ratio, and 14.28% and 10.34% higher acceptance ratio compared to that in two benchmark schemes DM-vSDN and CO-vSDN, respectively, in intra-domain VNE. Again, considering a substrate network of 4 domains having 100 nodes each, the proposed scheme achieved 1.96% and 6.56% lesser controller-to-switch delay, 18.33% and 5.74% higher throughput, 7.51% and 25.37% lesser end-to-end delay, 10.71% and 3.57% higher R/C ratio, and 30.90% and 24.14% higher acceptance ratio compared to that in two benchmark schemes DM-vSDN and CO-vSDN, respectively, in inter-domain VNE.

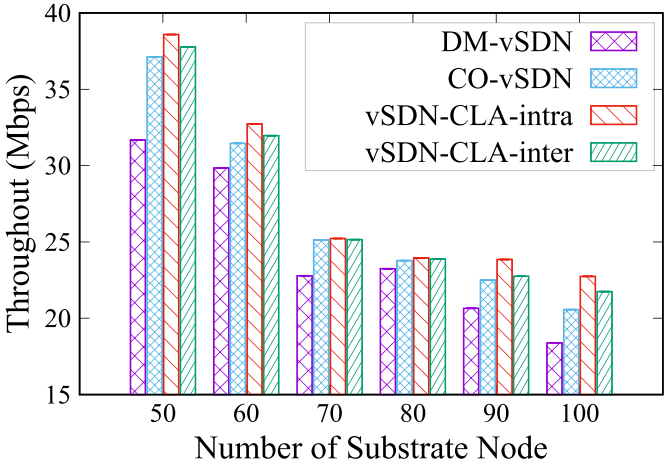

Fig. 10. Throughput.

8.4. Simulation result of FM-vSDN

In this section, we evaluate the performance of the FM-vSDN scheme. As the above benchmark schemes did not consider the flow migration concept, we use another benchmark scheme namely SDN-VN (Mijumbi et al., 2014) in this section.

We evaluate the performance in terms of average node and link resource utilization, and acceptance ratio with respect to the number of VN requests in both the intra-domain and inter-domain SDN environment. We take the total number of VN requests in between 50 and 500. The received results are plotted in Figs. 11–13, respectively. In the figures, we observe that the resource utilization and acceptance ratio is larger than that in benchmark scheme. Further, as we take the rewarded and penalized nodes to do the flow migration, there is no need to calculate any other parameters such as shortest path to take different actions of the flow migration in this proposed scheme. It shows the optimal resource routing.

Summary: We observe that the proposed scheme of flow migration outperforms the benchmark scheme in both the intra-domain and inter-domain SDN environment with respect to the above mentioned metrics. For example, considering 500 VN requests, the proposed scheme achieved 12.90% and 4.84% higher node utilization, 16.28% and 4.65% higher link utilization and 4.25% and 3.19% higher acceptance ratio in intra-domain and inter-domain SDN environment, respectively, compared to that in the benchmark scheme SDN-VN.

9. Conclusion and future work

This paper presents an ICLA based multi-domain VNE with controller selection strategy namely *vSDN-CLA*. The scheme is further extended in FM-vSDN by considering dynamic flow migration in SDN. The aim of this solution is to achieve an efficient mapping between VN

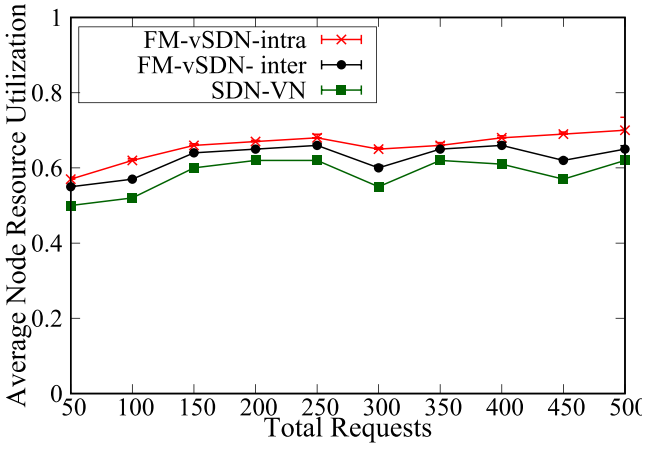

Fig. 11. Average node resource utilization.

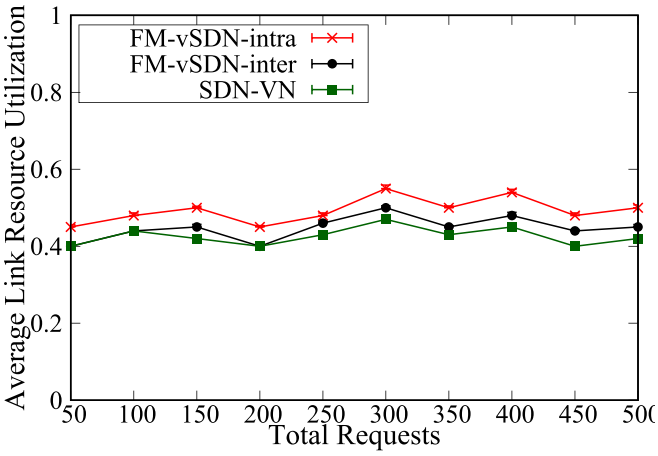

Fig. 12. Average link resource utilization.

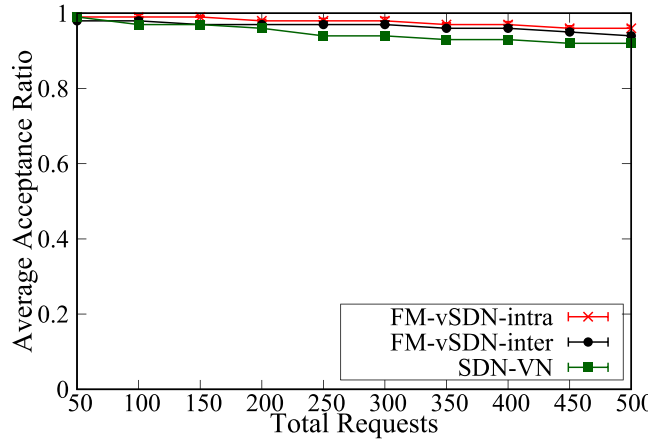

Fig. 13. Average acceptance ratio.

and substrate network followed by resource optimal routing in multi-domain SDN environment. At the same time, the proposed schemes satisfy the QoS requirements in terms of memory, CPU, bandwidth, and flow table capacity of the nodes. The simulation results show that the vSDN-CLA and FM-vSDN schemes significantly outperforms the existing benchmark schemes in terms of controller-to-switch delay, end-to-end delay, throughput, acceptance ratio, resource utilization, and embedding cost.

In this work, we restrict our analysis by not considering many factors such as energy consumption, fault tolerance and unavailability of path. We intend to design a more flexible scheme for the same environment considering those aspects in the future.

CRediT authorship contribution statement

Dipanwita Thakur: Conceptualization, Methodology, Visualization, Investigation, Validation, Writing - original draft. **Manas Khatua:** Supervision, Writing - review & editing.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jnca.2020.102639.

References

Aghababa, A.B., Fathinavid, A., Salari, A., Zavareh, S.E.H., 2012. A novel approach for malicious nodes detection in ad-hoc networks based on cellular learning automata. In: Proceedings of the World Congress on Information and Communication Technologies, pp. 82–88.

Blenk, A., Basta, A., Reisslein, M., Kellerer, W., 2016. Survey on network virtualization hypervisors for software defined networking. IEEE Commun. Surv. Tutor. 18, 655–685.

Chen, Y., Zheng, H., Wu, J., 2019. Consistency, feasibility, and optimality of network update in SDNs. IEEE Trans. Netw. Sci. Eng. 6, 824–835.

Chowdhury, M., Rahman, M.R., Boutaba, R., 2012. ViNEYard: virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Trans. Netw. 20, 206–219.

Chowdhury, N.M.M.K., Rahman, M.R., Boutaba, R., 2009. Virtual network embedding with coordinated node and link mapping. In: Proceedings of the IEEE INFOCOM, pp. 783–791.

De Oliveira, R.L.S., Shinoda, A.A., Schweitzer, C.M., Prete, L.R., 2014. Using mininet for emulation and prototyping software-defined networks. In: Proceedings of IEEE Colombian Conference on Communications and Computing, pp. 1–6.

Demicri, M., Ammar, M.H., 2014. Design and analysis of techniques for mapping virtual networks to software-defined network substrates. Comput. Commun. 45, 1–10.

Dietrich, D., Rizk, A., Papadimitriou, P., 2013. Multi-domain virtual network embedding with limited information disclosure. In: IFIP Networking Conference, pp. 1–9.

Esnaashari, M., Meybodi, M.R., 2008. A cellular learning automata based clustering algorithm for wireless sensor networks. Sens. Lett. 6, 723–735.

Esnaashari, M., Meybodi, M.R., 2015. Irregular cellular learning automata. IEEE Trans. Cybern. 45, 1622–1632.

Feng, M., Liao, J., Qing, S., Li, T., Wang, J., 2018. Cove: Co-operative virtual network embedding for network virtualization. J. Netw. Syst. Manag. 26.

Feng, T., Bi, J., Wang, K., 2014. Joint allocation and scheduling of network resource for multiple control applications in sdn. In: Proceedings of the IEEE Network Operations and Management Symposium, pp. 1–7.

Fischer, A., Botero, J.F., Beck, M.T., de Meer, H., Hesselbach, X., 2013. Virtual network embedding: a survey. IEEE Commun. Surv. Tutor. 15, 1888–1906.

Ghaderi, R., Esnaashari, M., Meybodi, M.R., 2014. A cellular learning automata-based algorithm for solving the coverage and connectivity problem in wireless sensor networks. Ad Hoc Sens. Wirel. Netw. 22, 171–203.

Ghorbani, S., Schlesinger, C., Monaco, M., Keller, E., Caesar, M., Rexford, J., Walker, D., 2014. Transparent, live migration of a software-defined network. In: Proceedings of the ACM Symposium on Cloud Computing, NY, USA pp. 3:13:14.

Gong, S., Chen, J., Kang, Q., Meng, Q., Zhu, Q., yi Zhao, S., 2016. An efficient and coordinated mapping algorithm in virtualized SDN networks. Front. Inform. Technol. Elect. Eng. 17, 701–716.

Guo, K., Wang, Y., Qiu, X., Li, W., Xiao, A., 2015. Particle swarm optimization based multi-domain virtual network embedding. In: IFIP/IEEE International Symposium on Integrated Network Management, pp. 798–801.

Haeri, S., Trajkovi, L., 2017. Virtual network embedding via Monte Carlo tree search. IEEE Trans. Cybern. 47, 1–12.

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, NY, USA, pp. 7–12.

Hu, Y., Wang, W., Gong, X., Que, X., Cheng, S., 2014. On the placement of controllers in software-defined networks. Ad Hoc Sens. Wirel. Netw. 22, 171–203.

Jimnez, Y., Cervell-Pastor, C., Garca, A.J., 2014. On the controller placement for designing a distributed SDN control layer. In: Proceeding of the IFIP Networking Conference, pp. 1–9.

Killi, B.P.R., Rao, S.V., 2018. On placement of hypervisors and controllers in virtualized software defined network. IEEE Trans. Netw. Serv. Manag. 15, 840–853.

Kreutz, D., Ramos, F.M., Verissimo, P., Rothenberg, C., Azodolmolky, S., Uhlig, S., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103, 14–76.

Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Hock, D., Jarschel, M., Hoffmann, M., 2015. Heuristic approaches to the controller placement problem in large scale SDN networks. IEEE Trans. Netw. Serv. Manag. 12, 4–17.

Li, S., Saidi, M., Chen, K., 2017. Multi-domain virtual network embedding with coordinated link mapping. Adv. Sci. Technol. Eng. Syst. J. 2, 545–552.

Luo, G., Ding, H., Zhou, J., Zhang, J., Zhao, Y., Chen, B., Ma, C., 2014. Survivable virtual optical network embedding with probabilistic network-element failures in elastic optical networks. In: 13th International Conference on Optical Communications and Networks, pp. 1–4.

Mijumbi, R., Serrat, J., Rubio-Loyola, J., Bouten, N., Latre, S., Turck, F.D., 2014. Dynamic resource management in SDN-based virtualized networks. In: Proceedings of IEEE International Workshop on Management of SDN and NFV Systems, pp. 412–417.

Mostafaei, H., Menthy, M., Obaidat, M.S., 2018. A learning automaton-based controller placement algorithm for software-defined networks. In: IEEE Global Communications Conference, pp. 1–6.

Rahman, M.R., Boutaba, R., 2013. SVNE: survivable virtual network embedding algorithms for network virtualization. IEEE Trans. Netw. Serv. Manag. 10, 105–118.

Rath, H.K., Revoori, V., Nadaf, S.M., Simha, A., 2014. Optimal controller placement in software defined networks (SDN) using a non-zero-sum game. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, pp. 1–6.

Samuel, F., Chowdhury, M., Boutaba, R., 2013. Polyvine: policy-based virtual network embedding across multiple domains. J. Inter. Serv. Appl. 4, 1–23.

Sun, G., Yu, H., Anand, V., Li, L., 2013. A cost efficient framework and algorithm for embedding dynamic virtual network requests. Future Generat. Comput. Syst. 29, 1265–1277.
Thakur, D., Khatua, M., 2018. Cellular learning automata-based virtual network embedding in software-defined networks. In: Proceedings of 2nd International Conference on Communication, Computing and Networking, pp. 173–182.
Thathachar, M.A.L., Sastry, P.S., 2002. Varieties of learning automata: an overview. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 32, 711–722.
Tuncer, D., Charalambides, M., Clayman, S., Pavlou, G., 2015. Adaptive resource management and control in software defined networks. IEEE Trans. Netw. Serv. Manag. 12, 18–33.
Wibowo, F.X., Gregory, M.A., Ahmed, K., Gomez, K.M., 2017. Multi-domain software defined networking: research status and challenges. J. Netw. Comput. Appl. 87, 32–47.
Xiao, X., Zheng, X., Zheng, Y., 2017. A multidomain survivable virtual netwok mapping algorithm. Secur. Commun. Network. 2017, 1–12.
Yang, H., Zhang, J., Ji, Y., Tian, R., Han, J., Lee, Y., 2015. Performance evaluation of multi-stratum resources integration based on network function virtualization in software defined elastic data center optical interconnect. Optic Express 23, 31192–31205.
Yang, H., Zhang, J., Zhao, Y., Han, J., Lin, Y., Lee, Y., 2016. SUDOI: software defined networking for ubiquitous data center optical interconnection. IEEE Commun. Mag. 54, 86–95.
Yu, M., Yi, Y., Rexford, J., Chiang, M., 2008. Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comp. Commun.Rev. 38, 17–29.
Zhong, X., Wang, Y., Qiu, X., Li, W., 2016. Flowvisor-based cost-aware vn embedding in openflow networks. Int. J. Netw. Manag. 26, 373–395.
Zhou, B., Gao, W., Zhao, S., Lu, X., Du, Z., Wu, C., Yang, Q., 2014. Virtual network mapping for multi-domain data plane in software-defined networks. In: Proceedings of the International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems, pp. 1–5.
Zhu, L., Sun, W., Hu, W., 2017. Delay constrained vSDN embedding in WAN. In: Computing Conference, pp. 1283–1289.
Zhu, Y., Ammar, M., 2006. Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of the IEEE INFOCOM, pp. 1–12.

Dipanwita Thakur is an Assistant Professor in the Department of Computer Science and Engineering at Banasthali Vidyapith, Rajasthan, India, since October 2009. Prior to that, she was a teaching faculty in the Dept. of Computer Science and Engineering, Banasthali Vidyapith, Rajasthan from 2008 to 2009 and was Technical Support Engineer in Netgear process, from 2007 to 2008. She received her M.Tech degree in Computer Science in 2007 from Banasthali Vidyapith, Rajasthan, India. Her research interests include Software Defined Networking, Cellular learning Automata and Network Virtualization. She is a member of the IEEE.

Manas Khatua is an Assistant Professor in the Dept. of Computer Science and Engineering at Indian Institute of Technology Guwahati, India, since May 2018. Prior to that, he was an Assistant Professor in the Dept. of Computer Science and Engineering, Indian Institute of Technology Jodhpur, from 2016 to 2018, and was a Postdoctoral Research Fellow at Singapore University of Technology and Design, Singapore, from 2015 to 2016. He completed his Ph.D. from Indian Institute of Technology Kharagpur, India, in 2015. He was associated with Tata Consultancy Services (India) for two and half years, and worked as a Lecturer of Bankura Unnayani Institute of Engineering, India, for more than two years. His research interests include Performance Evaluation of Protocols, Wireless LANs, Sensor Networks, Network Security and Internet of Things. He is an author of many reputed international journal and conference publications. He is a member of the IEEE and ACM.