Multi-Layer Virtual Network Embedding

Shihabur Rahman Chowdhury, *Student Member, IEEE*, Sara Ayoubi, Reaz Ahmed, Nashid Shahriar, *Student Member, IEEE*, Raouf Boutaba, *Fellow, IEEE*, Jeebak Mitra, and Liu Liu

***Abstract*—Network virtualization (NV), considered as a key enabler for overcoming the ossification of the Internet allows multiple heterogeneous virtual networks to co-exist over the same substrate network. Resource allocation problems in NV have been extensively studied for single layer substrates such as IP or Optical networks. However, little effort has been put to address the same problem for multi-layer IP-over-optical networks. The increasing popularity of multi-layer networks for deploying backbones combined with their unique characteristics (*e.g.,* topological flexibility of the IP layer) calls for the need to carefully investigate the resource provisioning problems arising from their virtualization. In this paper, we address the problem of multi-layer virtual network embedding (MULE; similar to multi-layer networks, this hybrid species brings the best of two species together.) on IP-over-optical networks. We propose two solutions to MULE: 1) an integer linear program formulation for the optimal solution (OPT-MULE) and 2) a heuristic to address the computational complexity of the optimal solution (FAST-MULE). We demonstrate through extensive simulations that on average our heuristic performs within ≈1.47× of optimal solution while executing several orders of magnitude faster. Simulation results also show that FAST-MULE incurs ≈66% less cost on average than the state-of-the-art heuristic while accepting ≈60% more virtual network requests on average.**

***Index Terms*—Computer network management, overlay networks.**

I. Introduction

MULTI-LAYER IP-over-Optical networks are becoming a popular choice among Infrastructure Providers (InPs) for deploying wide area networks [1]. Such multi-layer network typically consists of an optical substrate for the physical communication with an IP overlay on top [2]. This network model is being increasingly adopted for backbone networks as it offers the best of both worlds, *i.e.,* the flexibility in addressing, resource allocation, and traffic engineering of IP networks along with the high capacity provided by optical networks. Despite their increasing popularity, research on addressing resource provisioning challenges for virtualizing such networks is still in its infancy. A classical resource provisioning problem in network virtualization is Virtual Network Embedding (VNE), which consists in establishing a Virtual Network (VN) on a Substrate Network (SN) with objectives such as minimizing resource provisioning cost [3], [4], maximizing the number of admitted VNs [5], *etc*. VNE has been extensively studied for single-layer SNs [6] with significantly lesser attention paid to the multi-layer network substrates [7]. The topological flexibility provided by multi-layer networks [8] poses some unique challenges for VNE and calls for new investigations.

Several deployment models exist for multi-layer IP-over-Optical networks [9] including but not limited to: (i) IP over Dense Wavelength Division Multiplexed (DWDM); (ii) IP over Optical Transport Network (OTN) over DWDM. DWDM networks have specific constraints such as wavelength continuity for optical circuits and typically do not have transparent traffic grooming capabilities. A more favorable choice (also our choice of technology) is to deploy an OTN [10] over a DWDM network with advanced transport capabilities (*e.g.,* traffic grooming without optical-electrical-optical conversion). The OTN in turn can be *static*, *i.e.,* necessary interfaces on OTN nodes have been configured and the corresponding light paths in the DWDM layer have been lit to provision fixed bandwidth between OTN nodes. Or, the OTN can be *dynamic*, *i.e.,* more bandwidth between OTN nodes can be provisioned by lighting new light paths in the DWDM. Clearly, the VNE problem for each of these scenarios requires dedicated explorations due to their unique constraints. As a first step towards addressing VNE for multi-layer networks, we limit the scope of this paper to the case of a *static OTN* and leave the other possible deployment scenarios for future investigation.

Solving the VNE problem for multi-layer networks exhibits many unique challenges due to the topological flexibility offered by such networks. Concretely, although the OTN is fixed, the IP network is dynamic, *i.e.,* new IP links can be established when needed by provisioning necessary capacity from the OTN. Such flexibility can be exploited if residual resources in the IP layer are insufficient to admit a new VN, or to reduce the cost of VN embedding by creating new IP links that reduce network diameter. Provisioning new IP links in optical networks has been a tedious and manual task with a

Manuscript received March 2, 2018; accepted May 3, 2018. Date of publication May 8, 2018; date of current version September 7, 2018. This work was supported in part by Huawei Technologies and in part by an NSERC Collaborative Research and Development Grant. Additionally, this work benefited from the use of the CrySP RIPPLE Facility at the University of Waterloo. The associate editor coordinating the review of this paper and approving it for publication was C. Fung. *(Corresponding author: Shihabur Rahman Chowdhury.)*

S. R. Chowdhury, R. Ahmed, N. Shahriar, and R. Boutaba are with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sr2chowdhury@uwaterloo.ca; r5ahmed@uwaterloo.ca; nshahria@uwaterloo.ca; rboutaba@uwaterloo.ca).

S. Ayoubi is with INRIA, 75012 Paris, France (e-mail: sara.ayoubi@inria.fr).

J. Mitra is with Huawei Technologies Canada Research Center, Ottawa, ON K2K 3J1, Canada (e-mail: jeebak.mitra@huawei.com).

L. Liu is with Huawei Technologies Company, Ltd., Chengdu, China (e-mail: liuliu1@huawei.com).

Digital Object Identifier 10.1109/TNSM.2018.2834315

long turnaround time. However, with the advances in optical networking technologies [11] and centralized optical control plane [12]–[15], such provisioning tasks are more and more automated. Even then, one should not abuse such capability to sporadically establish new IP links since it remains more expensive than embedding virtual links on existing IP links. In this regard, we are faced with the following challenges: (i) strike a balance between obtaining a low cost VN embedding while minimizing the establishment of new IP links; (ii) simultaneously decide on whether to create an IP link or not and its embedding in the OTN.

In this paper, we study the problem of MUlti-Layer Virtual Network Embedding (*MULE*) focusing on IP-over-OTN substrate networks with the objective of minimizing total resource provisioning cost for embedding the VN while considering the possibility of establishing new IP links when necessary. Specifically, the contributions of this paper are as follows:

- *OPT-MULE*: An Integer Linear Program (ILP) formulation to find the optimal solution to *MULE*. The state-of-the-art in multi-layer VNE [7] does not optimally solve the problem. To the best of our knowledge, this is the first optimal solution the VNE problem for multi-layer IP-over-OTN networks.
- *FAST-MULE*: A heuristic to tackle the computational complexity of *OPT-MULE*. We also prove that our heuristic solves the problem optimally for a specific class of VNs, *i.e.,* star-shaped VNs. For arbitrary VNs, trace driven simulations show that *FAST-MULE* uses $\approx$1.47$\times$ more resources on average compared to *OPT-MULE* while executing several orders of magnitude faster. Further, our comparative analysis shows that *FAST-MULE* allocates $\approx$66% less resources on average compared to the state-of-the-art heuristic for multi-layer VNE [7], while accepting $\approx$60% more VN requests on average.

This paper extends our initial work presented in [16] on the following aspects. First, we provide a guideline on how to parallelize the proposed heuristic and leverage modern multi-core CPUs. Second, we perform more extensive performance evaluation of the proposed heuristic by performing a steady state analysis. The steady state analysis involves performance evaluation while considering arrival and departure of VNs over a longer period of time as opposed to performing micro-benchmarking for single VN instances. We compare the steady state performance of our proposed heuristic with that of the state-of-the-art heuristic for multi-layer VN embedding [7]. Finally, we present a more elaborate discussion on the research literature and contrast our contributions to the related works.

The rest of the paper is organized as follows. We begin with a discussion of related works in Section II. Then we introduce the mathematical notations representing the inputs to the problem and formally define the problem in Section III. In Section IV, we present *OPT-MULE*, an ILP formulation to optimally solve *MULE*, followed by our proposed heuristic, *FAST-MULE* in Section V. Our evaluation of the proposed solutions are presented in Section VI. Finally, we conclude with some future research directions in Section VII.

II. Related Works

A. Virtual Network Embedding

VNE is a well studied problem in network virtualization and a significant body of research has solved a number of its variants [4], [17]–[27]. However, it has been mostly studied for single layer SNs, *i.e.,* for IP, Optical or Wireless networks. Despite the existence of a significant number of proposals [28]–[30], VNE solutions for IP networks commonly involve allocating compute and bandwidth resources for the virtual nodes and links, respectively. In the case of optical networks, solving VNE involves allocating compute resources and wavelength for virtual nodes and links, respectively [31]. Optical networks have technological constraints such as discrete wavelength allocation, wavelength continuity *etc.* that add additional challenges to the VNE problem [32]. The state-of-the-art in optical network virtualization has mostly focused on single layer optical networks.

B. Multi-Layer Embedding

A few works in the research literature addressed the problem of embedding in multi-layer networks [7], [33], [34]. Savi *et al.* [33] consider the problem of application-aware traffic embedding on IP/Multi-layer Protocol Switching(MPLS)-over-Optical Network. Traffic requests are given between pairs of routers in the network, where each request has differentiated service requirements in terms of bandwidth, tolerable end-to-end delay, and tolerable end-to-end path availability. Here, the possibility of establishing new IP links is also considered when the IP/MPLS layer does not have sufficient capacity to meet the demand of a given request. However, in [33], the end points of the requests are fixed and it only addresses the link routing problem. Moreover, [33] neither does propose an optimal solution to the problem, nor presents complexity analysis of the proposed heuristic. Furthermore, the creation of new IP links is restricted to the pair of IP nodes that are not already connected. In [34], the problem of Service Function Chaining (SFC) in IP-over-OTN networks is addressed. This work considers service function chains distributed across multiple data centers, where the data centers, representing the electronic layer, are interconnected via an optical network. The authors propose an algorithm in [34] to route each SFC request across the data centers. Therefore, even though the substrate network is a multi-layer one, the routing problem is addressed only in for optical layer. Furthermore, the placement of the network functions in the requested SFC is considered given, and the different segments (pair of service functions) that make up the SFC are routed sequentially.

To the best of our knowledge, the only work that considered the problem of multi-layer virtual network embedding while considering both node and link embedding is presented in [7]. Zhang *et al.*, proposed a heuristic for solving the multi-layer VNE problem for IP-over-DWDM networks. They also consider the possibility of modifying IP layer topology by allocating wavelengths from the underlying DWDM network. Zhang *et al.*, proposed a two step embedding process that first embeds the virtual nodes then the virtual links, which limits the solution space and hence the optimality of the embedding. In contrast, we propose an ILP formulation for optimally solving the multi-layer VNE problem. Also, our heuristic does not embed the virtual nodes and links independently from each other, rather tries to embed them simultaneously.

C. Multi-Layer Network Optimization

An orthogonal but somehow related area of research in multi-layer network optimization focused on the issue of capacity planning in multi-layer networks [32], [35]. During the initial capacity planning a traffic matrix for the IP layer is given and sufficient capacity needs to be allocated in both IP and Optical layers to support that traffic matrix. Different variants of the problem exist that take different technological constraints and deployment models into account [36]–[39]. While the results from [36] are applicable for generic multi-layer network, however, that from [37] is specific to IP/MPLS-over-OTN-over-DWDM optical networks, with particular emphasis on the technological constraints of the OTN layer. Similar to [37], the work presented in [38] considers the capacity planning problem for OTN-over-DWDM networks.

Another research direction, that has been well explored in the research literature is that of protection planning for multi-layer networks [40]–[44]. Multi-layer protection planning involves deciding which layer will be in charge of protecting what, and coordinating the protection schemes across different layers [40], [45]. For instance, Gerstel *et al.* [45] showcase the limitations of traditional capacity planning in IP-over-OTN networks where each layer is treated in isolation. Subsequently, the authors motivate the advantages of coordinating across the different layers, and illustrate these benefits in Multi-Layer Restoration (MLR) planning. To achieve their goals, the authors compared MLR against restoration planning performed at the IP-layer alone and showed significant savings in the number of interfaces used and provisioned network resources (*i.e.,* wavelengths in their case). Bigos *et al.* [41] address the problem of designing a multi-layer protection scheme for MPLS-over-OTN networks. They evaluate single and multi-layer survivability schemes under different spare capacity allocation strategies (*e.g.,* shared vs. dedicated). In the single layer survivability scheme, they propose to protect every Label Switched Path (LSP) against failures in the IP or in the OTN layer. Whereas, in the multi-layer survivability scheme, the OTN layer is protected against physical link and OXC failures, and the IP layer is protected against routers and IP/Optical interface failures.

In contrast to capacity and protection planning, in multi-layer VNE, the endpoint of the demands, *i.e.,* virtual node placement, is not known in advance, making this one a fundamentally different problem. Further, the body of research in multi-layer capacity and protection planning has demonstrated clear advantages of resource allocation when the layers are jointly optimized as opposed to considering them in isolation [41], [45], [46]. Our solution approach also takes a joint optimization approach to the multi-layer VNE problem.

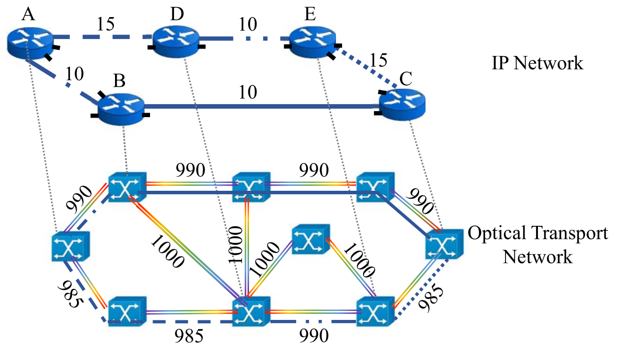


Fig. 1. MULE Illustrative Example.

III. MULE: Multi-Layer Virtual Network Embedding Problem

We first present a mathematical representation of the inputs, *i.e.,* the IP topology, the OTN topology, and the VN request. Then we give a formal definition of *MULE*, followed by an illustrative example.

A. Substrate Optical Transport Network (OTN)

We represent the substrate OTN as an undirected graph $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V}$ and $\hat{E}$ are the set of OTN capable devices (referred as OTN nodes in the remaining) and OTN links, respectively. Without loss of generality we assume the OTN links to be undirected, since such undirected OTN links can be either supported by specific technologies [47], [48] or by laying out multiple unidirectional fibers, or by using different wavelengths for sending and receiving within a single strand of fiber. Neighbors of an OTN node $\hat{u}$ are represented with $\mathcal{N}(\hat{u})$. We assume the OTN to be fixed, *i.e.,* light paths atop a DWDM layer have been already lit to provision OTN links $(\hat{u}, \hat{v})$ with bandwidth capacity $b_{\hat{u}\hat{v}}$. This pre-provisioned bandwidth can be used to establish IP links between IP routers. The cost of allocating one unit of bandwidth from an OTN link $(\hat{u}, \hat{v}) \in \hat{E}$ is $C_{\hat{u}\hat{v}}$. Fig. 1 illustrates an example of an OTN network, where the numbers on each link represent its residual capacity.

B. Substrate IP Network

The substrate IP network is an undirected graph $G' = (V', E')$. Each IP node $u' \in V'$ has $p_{u'}$ number of ports with homogeneous capacity $cap_{u'}$. Each IP node u' is connected to an OTN node $\tau(u')$ through a short-reach wavelength interface. Attachment between an IP and an OTN node is represented using a binary input variable $\tau_{u'\hat{u}}$, which is set to 1 only when IP node u' is attached to OTN node $\hat{u}$. An IP link is provisioned by establishing an OTN path that connects its end points. Note that, it is common in operator networks to establish multiple IP links between the same pair of IP nodes and bundle their capacities using some form of link aggregation protocol [49]. We also follow the same practice and use (u', v', i) to represent the i-th IP link between u' and v', where $1 \leq i \leq p_{u'}$. We use the binary input variable $\Gamma_{u'v'i}$ to indicate the existence of an IP link (u', v', i) in G'. $\Gamma_{u'v'i}$ is set to 1 when IP link (u', v', i) is present in G', otherwise it is set to 0. Bandwidth of an IP link is represented by $b_{u'v'i}$.

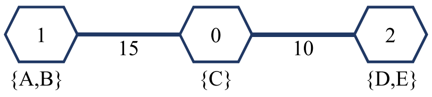


Fig. 2. Virtual Network.

Capacity of a new IP link *(u′, v′, i)* is set to $\min(cap_{u'}, cap_{v'})$. Fig. 1 illustrates an example IP network, where each IP link is mapped on an OTN path and the residual bandwidth capacity of an IP link is represented by the number on that link. The cost of allocating one unit of bandwidth from an IP link *(u′, v′, i)* $\in E'$ is $C_{u',v',i}$.

C. Virtual Network (VN)

A VN request is an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ and $\bar{E}$ are the set of virtual nodes (VNodes) and virtual links (VLinks), respectively. Each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ has a bandwidth requirement $b_{\bar{u}\bar{v}}$. Each VNode $\bar{u} \in \bar{V}$ has a location constraint set $\mathcal{L}(\bar{u}) \subset V'$ that represents the set of IP nodes where $\bar{u}$ can be embedded. $\mathcal{L}(\bar{u})$ can be determined by the InP based on geographical proximity requirement by the SP. Note that $\mathcal{L}(\bar{u})$ can contain all the IP nodes to represent an unconstrained scenario. We represent the location constraints using a binary input variable $\ell_{\bar{u}u'}$, which is set to 1 if IP node $u' \in \mathcal{L}(\bar{u})$. Fig. 2 illustrates a VN, where the number on each link represents VLink demand, and the set next to each node denotes that VNode's location constraints.

D. Problem Definition

Given a multi-layer SN composed of an IP network G' on top of an OTN network $\hat{G}$, and a VN request $\bar{G}$ with location constraint set $\mathcal{L}$:

- Map each VNode $\bar{u} \in \bar{V}$ to an IP node $u' \in V'$ according to the VNode's location constraint.
- Map each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ to a path in the IP network. This path can contain a combination of existing IP links and newly created IP links.
- Map all newly created IP links to a path in the OTN.
- The total cost of provisioning resources for new IP links and cost of provisioning resources for VLinks should be minimized subject to the following constraints:
  - IP links cannot be over-committed to accommodate the VLinks, and
  - the demand of a single VLink should be satisfied by a single IP path.

The embedding is subject to the constraints that both IP links and OTN links cannot be over-provisioned, and VLinks and IP links cannot be routed along multiple IP paths and multiple OTN paths, respectively (*i.e.,* no path splitting). Moreover, we do not consider neither VNode resource requirement nor VNode embedding cost. We assume that the virtualization enabled network devices have enough capacity to switch at line rate between any pair of ports and any complex control mechanism is decoupled and performed in a centralized control plane. Finally, we consider online version of the problem where VN requests arrive one at a time.

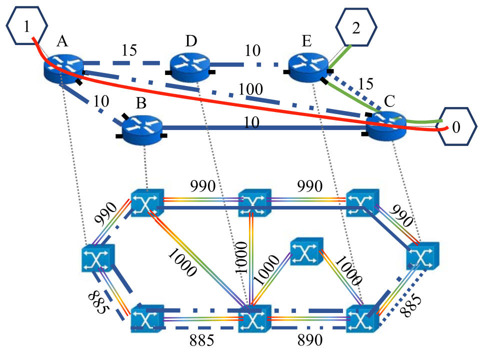


Fig. 3. Multi-Layer VN Embedding Example.

E. Illustrative Example

To better illustrate the problem and the underlying complexities, consider the case of embedding the VN presented in Fig. 2 over the multi-layer IP-over-OTN network in Fig. 1. Given the residual capacity of the IP links, clearly, there is not sufficient bandwidth in the IP network to route the VLink between VNodes 0 and 1. Hence, no feasible embedding of the VN exists. Indeed, if we were to place VNode 1 on either IP nodes A or B, and VNode 0 on IP node C (the only possible placement), we cannot route 15 units of bandwidth between IP nodes A and C, or B and C over an unsplittable path. In the single-layer embedding problem, such situation would have led to rejecting this VN. However, we can exploit the topological flexibility of multi-layer IP-over-OTN networks to establish new IP link(s) when there is insufficient capacity present in the IP layer. Fig. 3 illustrates and example solution of MULE where a new IP link has been provisioned between IP nodes A and C to accommodate the VLink between VNodes 0 and 1. The new IP link is supported by provisioning resources on the OTN path between the pair of OTN nodes connected to IP nodes A and C, respectively. This new IP link consumed an available port from IP nodes A and C. For the sake of simplicity, we assume uniform capacity of 100 units of bandwidth for all ports of an IP node. However, in the remainder of the paper we do not make any assumptions on the capacity of the IP ports.

Observe that in this example, there were several possibilities for provisioning the new IP link(s). For instance, if VNode 1 was embedded on IP node B instead of C, then the new IP link would have been between IP nodes B and C. Even when embedding VNode 1 on IP node A, two IP links could have been created to establish a path between IP nodes A and C through B. Hence, performing the node and link embedding separately impacts the cost of the resultant embedding solution, as well as the choice of new IP link(s) to setup. This stresses the need to jointly consider both VN embedding and provisioning of new IP links, which we will be addressing in the remainder of the paper.

IV. OPT-*MULE*: AN ILP FORMULATION

MULE jointly optimizes VN embedding, creation of new IP links and embedding of newly created IP links on the

TABLE I
SUMMARY OF KEY NOTATIONS

Inputs	
$\hat{G} = (\hat{V}, \hat{E})$	Substrate OTN
$b_{\hat{u}\hat{v}}$	Residual Bandwidth capacity of OTN link $(\hat{u}, \hat{v}) \in \hat{E}$
$C_{\hat{u}\hat{v}}$	Cost of allocating unit bandwidth on OTN link $(\hat{u}, \hat{v}) \in \hat{E}$ for provisioning an IP link
$G' = (V', E')$	Substrate IP Network
$\Gamma_{u'v'i} \in \{0,1\}$	$\Gamma_{u'v'i}$ = 1 if $(u', v', i) \in E'$
$b_{u'v'i}$	Residual Bandwidth capacity of IP link $(u', v', i) \in E'$
$C_{u'v'i}$	Cost of allocating unit bandwidth on IP link $(u', v', i) \in E'$ for provisioning a VLink
$p_{u'}$	Number of ports on IP node u'
$cap_{u'}$	Capacity of each port of IP node u'
$\tau_{u'\hat{u}} \in \{0, 1\}$	$\tau_{u'\hat{u}} = 1$ if IP node u' is attached to OTN node $\hat{u}$
$\bar{G} = (\bar{V}, \bar{E})$	Virtual Network Request
$b_{\bar{u}\bar{v}}$	Bandwidth requirement of VLink $(\bar{u}, \bar{v}) \in \bar{E}$
$\mathcal{L}(\bar{u})$	Location constraint set for VNode $\bar{u} \in \bar{V}$
$\ell_{\bar{u}u'} \in \{0, 1\}$	$\ell_{\bar{u}u'} = 1$ if $u \in \mathcal{L}(\bar{u})$, $u' \in V', \bar{u} \in \bar{V}$
Outputs	
$x^{\bar{u}\bar{v}}_{u'v'i} \in \{0, 1\}$	$x^{\bar{u}\bar{v}}_{u'v'i} = 1$ if $(u', v', i) \in E'$ is on the embedded IP path for $(\bar{u}, \bar{v}) \in \bar{E}$
$y_{\bar{u}u'} \in \{0, 1\}$	$y_{\bar{u}u'} = 1$ if $\bar{u} \in \bar{V}$ is mapped to $u' \in V'$
$\gamma_{u'v'i} \in \{0, 1\}$	$\gamma_{u'v'i} = 1$ if i-th IP link is created between u' and v'
$z^{u'v'i}_{\hat{u}\hat{v}} \in \{0, 1\}$	$z^{u'v'i}_{\hat{u}\hat{v}} = 1$ if $(\hat{u}, \hat{v}) \in \hat{E}$ is on the embedded OTN path for $(u', v', i) \in E'$

OTN layer. We present an Integer Linear Program (ILP) formulation for optimally solving *MULE*, namely OPT-*MULE*. We first introduce the decision variables used in our ILP (Section IV-A). Then we present our constraints (Section IV-B) followed by the objective function (Section IV-C). A list of key notations used in the ILP formulation is presented in Table I.

A. Decision Variables

A VLink must be mapped to a path in the IP network. The following decision variable indicates the mapping between a VLink $(\bar{u}, \bar{v}) \in \bar{E}$ and an IP link, $(u', v', i) \in E'$.

$$x^{\bar{u}\bar{v}}_{u'v'i} = \begin{cases} 1 & \text{if } (\bar{u}, \bar{v}) \in \bar{E} \text{ is mapped to } (u', v', i) \in E', \\ 0 & \text{otherwise.} \end{cases}$$

VNode mapping on IP node is denoted by:

$$y_{\bar{u}u'} = \begin{cases} 1 & \text{if } \bar{u} \in \bar{V} \text{ is mapped to } u' \in V', \\ 0 & \text{otherwise.} \end{cases}$$

The following decision variable determines the creation of new IP links:

$$\gamma_{u'v'i} = \begin{cases} 1 & \text{when } i\text{-th IP link is created between} \\ & \quad u' \text{ and } v', \\ 0 & \text{otherwise.} \end{cases}$$

Finally, a newly created IP link must be mapped to an OTN path. This mapping between such IP link and an OTN link is indicated by the following variable:

$$z^{u'v'i}_{\hat{u}\hat{v}} = \begin{cases} 1 & \text{if } (u', v', i) \in E' \text{ is mapped to } (\hat{u}, \hat{v}) \in \hat{E}, \\ 0 & \text{otherwise.} \end{cases}$$

In what follows, we use the notation V'^2 to denote the set of all pairs of IP nodes *(u', v')* such that $u' \neq v'$.

B. Constraints

1) VNode Mapping Constraint: Equations (1) and (2) ensure that each VNode is mapped to exactly one IP node according to the location constraints. Equation (3) restricts multiple VNodes to be mapped on the same IP Node.

$$\forall \bar{u} \in \bar{V}, \forall u' \in V' : y_{\bar{u}u'} \leq \ell_{\bar{u}u'} \tag{1}$$

$$\forall \bar{u} \in \bar{V} : \sum_{u' \in V'} y_{\bar{u}u'} = 1 \tag{2}$$

$$\forall u' \in V' : \sum_{\bar{u} \in \bar{V}} y_{\bar{u}u'} \leq 1. \tag{3}$$

2) VLink Mapping Constraints: Equation (4) ensures that VLinks are mapped only to existing or newly created IP links. Equation (5) ensures that each VLink is mapped to a non-empty subset of IP links. We prevent the formation of loops between parallel IP links by (6). Equation (7) prevents overcommitment of IP link bandwidth. Finally, (8), our flow-conservation constraint, ensures that VLinks are mapped on a continuous IP path.

$$\forall (\bar{u}, \bar{v}) \in \bar{E}, \forall \big(u', v'\big) \in V'^2, 1 \leq i \leq min(p_{u'}, p_{v'}) : x^{\bar{u}\bar{v}}_{u'v'i} \leq \gamma_{u'v'i} + \gamma_{v'u'i} + \Gamma_{u'v'i} \tag{4}$$

$$\forall (\bar{u}, \bar{v}) \in \bar{E} : \sum_{\forall (u', v') \in V'^2} \sum_{i=1}^{p_{u'}} x^{\bar{u}\bar{v}}_{u'v'i} \geq 1 \tag{5}$$

$$\forall (\bar{u}, \bar{v}) \in \bar{E}, \forall \big(u', v'\big) \in V'^2 : \sum_{i=1}^{p_{u'}} x^{\bar{u}\bar{v}}_{u'v'i} \leq 1 \tag{6}$$

$$\forall \big(u', v'\big) \in V'^2, 1 \leq i \leq p_{u'} : \sum_{\forall (\bar{u}, \bar{v}) \in \bar{E}} x^{\bar{u}\bar{v}}_{u'v'i} \times b_{\bar{u}\bar{v}} \leq b_{u'v'i} \tag{7}$$

$$\forall (\bar{u}, \bar{v}) \in \bar{E}, \forall u' \in V' : \sum_{\forall v' \in V'^2} \sum_{i=1}^{\min(p_{u'}, p_{v'})} \big(x^{\bar{u}\bar{v}}_{u'v'i} - x^{\bar{u}\bar{v}}_{v'u'i}\big) = y_{\bar{u}u'} - y_{\bar{v}u'}. \tag{8}$$

3) IP Link Creation Constraints: Equation (9) limits the number of incident IP links on an IP node to be within its available number of ports. Then, (10) ensures that a specific instance of IP link between a pair of IP nodes is either decided by the ILP or was part of the input, but not both at the same time.

$$\forall u' \in V' : \sum_{\forall v' \in V' | v' \neq u'} \sum_{i=1}^{\min(p_{u'}, p_{v'})} \gamma_{u'v'i} + \gamma_{v'u'i} + \Gamma_{u'v'i} \leq p_{u'} \quad (9)$$

$$\forall (u', v') \in V'^2, 1 \leq i \leq p_{u'} : \gamma_{u'v'i} + \Gamma_{u'v'i} \leq 1. \quad (10)$$

4) IP-to-OTN Link Mapping Constraints: First, we ensure, using (11), that only the newly created IP links are mapped on the OTN layer. Then, (12) is the flow conservation constraint that ensures continuity of the mapped OTN paths. Finally, (13) is our capacity constraint for OTN links.

$$\forall (u', v') \in V'^2, 1 \leq i \leq p_{u'}, (\hat{u}, \hat{v}) \in \hat{E} : z_{\hat{u}\hat{v}}^{u'v'i} \leq \gamma_{u'v'i} \quad (11)$$

$$\forall (u', v') \in V'^2, 1 \leq i \leq p_{u'}, \forall \hat{u} \in \hat{V} : \sum_{\forall \hat{v} \in \mathcal{N}(\hat{u})} \left(z_{\hat{u}\hat{v}}^{u'v'i} - z_{\hat{v}\hat{u}}^{u'v'i} \right) = \begin{cases} \gamma_{u'v'i} & \text{if } \tau_{u'\hat{u}} = 1, \\ -\gamma_{u'v'i} & \text{if } \tau_{v'\hat{u}} = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

$$\forall (\hat{u}, \hat{v}) \in \hat{E} : \sum_{\forall (u', v') \in V'^2} \sum_{i=1}^{p_{u'}} z_{\hat{u}\hat{v}}^{u'v'i} \times b_{u'v'i} \leq b_{\hat{u}\hat{v}}. \quad (13)$$

C. Objective Function

Our objective is to minimize the cost incurred by creating new IP links and also the cost of provisioning bandwidth for the VLinks. Cost for provisioning new IP links is computed as the cost of allocating bandwidth in the OTN paths for every new IP link. The cost of embedding a VN is computed as the total cost of provisioning bandwidth on the IP links for the VLinks. Our objective function is formulated as follows:

$$\text{minimize} \sum_{\forall (u', v') \in V'^2} \sum_{i=1}^{p_{u'}} \sum_{\forall (\hat{u}, \hat{v}) \in \hat{E}} z_{\hat{u}\hat{v}}^{u'v'i} \times b_{u'v'i} \times C_{\hat{u}\hat{v}} + \sum_{\forall (\bar{u}, \bar{v}) \in \bar{E}} \sum_{\forall (u', v') \in V'^2} \sum_{i=1}^{p_{u'}} x_{u'v'i'}^{\bar{u}\bar{v}} \times b_{\bar{u}\bar{v}} \times C_{u'v'i}. \quad (14)$$

D. Hardness of OPT-MULE

Consider the case where the IP layer has sufficient capacity to accommodate a given VN request. In this case, MULE becomes a single-layer VNE, which has been proven to be NP-Hard via a reduction from the multi-way separator problem [5]. Given that single-layer VNE is an instance of MULE, by restriction we conclude that MULE is also NP-Hard.

V. *FAST-MULE*: A Heuristic Approach

Given the NP-Hard nature of the multi-layer VNE problem and its intractability for large network instances, we propose FAST-*MULE*, a heuristic to solve the Multi-Layer VNE problem. We begin by explaining the challenges behind the design of FAST-*MULE* in Section V-A, followed by a description of its procedural details and an illustrative example in Section V-B and Section V-D, respectively. We analyze the running time of *FAST-MULE* in Section V-C. Then, we prove in Section V-E that *FAST-MULE* yields the optimal solution for star VN topologies with uniform bandwidth requirement. Finally, we provide a guideline on how to parallelize *FAST-MULE* for leveraging multiple CPU cores (Section V-F).

A. Challenges

1) Joint Mapping in IP and OTN Layers: One challenge of *MULE* is the fact that the embedding can take place in both layers. This occurs when a VN could not be accommodated by the existing IP links, and requires the creation of new ones. A plausible approach is to handle the embedding at each layer separately, *i.e.,* start by mapping the VN on the IP layer followed by mapping the new IP links on the OTN layer. Clearly, such disjoint embedding is far from optimal as there may not be sufficient bandwidth at the OTN level to accommodate the new IP links. To overcome this limitation, we equip *FAST-MULE* with the ability to consider both layers simultaneously when embedding a VN. This is achieved by collapsing the IP and OTN into a single layer graph, similar to [7]. Our collapsed graph contains all the IP and OTN nodes and links, as well as the links connecting IP nodes to OTN nodes. In contrast, [7] keeps the IP links and replaces the shortest paths in OTN with IP links that could have been created with those corresponding paths. In our case, a VLink embedding that contains OTN links indicates the creation of new IP links.

2) Joint VNode and VLink Embedding: Another challenge is to perform simultaneous embedding of a VNode and its incident VLinks. Embedding VNodes independently of their incident VLinks increases the chances of VN embedding failure. However, such joint embedding is hard to solve since it is equivalent to solving the NP-hard *Multi-commodity Unsplittable Flow with Unknown Sources and Destinations* [50]. Our goal is to equip FAST-*MULE* with the ability to perform joint embedding of VNodes along with their incident VLinks. To achieve this, we augment the collapsed graph with meta-nodes and modify its link capacities to convert the VNode and VLink embedding problem into a min-cost max-flow problem that we solve using Edmonds-Karp (EK) algorithm [51]. The flows returned by EK indicate both the VNodes and VLinks mapping. In what follows, we elucidate the details of this transformation along with how the embedding solution is extracted from the flows obtained from EK.

B. Heuristic Algorithm

Alg. 1 presents a high level view of *FAST-MULE*. From a very high level, the algorithm works as follows. First, we collapse the IP and OTN layers into a single-layer graph to perform joint optimization on both of the layers. Then, we incrementally embed the VN on the collapsed graph by extracting star subgraphs from the VN and jointly embedding the VNodes and VLinks of the star subgraph. We model the joint embedding problem as an instance of finding min-cost max-flow in the collapsed graph by setting appropriate flow

Algorithm 1: Multi-Layer VNE Algorithm

```
Input: $\hat{G} = (\hat{V},\hat{E})$, $G' = (V',E')$, $\bar{G} = (\bar{V},\bar{E})$
Output: Overlay Mapping Solution $\mathcal{M}$
function FAST-MULE()
    /*Initialize List of Settled Nodes*/
    $\mathcal{S} = \{\}$
    Step 1: Create Collapsed Graph
    $G$ = CreateCollapsedGraph($G'$,$\hat{G}$)
    forall $\bar{v} \in \bar{V}$ do
        if $\bar{v} \in \mathcal{S}$ then
            continue
        $\mathcal{S} = \mathcal{S} \cup \bar{v}$
        Step 2: Create Meta-Nodes
        $\mathcal{M}$.nmap = $\mathcal{M}$.nmap $\cup$ MapNode($\bar{v}$,$\mathcal{L}(\bar{v})$)
        for each ($\bar{u} \in \mathcal{N}(\bar{v})$) do
            if ($\bar{u}$ in $\mathcal{S}$) then
                continue
            if ($\mathcal{M}$.nmap($\bar{u}$) == NULL) then
                $V = V \cup$ CreateMetaNodes($\mathcal{L}(\bar{u})$)
            else
                $V = V \cup$ CreateMetaNodes($\mathcal{M}$.nmap($\bar{u}$))
        Step 3: Create Ref-Nodes
        $V = V \cup$ CreateRefNodes( $V$ )
        Step 4: Run Link Embedding Algorithm
        $\mathcal{M}$.emap = $\mathcal{M}$.emap $\cup$ EdmondsKarp($G$)
        $E = E \cup$ GetNewIPLinks($\mathcal{M}$.emap)
        $S = S \cup$ isSettled($\mathcal{N}(\bar{v})$)
    Return $\mathcal{M}$;
```

capacities and introducing additional meta-nodes and meta-links in the collapsed graph. We describe each of the phases from Alg. 1 in detail in the following.

Stage 1 (Creation of a Collapsed Graph): We begin by collapsing the OTN and IP networks to a single-layer graph to achieve a joint embedding across both the IP and the OTN layers (*i.e.,* to address the first challenge from Section V-A). The set of nodes in the collapsed graph contains all the IP and OTN nodes. The links in the collapsed graph consist of: (i) all the OTN links, (ii) added IP-to-OTN links (described later), and 3) all the IP links. We keep the residual capacities of the IP and OTN links as is. We assume the OTN links have significantly higher cost than the IP links. Therefore, new IP links are created only when they are really needed and can significantly reduce embedding cost. Finally, between every IP node u' and its corresponding OTN node $\tau(u')$, we create $p_{u'}$ links with capacity $cap_{u'}$. This guarantees that at most $p_{u'}$ new IP links can be created from u', and that their capacity cannot exceed node u''s port capacity.

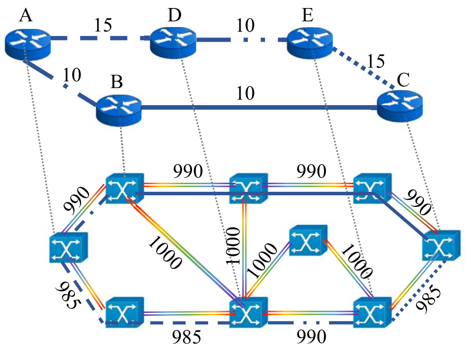
Stage 2 (Extraction of Star-Shaped Sub-Graphs From VN): Next, we randomly pick a VNode $\bar{v} \in \bar{V}$ and embed $\bar{v}$ with its incident VLinks. Embedding $\bar{v}$'s incident links entails embedding its neighbors as well. This means that we are embedding a star-shaped subgraph of the VN at each iteration. Incremental embedding of star subgraphs was performed to jointly embed nodes and links of the VN as much as possible (*i.e.,* to address the second challenge from Section V-A). To achieve this, we begin by mapping our current VNode $\bar{v}$, *i.e.,* the center of the star to a random IP node in its location constraint set (denoted as *source* in the following). Then we construct a flow network in such a way that the paths contributing to a min-cost max-flow in the flow network correspond to the embedding of the VLinks incident to $\bar{v}$.

Stage 3 (Addition of Meta-Nodes): We create a flow network by replacing every link in the collapsed graph with directional links in both directions. Then, $\forall \bar{u} \in \mathcal{N}(\bar{v})$, we add a meta-node in the flow network that we connect to every node in $\mathcal{L}(\bar{u})$. These meta-nodes are in-turn connected to a single meta-node, that we denote as the *sink*. After adding the meta-nodes we set the link capacities as follows:

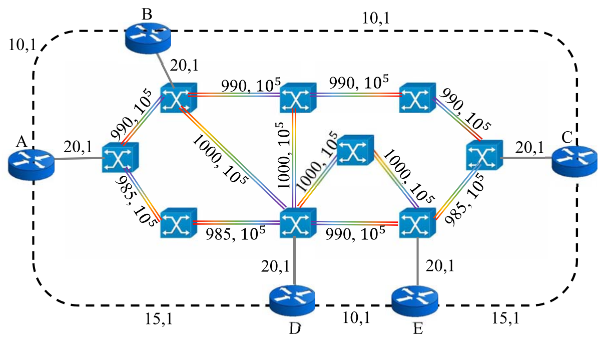
- We set the flow capacity of a link (*u, v*) from the collapsed graph that is not connected with any meta-node to $\frac{b_{uv}}{\max_{\forall \bar{u} \in \mathcal{N}(\bar{v})}(b_{\bar{u}\bar{v}})}$. Setting such capacity puts an upper limit on the maximum number of VLinks that can be routed through these links. Although this can lead to resource fragmentation and in the worst case rejection of a VN, it ensures that no capacity constraints are violated.
- We set the capacity of the links incident to a meta-node to 1. This guarantees that at most $|\mathcal{N}(\bar{v})|$ flows can be pushed from *source* to *sink.*

Stage 4 (Addition of Referee Nodes): Location constraint sets of different VNodes in a single VN may overlap. We denote such VNodes as *conflicting nodes* and the intersection of their location constraint sets as the *conflict set*. Every node in the conflict set is denoted as a conflict node. When conflicting VNodes are incident to the same *start* node, we end up with an augmented graph where all the nodes in the conflict set are connected to more than one meta-node. This is problematic because EK may end up routing multiple VLinks via the same conflict node, thereby violating the one-to-one node placement constraint. To resolve this issue, we introduce "Referee Nodes" (Ref-Nodes). Ref-Nodes are meta-nodes that are added to resolve the case of conflicting VNodes. In presence of a conflict, conflict nodes will be connected to more than one meta-node at the same time. Ref-Nodes are thus introduced to break this concurrency by removing the conflicting connections, and replacing them with a single connection to a Ref-node. The Ref-node is subsequently connected to all the meta-nodes of the conflicting nodes. This ensures that at most a single VLink will be routed through any conflict node. Further, when a conflict node is selected to host a given VNode, no other IP nodes for the same VNode will be selected, thereby ensuring an one-to-one assignment.

Stage 5 (Execution of the Edmonds-Karp Algorithm): Now we have an instance of the max-flow problem that we will solve using the Edmonds-Karp (EK) Algorithm []. We have set the capacity of the links in the flow network in such a way that EK can push at most $|\mathcal{N}(\bar{v})|$ flows, indicating the VLink embedding of $\bar{v}$'s incident links. Note that the only way to push $|\mathcal{N}(\bar{v})|$ flows is by having each flow traverse a unique meta-node to reach the *sink*. The VNode embedding of $\bar{v}$'s neighbors can be extracted by examining each flow to find the incident IP node of each meta-node. If any



(a) IP-over-OTN Graph



(b) Collapsed IP-OTN Graph

Fig. 4. Transformation from multi-layer to single-layer substrate network.

of the obtained flows is routed via an OTN path, then a new IP link is established and added to the collapsed graph. This allows subsequent iterations to use the newly created IP link. If at any iteration EK returns less than $|\mathcal{N}(\bar{v})|$ flows, this indicates an embedding failure, and the algorithm terminates. Otherwise, the algorithm returns to Stage 2 and repeats until all the VNodes are settled.

C. Running Time Analysis

We first introduce the following notations for running time analysis:

- I = the number of iterations of FAST-*MULE*
- $|V|$ = the number of nodes in the collapsed graph, where $|V| = O(|\hat{V}| + |V'|)$
- $|E|$ = the number of links in the collapsed graph, where $|E| = O(|\hat{E}| + |E'| + |V'|)$. The last element represents the number of IP-OTN links.

During each iteration (*i.e.,* embedding of a star subgraph from the VN), we execute the EK algorithm to find a min-cost max-flow in the collapsed graph. We replaced the augmenting path finding procedure of EK with Dijkstra's shortest path algorithm. Therefore, the running time of EK becomes $O(|V||E|^2 \log |V|)$. This renders the time complexity of our proposed approach to $O(I|V||E|^2 \log |V|)$. If we consider the worst-case scenario where the VN is in the form of a chain, and the nodes are traversed sequentially, then $I = |\bar{V}| - 1$, which results in a worst-case time complexity of $O(|\bar{V}||V||E|^2 \log |V|)$.

D. Illustrative Example

Fig. 4(b) illustrates how the IP-over-OTN graph in Fig. 4(a) has been converted into a collapsed IP-OTN graph. The collapsed graph is composed of the OTN nodes, OTN links, IP Nodes, IP-OTN links (represented by the single straight gray lines), and IP links (represented by the dashed black lines). Here, we assume that each IP node has a single residual port of capacity 20. The numbers on each link represent the capacity of the link followed by the cost of using this link. Observe that we set the cost of the IP links to 1, whereas the cost of the OTN links is set to a really high number to discourage the routing from passing through OTN links.

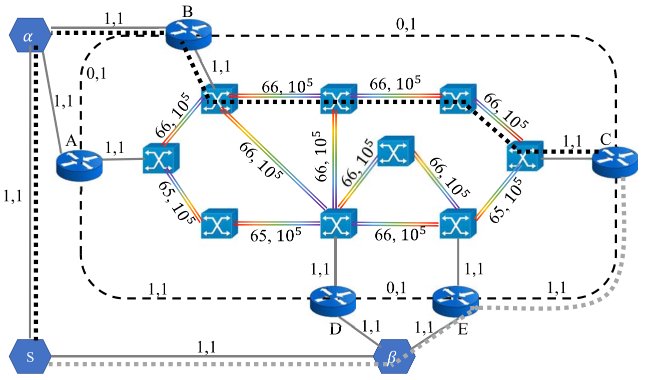


Fig. 5. Illustrative Example.

Next, we showcase how *FAST-MULE* embeds the VN in Fig. 2 atop the collapsed graph, as illustrated in Fig. 5. We consider that VNode 0 is the start node. Hence, the source node at this iteration of EK is IP node *C*. The sink node is meta-node *s* attached to the meta-nodes α and β of VNodes 1 and 2, respectively. Given that the maximum demand of VNode 0's incident links is 15, the capacity of each link in the collapsed graph (except links incident to meta-nodes whose capacity is fixed to 1) is replaced by the number of VLinks of capacity 15 it can accommodate. Running EK on the augmented graph (Stage 5) returns two flows between the source node *C* and the sink node *s*, indicated by the black and grey dotted lines in Fig. 5. Here, we observe that EK can only route VLink (0,2) via existing IP links (grey flow); whereas VLink (0,1) is routed through OTN links (black flow), thereby creating a new IP link (B,C) with capacity 20. Further, by examining the terminating IP nodes in every flow, we identify the VNode embedding of nodes 1 and 2 as IP nodes B and E, respectively.

E. Optimality of FAST-MULE for Star VN Topology

Recall that in Alg. 1, the joint node and link embeddings are executed iteratively on a subgraph of the VN until all the VNodes are settled. This iterative scheme renders a sub-optimal solution. However, if we could perform a joint node and link embedding on the entirety of the VN in a single iteration, that would guarantee that the obtained solution is indeed optimal. Such embedding is possible when all the nodes in the VN are only connected to a single node, and if the latter is selected as the start node, *i.e.,* the VN topology is a star. A star VN topology *S(N)* contains a center node $\bar{u}$ and *N* links connecting $\bar{u}$ to *N* leaf nodes $\{\bar{v}_1, \bar{v}_2, \ldots \bar{v}_N\}$. In the sequel, we prove that Alg. 1 can find the optimal solution in polynomial time when the VN request is a star topology (typically used to support multi-cast services [4]) with identical bandwidth demand on all VLinks.

Theorem 1: Given a star VN topology $\bar{G} = S(N)$ with uniform bandwidth demand β for all VLinks, Alg. 1 obtains the optimal solution in polynomial time.

Proof: The optimal embedding of $\bar{G}$, $\mathcal{M}^*$, is the one where the VNodes are placed on the IP nodes that provide the lowest cost link embedding. The cost includes both the cost of provisioning new IP links and the cost of allocating bandwidth for VLinks. We denote the cost of $\mathcal{M}^*$ as $\theta^* = \beta \sum_{i=1}^{N} \sum_{u'v' \in P_{\bar{u}\bar{v}_i}} C_{u'v'}$, where $P_{\bar{u}\bar{v}_i}$ is the embedding path for VLink $(\bar{u}, \bar{v}_i)$. Without loss of generality, we abstract a newly created IP link *(u′, v′)*'s cost as $C_{u'v'}$. Let $\mathcal{M}$ be the solution obtained by Alg. 1. For simplicity, we assume the central node $\bar{u}$ has exactly one IP node in its location constraint set. $\mathcal{M}$ consists of placing $\bar{u}$ on the IP node in its location constraint set, *v′*, followed by running EK from *v′* to the sink node *s*. EK will return the min-cost max-flow from *v′* to the sink node *s*. Given that the capacity of all the incident links to *s* are 1, the number of flow augmenting paths will be at most the number of leaf nodes in $\bar{G}$ and exactly 1 unit of flow will be pushed through each of these augmenting paths. Therefore, upon successful embedding, EK will return *N* flow augmenting paths with minimum cost θ. Now recall that the only way to push *N* flows towards the sink is to traverse every meta-node once; which entails the traversal of one node from each location constraint set. The traversed nodes represent the VNode embedding of all the leaf nodes in *S(N)*. Therefore, the flow augmenting paths represent a valid embedding of *S(N)*. We can characterize θ as, $\theta = \sum_{i=1}^{N} \sum_{(u,v) \in F_i} C_{uv} \times f_{uv}$, where F_i is the *i*-th flow augmenting path and f_{uv} is the flow pushed along link (*u, v*) in the flow network constructed from the collapsed graph. Note that, $f_{uv} = 1$, therefore, the cost becomes, $\theta = \sum_{i=1}^{N} \sum_{(u,v) \in F_i} C_{uv}$. If we can prove that $\sum_{i=1}^{N} \sum_{(u,v) \in F_i} C_{uv} = \sum_{i=1}^{N} \sum_{u'v' \in P_{\bar{u}\bar{v}_i}} C_{u'v'}$ then our proof is complete. Since θ^* is the optimal objective value, let, $\sum_{i=1}^{N} \sum_{(u,v) \in F_i} C_{uv} > \sum_{i=1}^{N} \sum_{u'v' \in P_{\bar{u}\bar{v}_i}} C_{u'v'}$. Then it implies that if we pushed the flows along the paths $\bigcup_{i=1}^{N} P_{\bar{u}\bar{v}_i}$ (the newly created IP links can be expanded to a set of OTN links to match the paths in the collapsed graph), we would have obtained a lower cost solution to min-cost max-flow problem, which contradicts that θ is the minimum cost of our min-cost max-flow problem for the converted flow network. Therefore, $\sum_{i=1}^{N} \sum_{(u,v) \in F_i} C_{uv} = \sum_{i=1}^{N} \sum_{u'v' \in P_{\bar{u}\bar{v}_i}} C_{u'v'}$, completing our proof. ■

If the central node, $\bar{u}$, has more than one candidate node in its location constraint set, then running Alg. 1 $|\mathcal{L}(\bar{u})|$ times is sufficient to obtain the lowest cost mapping solution, and the running time of Alg. 1 still remains polynomial.

F. Parallel Implementation of FAST-MULE

Note that during the execution of stage 2 in FAST-*MULE*, *i.e.,* during the extraction of star-subgraphs from the VN, we randomly chose a VNode as the center node of the extracted star graph. Indeed, the order in which the VNodes are chosen for star subgraph extraction has an impact on the performance of the heuristic. Therefore, we propose to execute the heuristic for a set of VNode orderings and choose the least cost one from the resulting solutions. Clearly, this means increasing the order of complexity for FAST-*MULE*.

One way to consider different VNode ordering in FAST-*MULE* is to consider the different VNode orders in parallel, *i.e.,* implement FAST-*MULE* as a multi-threaded program to utilize the multiple CPU cores on modern machines. Each thread of execution computes a solution to *MULE* by taking a VNode ordering as an input. Since, one execution of FAST-*MULE* for one VNode ordering is independent of another execution with a different VNode ordering, therefore, they can be run in parallel without requiring any synchronization between the threads. After the parallel executions finish, we can choose the best embedding, *i.e.,* the least cost embedding among all the parallel executions.

VI. Evaluation Results

We evaluate our proposed solutions for *MULE* through simulations. Due to the lack of publicly available real world multi-layer network topologies, we resort to generating synthetic topologies with varying sizes for our performance evaluation. We first describe our simulation setup in Section VI-A and the evaluation metrics in Section VI-B. Then we present our evaluation results based on the following two scenarios: (i) micro-benchmarking of FAST-*MULE* by comparing with the optimal solution and to D-VNE [7], the state-of-the-art heuristic for solving multi-layer VNE problem (Section VI-C), and (ii) steady-state analysis of the performance of FAST-*MULE* and comparison with that of D-VNE [7] (Section VI-D). For the micro-benchmarking scenario, we consider the VN requests in isolation, assuming each VN request can be successfully embedded on the SN. Micro-benchmarking allows us to measure how resource efficient is FAST-*MULE* compared to the optimal solution and to D-VNE [7]. In contrast, for the steady-state scenario, we consider VN arrival and departure over a period of time and consider the possibility of failing to embed VN requests on the SN. The steady-state analysis gives insights on substrate resource utilization over a longer period of time.

A. Simulation Setup

1) Testbed: We have implemented *OPT-MULE* and *FAST-MULE* using IBM ILOG CPLEX 12.5 C++ libraries and Java, respectively. *OPT-MULE* was run on a machine with 4×8 core 2.4Ghz Intel Xeon E5-4640 CPU and 512GB of memory, whereas, we used a machine with 2×8 core 2Ghz Intel Xeon E5-2650 CPU and 256GB memory to evaluate *FAST-MULE* We used a home-grown discrete event simulator to simulate the arrival and departure of VNs for the steady state scenario.

2) Multi-Layer IP-Over-OTN Topology: As mentioned earlier, due to the lack of publicly available real multi-layer network topologies, we resorted to synthetically generating the multi-layer SN topologies. For the micro-benchmarking scenario, we generated OTNs by varying the size between 15–100 nodes. For each OTN, we generated an IP topology with a node count of 60% of that of the OTN. Each node in the IP topology was attached to exactly one node in the OTN topology. For both the OTN and the IP topologies, we set a link generation probability to match their average node degree to known ISP topologies [52]. For the steady state scenario, we generated a larger SN topology with a 150 node OTN and 90 node IP network. Choice of such a size is based on the average size of known ISP networks found in [52]. The link generation probability was again chosen to ensure node degrees are similar to known ISP topologies. For both scenarios, OTN links were assigned a capacity of 100Gbps, while IP links were assigned a capacity randomly chosen between 10–20Gbps. Finally, we used a constrained shortest-path algorithm to map the input IP links over OTN paths.

3) VN Topology: For the micro-benchmarking scenario, we generated 20 VNs for each combination of IP and OTN, each VN with 4–8 VNodes. For the steady state case, we varied the size of the VN between 4–15 VNodes. For both scenarios, we set a 50% probability of having a link between every pair of VNodes. VLink capacities were randomly set between 50%–100% of that of the IP links. For each VNode, we generated a location constraint set by randomly selecting an IP node and including all the IP nodes withing its 3-hop reach. For random graph generation (both the VN and the SN) we used Erdos – Renyi method [53].

We evaluated the arrival and departure of VNs in the steady state scenario by simulating a Poisson process. We varied the VN arrival rate between 4 to 10 VNs per 100 time units, with a VN lifetime exponentially distributed with a mean of 1000 time units. These chosen set of parameters conforms with the ones used in [5], [20], and [26].

B. Evaluation Metrics

1) Cost Ratio: This is the ratio of costs obtained by two different approaches for solving the same problem instance, where cost is computed using (14). Cost ratio measures the relative performance of two approaches.

2) Execution Time: The time required for an algorithm to solve one instance of *MULE*.

3) Acceptance Ratio: The fraction of VN requests that have been successfully embedded on the SN over all the VN requests.

4) Utilization: The Utilization of an IP link is computed as the ratio of total bandwidth allocated to the embedded VLinks to that IP links capacity

5) Embedding Path Length: The length of IP (or OTN) path corresponding to a VLink's (or new IP link's) embedding.

C. Micro-Benchmarking Results

We focus our micro-benchmarking on the following aspects: (i) cost comparison between *FAST-MULE* and *OPT-MULE* to evaluate how well *FAST-MULE* compares to the optimal, (ii) impact of VNode ordering on *FAST-MULE*'s performance, and (iii) comparison of *FAST-MULE* with the state-of-the-art heuristic [7] for solving multi-layer VNE problem.

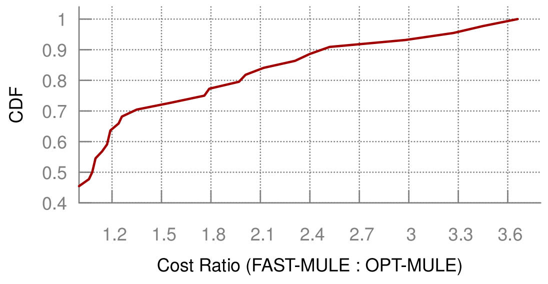


Fig. 6. FAST-*MULE* to OPT-*MULE* Cost Ratio.

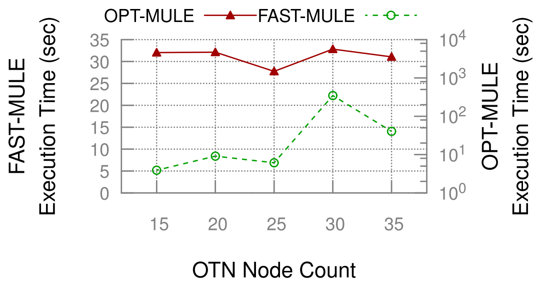


Fig. 7. Comparison of Execution Time.

1) Comparison of FAST-MULE With OPT-MULE: First, we empirically measure the extent of additional resources allocated by *FAST-MULE* compared to *OPT-MULE*. Our cost function is proportional to the total bandwidth allocated for a VN and the new IP links. Therefore, cost ratio of *FAST-MULE* to *OPT-MULE* gives the extent of additional resources allocated by *FAST-MULE*. Fig. 6 shows the Cumulative Distribution Function (CDF) of cost ratio between *FAST-MULE* and *OPT-MULE*. Note that, *OPT-MULE* scaled up to only 35-node OTN. To mitigate the impact of VNode ordering during embedding, we run *FAST-MULE* 75 times, each time with a different VNode embedding order and take the best solution at the end. We observe from the results that the end. We observe from the results that 50% of the VNs admitted by FAST-*MULE* have an embedding cost within 10% of the optimal solution. On average, the admitted VNs have a cost within 1.47× of that of the optimal solution. These results are indeed promising given that *FAST-MULE* achieves this while executing 440× faster than *OPT-MULE* on average (10s for *FAST-MULE* vs. >1hr per VN for *OPT-MULE*).

To further showcase the advantage of FAST-*MULE* compared to *OPT-MULE*, we plot their execution times against varying SN size in Fig. 7. For similar problem instances in our evaluation, *FAST-MULE* executed 200× to 900× faster than *OPT-MULE*. Even after increasing the SN size, the execution time of *FAST-MULE* remained in the order of tens of seconds.

2) Trade-Off Between Cost Ratio and Execution Time: We also evaluated the impact of the number of VNode orderings considered for the embedding. We present the results in Fig. 8, which shows how increasing the number of considered VNode

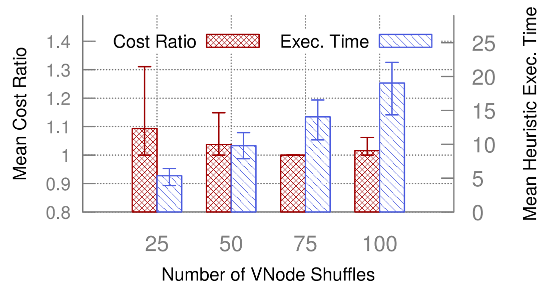
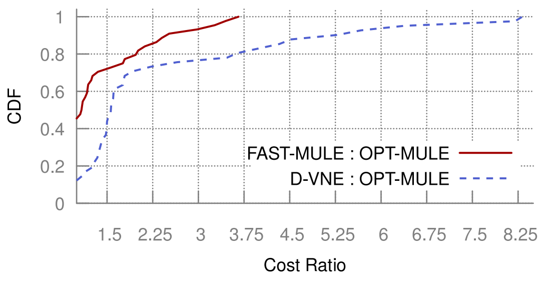
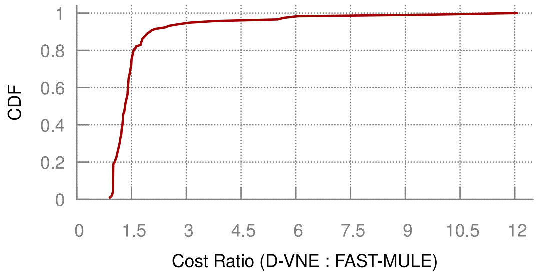


Fig. 8. Impact of VNode Shuffle on *FAST-MULE*'s Performance.



(a) Comparison with OPT-*MULE*



(b) D-VNE to FAST-*MULE* Cost Ratio

Fig. 9. Comparison between FAST-*MULE* and D-VNE [7].

orderings impacts the cost ratio and the execution time of *FAST-MULE*. Clearly, as we increase the number of considered VNode orderings, *FAST-MULE* to *OPT-MULE* cost ratio decreases. This comes at the expense of increased execution time, which still remains in the order of tens of seconds. However, the gain becomes marginal as we go beyond 75 iterations. Hence, in our evaluation we opt for feeding *FAST-MULE* with 75 VNode orderings and select the best solution.

3) Comparison of FAST-MULE With D-VNE [7]: Now, we evaluate how well FAST-*MULE* performs compared to the state-of-the art heuristic for multi-layer VNE [7]. We refer to [7] by D-VNE in the remaining. D-VNE constructs an auxiliary graph from the IP and Optical layers. The auxiliary graph contains precomputed optical paths that can be potentially chosen for creating new IP links. In contrast, we do not precompute paths in the OTN layer and let the embedding decide the best set of paths for jointly embedding VLinks and possible new IP links. D-VNE first embeds the VNodes using a greedy matching approach and then uses shortest path algorithm to route the VLinks between embedded VNodes. We

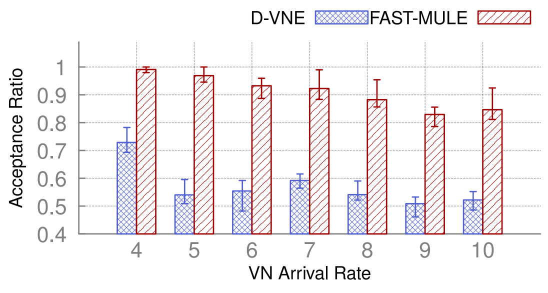


Fig. 10. VN Acceptance Ratio.

modified D-VNE to fit to our context where we do not perform wavelength allocation and omit node resource requirements.

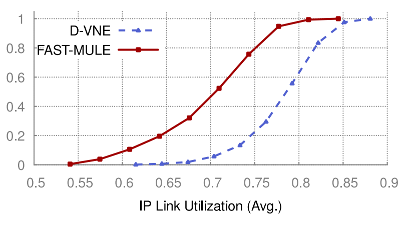
We begin by evaluating the cost ratio of D-VNE to *OPT-MULE* (Fig. 9(a)). The performance gap between D-VNE and *FAST-MULE* is evident from Fig. 9(a). D-VNE could embed VNs within 1.5× the cost of the optimal for ≈40% of the cases, whereas, *FAST-MULE* remains within the same bound for more than 70% of the cases. A head-to-head comparison between D-VNE and FAST-*MULE* is presented in Fig. 9(b). We observe that on average D-VNE allocates ≈66% more resources compared to *FAST-MULE*. These results reflect the advantage of a joint embedding scheme compared to a disjoint approach adopted by D-VNE.

D. Steady State Analysis

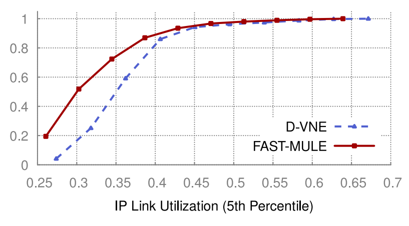
We perform a steady state analysis using the VN arrival rate and duration parameters described in Section VI-A for a total of 10000 time units. The total number of VNs across the simulations were varied between 400 - 950. The steady state performance analysis is focused on the following aspects: (i) comparing the acceptance ratio obtained by FAST-*MULE* to that of D-VNE under different loads, (ii) analyze and compare the load distribution on the SN, and (iii) analyze topological properties of the solution.

1) Acceptance Ratio: In this section, we present results on the acceptance ratio obtained by *FAST-MULE* and compare that with the acceptance ratio obtained by using D-VNE [7]. We consider the first 1000 time units of the simulation as the warm up period and discard the values from this duration. We compute the mean of the acceptance ratio obtained during the rest of the simulation period and report it along with the 5th and 95th percentile values against different VN arrival rates in Fig. 10. The results show that *FAST-MULE* outperforms D-VNE in all cases and accepts at least ≈37.5% more VNs over all cases. When the system load is increased, *i.e.,* for higher acceptance ratio, this gap is even bigger. For instance, when VN arrival rate is 10 VNs/100 time unit, *FAST-MULE* accepts ≈78%× more VNs compared to D-VNE.

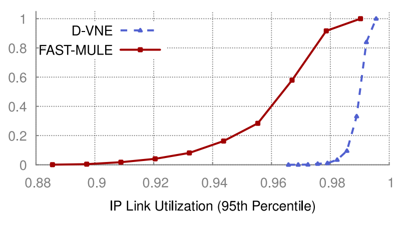
There can be several contributing factors to such behavior. For instance, one possibility is that the SN is becoming saturated (due to sub-optimal embedding with longer embedding paths), hence, VN requests are being rejected more often. Another possibility is that sufficient capacity in the SN is available, however, an algorithm is unable to exploit the available capacity or exploit the topological flexibility offered by the multi-layer network. In the following, we analyze how



(a) CDF of IP Link Utilization (Avg.)



(b) CDF of IP Link Utilization (5th Percentile)



(c) CDF of IP Link Utilization (95th Percentile)

Fig. 11. Load Distribution at the IP Layer.

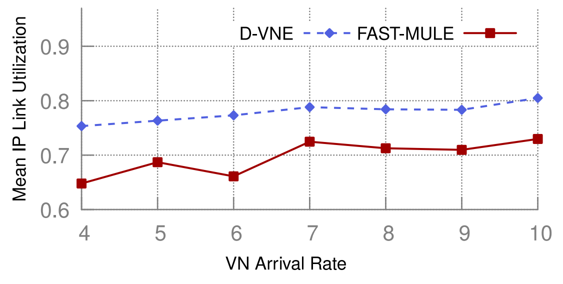


Fig. 12. Mean IP Link Utilization with Varying Load.

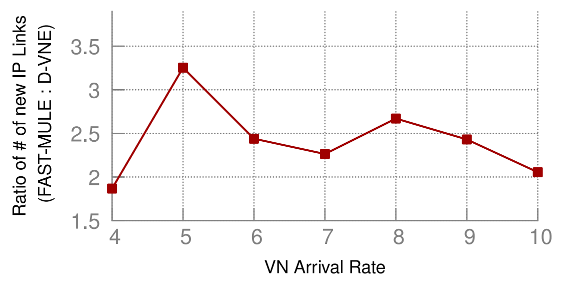
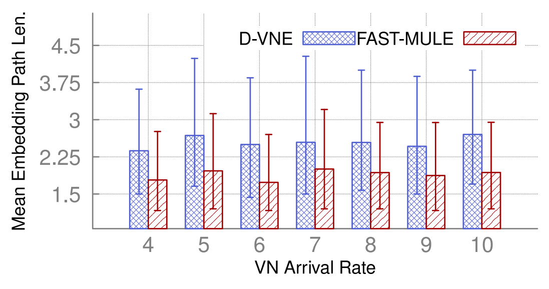
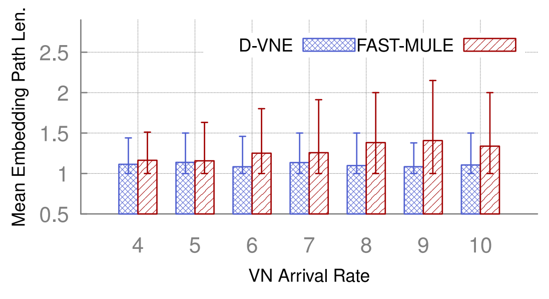


Fig. 13. Ratio of Newly Created IP Links (*FAST-MULE* : D-VNE) with Varying Load.



(a) VN to IP Mapping



(b) IP to OTN Mapping

Fig. 14. Mean Embedding Path Length.

these algorithms distribute load over the substrate and how much they are able to exploit the topological flexibility to gain further insight into the difference in acceptance ratio.

2) Load Distribution: We measure the utilization of IP links at each VN arrival and departure event. We present the mean IP link utilization for varying load (*i.e.,* VN arrival rate) in Fig. 12. One interesting observation is that, although D-VNE yields a lower acceptance ratio, it exhibited a higher mean link utilization compared to *FAST-MULE* ($\approx$10% more). However, this plot does not capture the variance in link utilization and does not say much about how the load is distributed over the IP links.

We present a further break down of IP link utilization in Fig. 11. Specifically, where we present the Cumulative Distribution Function (CDF) mean, 5th percentile, and 95th percentile link utilizations in Fig. 11(a), Fig. 11(b), and Fig. 11(c), respectively. The results for load distribution is also consistent with that from Fig. 12, *i.e.,* at all the spectrum of distribution D-VNE is exhibiting slightly higher link utilization while yielding lower acceptance ratio.

Another aspect that can also be tributary to such behavior is the extent to which the algorithms are exploiting the topological flexibility of multi-layer networks. After the end of each simulation we counted the total number of new IP links that were established by *FAST-MULE* and D-VNE, respectively, and present the ratio of these numbers in Fig. 13. As we can observe, *FAST-MULE* created more IP links compared to D-VNE and hence was able to accept more VNs in the long run. Because of the higher number of IP links, the graph diameter reduced and resulted in possibly shorter embedding paths, hence, lower utilization of individual links. Because of the joint optimization approach, *FAST-MULE* was able to make better decisions regarding creation of new IP links and also for embedding paths, hence, the acceptance ratio and lower link utilization.

3) Topological Properties of the Solution: For each simulation setting, we computed the mean embedding path length for both the VLinks and the newly created IP links and present the result in Fig. 14. We observe from Fig. 14(a) that *FAST-MULE* embedded the VLinks on shorter paths ($\approx$30%) compared to D-VNE. This is a combined effect of being able to create more IP links on the long run as well as the joint embedding of

VNodes and VLinks whenever possible. We also present the mean embedding path length for the newly created IP links in Fig. 14(b). Since the OTN is static and *FAST-MULE* established significantly more new IP links compared to that of D-VNE, IP link embedding paths became longer in case of *FAST-MULE*.

VII. Conclusion

This paper studied MULE, *i.e.,* multi-layer virtual network embedding on an IP-over-OTN substrate network. We proposed an ILP formulation, *OPT-MULE*, for optimally solving *MULE* and a heuristic, *FAST-MULE*, to address the computational complexity of the ILP. To the best of our knowledge, this is the first optimal solution to multi-layer VNE. Our evaluation of *FAST-MULE* shows that it performs within 1.47$\times$ of the optimal solution on average. *FAST-MULE* also outperformed the state-of-the-art heuristic for multi-layer VNE and allocated $\approx$66% less resources on average while accepting $\approx$60% more VN requests on average. Finally, we also proved that our proposed heuristic obtains optimal solution for star shaped VNs with uniform bandwidth demand in polynomial time.

We hope that this first endeavor will stimulate further research in multi-layer network virtualization. One possible future direction is to consider a dynamic OTN where more capacity can be provisioned by establishing new light paths in the underlying DWDM. Technological constraints posed by different optical network technologies such as wavelength continuity of DWDM networks or sub-wavelength resource allocation capabilities of elastic optical networks [31] are other interesting directions worth exploring in the future.

Acknowledgment

The authors would like to thank the anonymous reviewers of IEEE/ACM/IFIP CNSM 2017 for their valuable feedback.

References

[1] X. Zhao, V. Vusirikala, B. Koley, V. Kamalov, and T. Hofmeister, "The prospect of inter-data-center optical networks," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 32–38, Sep. 2013.

[2] N. Ghani, S. Dixit, and T.-S. Wang, "On IP-over-WDM integration," *IEEE Commun. Mag.*, vol. 38, no. 3, pp. 72–84, Mar. 2000.

[3] J. F. Botero *et al.*, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.

[4] 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.

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

[6] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[7] J. Zhang *et al.*, "Dynamic virtual network embedding over multilayer optical networks," *J. Opt. Commun. Netw.*, vol. 7, no. 9, pp. 918–927, Sep. 2015.

[8] X. Jin *et al.*, "Optimizing bulk transfers with software-defined optical WAN," in *Proc. ACM SIGCOMM*, Florianópolis, Brazil, 2016, pp. 87–100.

[9] F. Rambach *et al.*, "A multilayer cost model for metro/core networks," *J. Opt. Commun. Netw.*, vol. 5, no. 3, pp. 210–225, Mar. 2013.

[10] "Interfaces for the optical transport network," Int. Telecommun. Union, Geneva, Switzerland, Rep. g.709/y.1331, 2016. [Online]. Available: http://www.itu.int/rec/T-REC-G.709/

[11] A. L. Chiu *et al.*, "Architectures and protocols for capacity efficient, highly dynamic and highly resilient core networks [invited]," *J. Opt. Commun. Netw.*, vol. 4, no. 1, pp. 1–14, Jan. 2012.

[12] *OpenFlow-Enabled Transport SDN*. Accessed: Feb. 2018. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/05/sb-of-enabled-transport-sdn.pdf

[13] C. Janz, L. Ong, K. Sethuraman, and V. Shukla, "Emerging transport SDN architecture and use cases," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 116–121, Oct. 2016.

[14] *Cisco nLight Technology: A Multi-Layer Control Plane Architecture for IP and Optical Convergence*. Accessed: Feb. 2018. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3750-series-switches/whitepaper_c11-718852.html

[15] S. Dahlfort and D. Caviglia, "IP-optical convergence: A complete solution," *Ericsson Rev.*, vol. 9, no. 1, pp. 34–40, 2014.

[16] S. R. Chowdhury *et al.*, "MULE: Multi-layer virtual network embedding," in *Proc. IEEE/ACM/IFIP CNSM*, Tokyo, Japan, 2017, pp. 1–9.

[17] M. Chowdhury, F. Samuel, and R. Boutaba, "PolyViNE: Policy-based virtual network embedding across multiple domains," in *Proc. ACM VISA Workshop*, 2010, pp. 49–56.

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

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

[20] M. R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 2, pp. 105–118, Jun. 2013.

[21] M. F. Zhani, Q. Zhang, G. Simona, and R. Boutaba, "VDC planner: Dynamic migration-aware virtual data center embedding for clouds," in *Proc. IFIP/IEEE IM*, Ghent, Belgium, 2013, pp. 18–25.

[22] Z. Zhang *et al.*, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.

[23] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.

[24] N. Shahriar *et al.*, "Connectivity-aware virtual network embedding," in *Proc. IFIP Netw.*, Vienna, Austria, May 2016, pp. 46–54.

[25] S. Ayoubi, Y. Chen, and C. Assi, "Towards promoting backup-sharing in survivable virtual network design," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 3218–3231, Oct. 2016.

[26] S. R. Chowdhury *et al.*, "Dedicated protection for survivable virtual network embedding," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 913–926, Dec. 2016.

[27] N. Shahriar *et al.*, "Virtual network survivability through joint spare capacity allocation and embedding," *IEEE J. Sel. Areas Commun.*, to be published.

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

[29] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.

[30] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[31] G. Zhang, M. De Leenheer, A. Morea, and B. Mukherjee, "A survey on OFDM-based elastic core optical networking," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 65–87, 1st Quart., 2013.

[32] R. Nejabati, E. Escalona, S. Peng, and D. Simeonidou, "Optical network virtualization," in *Proc. ONDM*, Bologna, Italy, Feb. 2011, pp. 1–5.

[33] M. Savi *et al.*, "Benefits of multi-layer application-aware resource allocation and optimization," in *Proc. IEEE EuCNC*, Oulu, Finland, 2017, pp. 1–5.

[34] Y. Li, H. Li, Y. Liu, and Y. Ji, "Multi-layer service function chaining scheduling based on auxiliary graph in IP over optical network," *Proc. SPIE*, vol. 10464, Oct. 2017, Art. no. 1046424.

[35] Ć. Rožić, D. Klonidis, and I. Tomkos, "A survey of multi-layer network optimization," in *Proc. ONDM*, Cartagena, Spain, 2016, pp. 1–6.

[36] M. Duelli, E. Weber, and M. Menth, "A generic algorithm for CAPEX-aware multi-layer network design," in *Proc. ITG Symp. Photon. Netw. (VDE)*, Leipzig, Germany, 2009, pp. 1–8.

[37] I. Katib and D. Medhi, "IP/MPLS-over-OTN-over-DWDM multilayer networks: An integrated three-layer capacity optimization model, a heuristic, and a study," *IEEE Trans. Netw. Service Manag.*, vol. 9, no. 3, pp. 240–253, Sep. 2012.

[38] C. Govardan *et al.*, “A heuristic algorithm for network optimization of OTN over DWDM network,” in *Proc. IEEE ANTS*, 2015, pp. 1–6.

[39] E. Palkopoulou, D. A. Schupke, and T. Bauschert, “Energy efficiency and CAPEX minimization for backbone network planning: Is there a tradeoff?” in *Proc. IEEE ANTS*, New Delhi, India, 2009, pp. 1–3.

[40] H. Zhang and A. Durresi, “Differentiated multi-layer survivability in IP/WDM networks,” in *Proc. IEEE/IFIP NOMS*, Florence, Italy, 2002, pp. 681–694.

[41] W. Bigos, B. Cousin, S. Gosselin, M. Le Foll, and H. Nakajima, “Survivable MPLS over optical transport networks: Cost and resource usage analysis,” *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, pp. 949–962, Jun. 2007.

[42] W. Lu, X. Yin, X. Cheng, and Z. Zhu, “On cost-efficient integrated multilayer protection planning in IP-over-EONs,” *J. Lightw. Technol.*, vol. 36, no. 10, pp. 2037–2048, May 15, 2018.

[43] A. Alashaikh, D. Tipper, and T. Gomes, “Supporting differentiated resilience classes in multilayer networks,” in *Proc. IEEE DRCN*, Paris, France, 2016, pp. 31–38.

[44] M. Tornatore, D. Lucerna, B. Mukherjee, and A. Pattavina, “Multilayer protection with availability guarantees in optical WDM networks,” *J. Netw. Syst. Manag.*, vol. 20, no. 1, pp. 34–55, 2012.

[45] O. Gerstel *et al.*, “Multi-layer capacity planning for IP-optical networks,” *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 44–51, Jan. 2014.

[46] P. Demeester *et al.*, “Resilience in multilayer networks,” *IEEE Commun. Mag.*, vol. 37, no. 8, pp. 70–76, Aug. 1999.

[47] D. BianchiAn, G. Parthasarathy, and Y. Xu, “OTN system and method for supporting single-fiber bidirectional transmission of supervisory channel light,” WO Patent WO2 015 127 780, Sep. 2015. [Online]. Available: https://patents.google.com/patent/WO2015127780A1/en

[48] S. Chen *et al.*, “Full-duplex bidirectional data transmission link using twisted lights multiplexing over 1.1-km orbital angular momentum fiber,” *Sci. Rep.*, vol. 6, Nov. 2016, Art. no. 38181.

[49] *Link Aggregation Control Protocol*. Accessed: Feb. 2018. [Online]. Available: http://www.ieee802.org/3/ad/public/mar99/seaman_1_0399.pdf

[50] Y. Dinitz, N. Garg, and M. X. Goemans, “On the single-source unsplittable flow problem,” in *Proc. IEEE FOCS*, 1998, pp. 290–299.

[51] J. Edmonds and R. M. Karp, “Theoretical improvements in algorithmic efficiency for network flow problems,” *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[52] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with rocketfuel,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 133–145, 2002.

[53] P. Erdös and A. Rényi, “On random graphs, I,” *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

Shihabur Rahman Chowdhury (S’13) received the B.Sc. degree in computer science and engineering from the Bangladesh University of Engineering and Technology. He is currently pursuing the Ph.D. degree with the David R. Cheriton School of Computer Science, University of Waterloo. His research interests include virtualization and softwarization of computer networks. He was a recipient of several scholarships and awards, including the Best Paper Award at IEEE/ACM/IFIP CNSM 2017, the MITACS Globalink Research Award, the Ontario Graduate Scholarship, President’s Graduate Scholarship, GoBell Scholarship, and Graduate Excellence Scholarship in Computer Science with the University of Waterloo.

Sara Ayoubi received the Ph.D. degree in information and systems engineering from Concordia University. In 2016, she was a Post-Doctoral Fellow with the Cheriton School of Computer Science, University of Waterloo. She is currently a Post-Doctoral Fellow with Inria, Paris. She is the Co-Founder of the Montreal Operations Research Student Chapter. Her research interests are in the fields of operations research, networks, and computer systems. She was a recipient of several awards, including the Dissertation Prize for Engineering and Computer Science and the Best paper Award at IEEE CloudNet 2015. She was a co-recipient of the Best Paper Award at IEEE/IFIP CNSM 2017, and selected as a rising star in EECS in 2017.

Reaz Ahmed received the B.Sc. and M.Sc. degrees in computer science from the Bangladesh University of Engineering and Technology in 2000 and 2002, respectively, and the Ph.D. degree in computer science from the University of Waterloo in 2007. His research interests include future Internet architectures, information-centric networks, network virtualization, and content sharing peer-to-peer networks with focus on search flexibility, efficiency, and robustness. He was a recipient of the IEEE Fred W. Ellersick Award in 2008.

Nashid Shahriar (S’16) received the B.Sc. and M.Sc. degrees in computer science and engineering from the Bangladesh University of Engineering and Technology in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science, University of Waterloo. His research interests include network virtualization, 5G networks, and network reliability. He was a recipient of Ontario Graduate Scholarship, President’s Graduate Scholarship, and David R. Cheriton Graduate Scholarship with the University of Waterloo.

Raouf Boutaba (F’12) received the M.Sc. and Ph.D. degrees in computer science from University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a Professor of computer science with the University of Waterloo. His research interests include resource and service management in networks and distributed systems. He was a recipient of several best paper awards and recognitions, including the Premiers Research Excellence Award, the IEEE ComSoc Hal Sobol Award, the Fred W. Ellersick Award, the Joe LociCero Award, the Dan Stokesbury Award, the Salah Aidarous Award, and the IEEE Canada McNaughton Gold Medal. He was the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT from 2007 to 2010 and on the editorial boards of other journals. He is a fellow of the Engineering Institute of Canada and the Canadian Academy of Engineering.

Jeebak Mitra received the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of British Columbia in 2005 and 2010, respectively. From 2010 to 2011, he was a Senior System Engineer with Riot Micro, leading the system level design for a local thermal equilibrium baseband. From 2011 to 2012, he was a Team Leader for physical layer DSP design with BLINQ Networks, Ottawa, focusing on small cell backhaul products. Since 2013, he has been a Senior Staff Engineer with the Huawei Technologies Canada Research Center, Ottawa, in the areas of algorithm design and implementation for coherent high-speed optical transceivers and flexible optical networks. His research interests lie in the area of high-performance communication systems design focusing on optical and wireless networks. He was a recipient of the Best Student Paper Award at the IEEE Canadian Conference in Electrical and Computer Engineering 2009. He was a co-recipient of the Best Paper Award at CNSM 2017.

Liu Liu received the M.Sc. and Ph.D. degrees in communication and information systems from the University of Electronic Science and Technology of China in 2011 and 2015, respectively. He was a Visiting Scholar in computer science and engineering with the State University of New York at Buffalo from 2012 to 2014. He joined Huawei as a Research Engineer in 2015. His research interests focus on network planning and optimization, uncertainty optimization, approximation algorithms, and cloud computing.