This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

Virtual Network Embedding with Guaranteed Connectivity under Multiple Substrate Link Failures

Nashid Shahriar, *Student Member, IEEE*, Reaz Ahmed, Shihabur Rahman Chowdhury, *Student Member, IEEE*, Md Mashrur Alam Khan, Raouf Boutaba, *Fellow, IEEE*, Jeebak Mitra, and Feng Zeng

***Abstract*—This paper addresses Co*nnectivity-aware* Vi*rtual* N*etwork* E*mbedding* (*CoViNE*) problem, which consists in embedding a virtual network (VN) on a substrate network while ensuring VN connectivity (without any bandwidth guarantee) against multiple substrate link failures. *CoViNE* provides a weaker form of survivability incurring less resource overhead than traditional VN survivability models. To optimally solve *CoViNE*, we present an Integer Linear Program (ILP), namely *CoViNE-opt*. *CoViNE-opt* enumerates an exponential number of edge-cuts in a VN severely limiting its scalability. Therefore, we decompose CoViNE into three sub-problems: i) augmenting a VN with virtual links to provide necessary connectivity, ii) identifying the virtual links that should be embedded disjointly, and iii) computing a VN embedding while satisfying the disjointness constraints. We introduce *conflicting set* abstraction that allows to address sub-problems (i) and (ii) without enumerating all the edge-cuts of a VN. We propose two novel solutions to *CoViNE* leveraging conflicting set, namely *CoViNE-ILP* and *CoViNE-fast*. *CoViNE-ILP* uses a heuristic algorithm to address sub-problems (i) and (ii), while an ILP is used for sub-problem (iii). In contrast, *CoViNE-fast* uses heuristics for solving all three sub-problems. Through simulation, we evaluate the optimality and scalability of our solutions and demonstrate a failure restoration use-case enabled by *CoViNE*.**

I. Introduction

Perceived as a key enabling technology for 5G mobile networks [1], [2], Network Virtualization (NV) (*aka* Network Slicing) enables an Infrastructure Provider (InP) to better utilize Substrate Network (SN) resources by embedding Virtual Networks (VNs) in support of services with differing requirements from multiple Service Providers (SPs). NV allows the co-existence of heterogeneous services from different SPs on the same infrastructure and opens new revenue streams for the InP [3]. However, benefits from NV come at additional resource management challenges for an InP. A well studied resource management challenge in NV is to efficiently map the virtual nodes and virtual links from an SP's VN request onto substrate nodes and paths, respectively, also known as the Virtual Network Embedding (VNE) problem [4].

NV operates in a dynamic environment where substrate resources may fail and multiple concurrent failures is not a rare event [5]. Surviving failures is of paramount importance, since a single failure in an SN may result in multiple failures in the embedded VNs. Finding a VN embedding that can survive failures in an SN is known as the Survivable Virtual Network Embedding (SVNE) problem [6]. The majority of the works on SVNE focus on link failures, as they occur more frequently than node failures [7]. SVNE approaches, in general, allocate redundant resources for each (or selected) virtual link(s) and node(s), either pro-actively while computing the embedding or reactively after a failure occurs [8]. Traditionally, proactive SVNE approaches focus on guaranteeing virtual link demand in the presence of failure(s). These approaches assume that InPs handle substrate failures by provisioning redundant backup resources to guarantee virtual links' bandwidth in the event of failures, which in turn incurs additional cost to SPs. Backup resource requirement can increase substantially in a multiple (*e.g.*, k) link failure scenario because of the combinatorial number of k link failure possibilities.

In this paper, we focus on a different form of survivability than traditional proactive SVNE, namely **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (*CoViNE*). Our goal is to find a VN embedding that remains connected (without any bandwidth guarantee) in the presence of multiple substrate link failures. Guaranteeing connectivity in the VN embedding will incur less resource overhead and reduced cost of leasing resources for a VN, however, providing a weaker form of survivability. This survivability model is well-suited for VNs that carry best-effort traffic and can tolerate disruption during failure restoration. Upon failures, the affected VN traffic can be rerouted to alternate paths following any predefined policy, *e.g.*, customer priority. This survivability model also allows to delegate failure handling responsibility to an SP, which can then use a Software Defined Network (SDN) controller to employ their own network design and restoration techniques instead of simply relying on the InP [9]–[11].

Although our focus is NV, *CoViNE* is equally applicable to IP-over-Wavelength-division multiplexing (WDM) networks. The problem of ensuring IP network connectivity in the presence of underlying WDM link failure(s) is known as *link survivable mapping*. Two variations of the link survivability problem have been studied in IP-over-WDM literature [12]: i) *weakly link survivable mapping* (WLSM) ensures only IP-layer connectivity; ii) *strongly link survivable* mapping guarantees both connectivity and bandwidth of the failed IP

This work was supported in part by Huawei Technologies and in part by an NSERC Collaborative Research and Development Grant. This work also benefited from the use of CrySP RIPPLE facility at the University of Waterloo.

Nashid Shahriar (email: nshahria@uwaterloo.ca), Shihabur Rahman Chowdhury (email: sr2chowdhury@uwaterloo.ca), and Raouf Boutaba (email: rboutaba@uwaterloo.ca) are with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON CA N2L 3G1

Reaz Ahmed (email: reaz@google.com) is with Google, Kitchener ON CA N2H 5G5

Md Mashrur Alam Khan (email: mashrur.alamkhan@ceridian.com) is with Ceridian Canada, North York ON CA M2P 2B7

Jeebak Mitra (email: jeebak.mitra@huawei.com) is with Huawei Technologies Canada Research Center, Ottawa ON CA K2K 3J1

Feng Zeng (email: zengfeng137140@huawei.com) is with Huawei Technologies Co., Ltd., Chengdu, Sichuan Province, P.R. China

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

link(s) against WDM link failures. However, a major difference between VNE and mapping in IP-over-WDM networks is that IP routers are attached to fixed locations in the later, whereas the placement of virtual nodes is an outcome of VNE algorithm in the former. Therefore, solutions for IP-over-WDM networks such as [13]–[15] cannot be directly applied to *CoViNE*. As a matter of fact, WLSM problem is merely a special case of *CoViNE*.

Solving *CoViNE* under k substrate link failures requires satisfying the following necessary and sufficient condition [16]: at least one link in each edge-cut of a VN must remain connected after any k substrate link failures. Given that there can be an exponential number of edge-cuts in a VN and also the combinatorial number of k substrate link failure possibilities, the aforementioned condition becomes impractical to be satisfied even for small substrate networks [16]. Alternatively, the same connectivity guarantee of a VN under k substrate link failures can be achieved by the following two necessary conditions: i) the VN must be $k+1$ edge connected, and ii) at least $k+1$ edge-disjoint paths must exist between every pair of virtual nodes in the embedding of the VN on the SN. The first condition can be satisfied by augmenting the VN with additional virtual links if needed [14], [17]. However, the number of augmented virtual links should be minimized since these augmented links consume substrate resources during VNE. A naive way to satisfy the second condition is to embed all the virtual links of a $k+1$ edge connected VN on disjoint paths in the SN at the cost of increased resource requirements for embedding virtual links [18]. However, as we demonstrate in Section III-D, not all virtual links need to be embedded on disjoint substrate paths to satisfy the second condition. Therefore, a sought-after solution to *CoViNE* should simultaneously optimize the number of augmented virtual links and the disjointness constraints in order to minimize the cost of VN embedding.

Optimally solving *CoViNE* for k substrate link failures is intractable since it entails to jointly optimize VN augmentation, disjointness constraints computation, and embedding of the augmented VN while satisfying the disjointness and capacity constraints. As each of these problems, when solved independently, is either NP-complete [19] or NP-Hard [4], [18], addressing them simultaneously exacerbates the complexity of *CoViNE*. This challenging problem has not been well studied in the NV literature, although WLSM problem (a special case of *CoViNE*) has received significant attention in the IP-over-WDM literature. The majority of the solutions to WLSM have overlooked the critical step of augmentation, assuming that input VNs have the necessary edge-connectivity [16], [20]–[22], which is not always the case. In addition, a significant body of existing literature on WLSM focus on single substrate link failure [15], [20], [21]. Solutions for multiple substrate link failures either fall short in dealing with arbitrary VN topologies [17], or consider only a special case, *i.e.*, Shared Risk Link Group (SRLG) failures [16], [22]. The authors in [14] address a WLSM problem that considers both augmentation and multiple link failures. However, the heuristic solution in [14] requires a large number of virtual links to be embedded disjointly, possibly imposing an unsatisfiable number of disjointness constraints. To overcome these limitations, in this paper, we present novel solutions to *CoViNE* that can embed an arbitrary VN topology on an SN, while guaranteeing VN connectivity against k substrate link failures and minimizing substrate resource consumption. Our solutions, if needed, augment a VN with minimal number of virtual links and preserve the topological structure of the VN to remain transparent to the SP operating the VN. Specifically, we make the following contributions, which build on our earlier study presented in [23].

First, we present *CoViNE*, an alternate survivability model for VNE, which embeds a VN on an SN subject to the constraint that the VN remains connected under k substrate link failures, *i.e.*, at least one working path exists between every pair of virtual nodes when up to k substrate links fail. While doing so, *CoViNE* minimizes the bandwidth provisioning cost in the SN. The survivability model proposed by *CoViNE* significantly reduces backup resource requirement compared to traditional survivability approaches in the SVNE literature.

Second, we propose three novel solutions to *CoViNE*:

CoViNE-opt: An Integer Linear Program (ILP) formulation that jointly optimizes VN augmentation, disjointness constraints computation, and embedding of the augmented VN to optimally solve *CoViNE*. *CoViNE-opt* has an exponential number of variables and constraints that severely limits its scalability. To scale to larger problem instances, we decompose *CoViNE* into three sub-problems: (i) augmenting the VN with zero or more virtual links to make it $k+1$-edge connected; (ii) computing the set of virtual links to be embedded disjointly for ensuring connectivity against k substrate link failures; and (iii) embedding the VN while satisfying the aforementioned disjointness constraints. The following two approaches (*i.e.*, *CoViNE-ILP* and *CoViNE-fast*) sequentially solve all the sub-problems of *CoViNE* in a more scalable manner, however, without guaranteeing an optimal solution.

CoViNE-ILP: Employs a heuristic that solves sub-problems (i) and (ii) in polynomial time. This heuristic leverages *conflicting set* abstraction resulting from a theoretical analysis of *CoViNE*. The *conflicting set* abstraction allows to generate a polynomial number of variables and constraints to be used by an ILP for solving sub-problem (iii). The complexity of this ILP limits its applicability to substrate networks of few hundred nodes.

CoViNE-fast: Uses heuristics for all three sub-problems of *CoViNE* to scale to larger problem instances.

Finally, we perform extensive simulations to evaluate the optimality and scalability of the proposed solutions under single and double substrate link failures. We restrict our simulations to one and two link failure cases since the probability of more than two simultaneous link failures is extremely low [7], [24]. We also compare our solutions with a VNE approach that does not guarantee VN connectivity upon failures [4]. Although our sequential solutions (*i.e.*, *CoViNE-ILP* and *CoViNE-fast*) do not guarantee optimality, simulation results show that they perform close to *CoViNE-opt*, and scale to larger problem instances. Finally, we demonstrate how *CoViNE* can enable a VN operator to recover from link failures without depending on the SN provider.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

The rest of this paper is organized as follows. We discuss the related literature in Section II. In Section III, we present the system model, *CoViNE* problem statement, definitions, and assumptions. An ILP formulation for optimally solving *CoViNE* is presented in Section IV. The theoretical analysis of *CoViNE* is laid in Section V. Then, we present a heuristic algorithm for VN augmentation and computing disjointness constraints in Section VI-A, followed by an ILP formulation and a heuristic algorithm for *CoViNE* embedding in Section VI-B and in Section VI-C, respectively. Evaluation results for *CoViNE* are presented in Section VII. Finally, we conclude in Section VIII with some future research directions.

II. Related Works

Survivability in NV and IP-over-WDM networks has been well studied over the past years [6], [15], [17], [20], [25]–[28]. We discuss the most prominent approaches addressing survivability from both NV (§ II-A) and IP-over-WDM literature (§ II-B), and contrast them with our solutions for *CoViNE*.

A. Survivable Virtual Network Embedding (SVNE)

Rahman *et al.*, first formulated the SVNE for single substrate link failure as a mixed integer liner program [6]. Subsequent research works have addressed different aspects of SVNE such as substrate node failures [29]–[32] and link failures [25], [27], [33], [34]. SVNE approaches for node failures can be broadly classified into two groups. The first group of works proposed to pro-actively provision dedicated or shared resources in the SN to survive single or multiple node failures [29], [30], [32]. The second group proposed to reactively compute VN embedding after one or more nodes have failed in the SN [31], [35], [36]. Another stream of SVNE research has focused on ensuring VN survivability during one or more substrate link failures, both pro-actively during VN embedding [25], [27], [28], [34] and reactively after a failure [33], [37]. In contrast, we address multiple substrate link failures and propose a different form of survivability instead of guaranteeing full bandwidth of the virtual links during failures as considered in the SVNE approaches.

Recently, there has been a growing interest in designing connectivity-guaranteed VNE schemes [16], [38]. For instance, the proposal from Zhu *et al.* [38] ensures that the unaffected part of a VN remains connected under a single substrate node failure to continue the services provided by the VN. They formulate the problem as an ILP and develop an ant colony optimization algorithm to obtain a local optimal solution. Hmaity *et al.* [39] distinguish between network connectivity and content connectivity and focus on providing content connectivity against double link failures. Their approach guarantees connectivity in a VN after single substrate link failure and maintains content connectivity in the presence of double substrate link failures. They argue that guaranteeing content connectivity may require lower amount of resources compared to network connectivity, and can be the sought after choice for Content Delivery Networks. Zhou *et al.* [16] propose to use cross-layer spanning trees to design survivable cloud networks against SRLG failure. Each spanning tree protects one SRLG by mapping the virtual links in the spanning tree in the paths that avoid the substrate links in the SRLG. They also discuss a way to extend their SRLG based solution to k substrate link failures. However, this solution needs a combinatorial number of spanning trees to protect in order to survive arbitrary k link failures. In contrast, our solution computes only one spanning tree of the VN even to survive any k substrate link failures.

B. Survivability in IP-over-WDM Network

Modiano *et al.* [20] presented an ILP formulation for survivable link routing of an IP network on WDM SN in the presence of a WDM link failure. Their formulation explores exponential number of edge-cuts in the IP network and ensures that the IP links belonging to a edge-cut are routed on at least two disjoint WDM paths. This approach does not scale well as the number of edge-cuts grows exponentially with the size of the IP network. Todimala *et al.* [22] have proposed an improved ILP formulation by identifying polynomial number of primary cuts in planar and hierarchical planar cyclic graphs representing IP networks. Their formulation computes the survivable routing of these sub-classes of IP networks against single node or SRLG failure(s). However, this formulation is not applicable to non-planar IP topologies and arbitrary failure scenarios. Lee *et al.* [40] extended the Max-flow min-cut theorem to multi-layer networks and proposed approximation algorithms for survivable IP network mapping on an WDM SN, while maximizing the minimum cross layer cut. Zhou *et al.* [41] identified four cross-layer metrics and presented mixed-integer linear programs (MILPs) to determine a mapping that maximizes one of the defined metrics. They also provided an MILP formulation for augmenting the IP topology. A major drawback of these MILP-based approaches is that they do not scale well with VN or SN size, because of the inherent complexity of the LP-solvers.

Several heuristic based approaches have been proposed for survivable IP link mapping in large WDM networks. Kurant *et al.* [13], [17] proposed SMART, a framework for finding survivable mapping of an IP network by repeatedly picking sub-graph (e.g., cycles) of the network and mapping the IP links in the sub-graph on disjoint paths in the WDM SN. SMART can ensure connectivity under double WDM link failures for IP networks having a few special structures, thus limiting its applicability. An extension to SMART has been proposed in [21] that exploits the duality between circuits and cuts in the graph representing an IP topology. In [42], Thulasiraman *et al.* remove some of the shortcomings of the dual framework of SMART by using generalized circuit and cutset cover sequences. In another extension of SMART, Javed *et al.* [43] used the concept of randomized rounding discussed in [44] to find disjoint WDM paths for the IP links that achieved higher success rate than SMART. Zhou *et al.* [15] proposed an algorithm that identifies a set of spanning trees of an IP network and computes a shortest-path based routing of the IP links such that at least one of the spanning trees remains unaffected after a WDM link failure. Another school of thought is to compute restoration path for each IP link in a way that the IP link and its restoration path are not affetced

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

by the same substrate link failures [5], [12]. However, these approaches require to know the set of substrate links that will potentially fail together. In contrast, our solution is generic, *i.e.*, does not assume any specific property of the IP network or the WDM SN, and can ensure connectivity in the presence of multiple WDM link failures.

C. Virtual Link Augmentation

IP link augmentation strategies such as [15], [45], [46] to survive WDM link failures primarily focus on single link failure scenarios and cannot be generalized to multiple failures. Zhou *et al.*, propose to augment logical VLinks between arbitrary pairs of VNodes [21]. In contrast, we propose to perform augmentation only between pairs of adjacent VNodes not to alter the VN topology. Thulasiraman *et al.*, propose an augmentation strategy for ensuring survivability under k WDM link failures [14]. They propose to augment IP links until a complete subgraph of $k+2$ nodes in the IP network is constructed and the remaining nodes are $k+1$ edge connected to the subgraph. Their solution maps all the IP links in the complete subgraph of $k+2$ nodes onto mutually disjoint WDM paths. To survive k link failures, this approach requires higher number of IP links to be augmented and more disjointness constraints to be satisfied than those of our approach.

D. Complexity of the VNE problem

Optimally solving the general case of the VNE problem without any survivability requirements, is at least as hard as the NP-Hard *Multi-commodity Unsplittable Flow Problem (MCUFP)* [47] when the source and destination of the flows are unknown. The best known approximation bound for the MCUFP with known sources and destinations is $(2 + \epsilon)$ for very simple classes of graphs, namely, line and cycle graphs [48]. A Linear Programming relaxation based algorithm for unsplittable flows on trees has been proposed in [49], however, the approximation ratio is a logarithmic function of the number of nodes. In reality, SNs are more densely connected than line and cycle graphs, and trees. Finding a constant factor approximation algorithm for general graphs still remains an open problem [50]. Moreover, line graphs and trees are 1-edge connected, therefore, they are not suitable for deploying SNs to guarantee VN connectivity under $k \geq 1$ link failures. Finally, a more recent study has proved that the general case of the VNE problem with capacity constraints on the links is $\mathcal{NP}$-complete and cannot be approximated under any objective unless $\mathcal{P} = \mathcal{NP}$ [51].

III. Preliminaries

The subsequent sections build upon the background, definitions, and assumptions presented in this section. Table I lists all the major notations used in the rest of this paper.

A. System Model

We represent an SN as an undirected graph, $G = (V, E)$, where V and E denote the set of Substrate Nodes (SNodes) and Substrate Links (SLinks), respectively. The set of neighbors of an SNode $u \in V$ is denoted by $\mathcal{N}(u)$. Bandwidth capacity of an SLink $(u, v) \in E$ is b_{uv}, while the cost of allocating one unit of bandwidth in (u, v) is C_{uv}. Similarly, a VN is represented as an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ and $\bar{E}$ denote the set of Virtual Nodes (VNodes) and Virtual Links (VLinks), respectively. The set of neighbors of a VNode $\bar{v} \in \bar{V}$ is denoted by $\mathcal{N}(\bar{v})$. Each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ has bandwidth requirement $b_{\bar{u}\bar{v}}$. Each VNode $\bar{u} \in \bar{V}$ has a location constraint, $L(\bar{u}) \subseteq V$, that denotes the set of SNodes where $\bar{u}$ can be embedded. We represent the location constraint $L(\bar{u}) \subseteq V$ of $\bar{u} \in \bar{V}$ with the binary variable $\ell_{\bar{u}u}$ that is set to 1 if $\bar{u} \in \bar{V}$ can be mapped to $u \in V$, 0 otherwise.

B. CoViNE Problem Statement

Given an SN $G = (V, E)$, a VN $\bar{G} = (\bar{V}, \bar{E})$, and location constraints $L(\bar{u})$, $\forall \bar{u} \in \bar{V}$, CoViNE finds an embedding that

- provides a function $f : \bar{V} \rightarrow V$ to map every VNode $\bar{u} \in \bar{V}$ to exactly one SNode $u \in V$ while satisfying the location constraint and incurring no overlap, *i.e.*, $\forall \bar{u}, \bar{v} \in \bar{V} \wedge \bar{u} \neq \bar{v} \implies f(\bar{u}) \neq f(\bar{v})$ and $\forall \bar{u} \in \bar{V}\ \ f(\bar{u}) \in L(\bar{u})$,
- provides a function $g : \bar{E} \rightarrow 2^E$ to map each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ to a substrate path $Q^{f(\bar{u})f(\bar{v})}$ with sufficient bandwidth to satisfy the VLink demand $b_{\bar{u}\bar{v}}$,
- ensures the connectivity in $\bar{G}$ in the presence of up to k SLink failures in G,
- minimizes the total cost of embedding in terms of substrate bandwidth consumption as defined by the following

$$\sum_{\forall (\bar{u},\bar{v}) \in \bar{E}} \sum_{\forall (u,v) \in Q^{f(\bar{u})f(\bar{v})}} C_{uv} \times b_{\bar{u}\bar{v}} \tag{1}$$

C. Definitions and Design Choices

Definition 1. ***Edge-cut:*** *An edge-cut $\bar{C}_i \subset \bar{E}$ of a VN $\bar{G}$ is the set of VLinks that connect the VNodes in an non-empty set $\bar{S} \subset \bar{V}$ to the VNodes in $\bar{V} \setminus \bar{S}$ and the removal of the VLinks in $\bar{C}_i$ partitions the VN. Let $\bar{C}^{\bar{G}}$ be the set of all edge-cuts in the VN $\bar{G}$, $\bar{C}^{\bar{G}} = \{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_m\}$. Since the number of non-empty proper subsets (i.e., $\bar{S} \neq \phi \wedge \bar{S} \neq \bar{V}$) of $\bar{V}$ is $2^{\bar{V}} - 2$, we have $m = 2^{\bar{V}} - 2$. Let $|\bar{C}_i|$ denote the number of VLinks in the edge-cut $\bar{C}_i$.*

A VN embedding remains connected during k SLink failures if the following two necessary conditions are met: i) the VN is $k+1$ edge-connected following the definition of $k+1$ edge-connected graphs, implying that the size of each edge-cut is at least $k+1$, $\forall C_i \in \bar{C}^{\bar{G}}, |\bar{C}_i| >= k+1$, ii) the VLinks in each edge-cut C_i are embedded on at least $k+1$ edge-disjoint paths in the SN. This can be trivially proved since the failure of k SLinks can impact at most k edge-disjoint paths in the SN, leading to at most k VLink failures from an edge-cut in a VN leaving at least one VLink to ensure connectivity. However, if the given VN $\bar{G}$ lacks $k+1$ connectivity, we need to augment $\bar{G}$ with additional VLinks. This augmentation can be done in two ways: i) augment VLinks between arbitrary pair of VNodes, which is a well studied problem [45], [46]; ii) augment parallel VLinks between already adjacent VNodes in $\bar{G}$ [12], [15], [21]. Arbitrary augmentation can ensure $k+1$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

edge connectivity by introducing minimal number of VLinks, however, this approach will change the input VN topology. Although parallel VLink augmentation may not yield minimal resource usage, it does not alter the input VN topology. From a VN operator perspective, it is very important to preserve its VN topology. Hence, we opt for the second alternative, *i.e.*, augmenting a VN with only parallel VLinks.

Definition 2. ***k-protected VN:*** *A k-protected VN, $\hat{G} = (\hat{V}, \hat{E})$, is a VN that becomes $k+1$ edge connected after augmenting the minimum number of parallel VLinks to a VN, $\bar{G} = (\bar{V}, \bar{E})$. The k-th parallel VLink between $\bar{u}$ and $\bar{v}$ is denoted by $(\bar{u}, \bar{v})^k$, where $(\bar{u}, \bar{v})^0$ or simply $(\bar{u}, \bar{v})$ represents the input VLink between $\bar{u}$ and $\bar{v}$. Here, $\hat{V} = \bar{V}$ and $\hat{E} = \bar{E} \cup \tilde{E}$, where $\tilde{E} = \{(\bar{u}, \bar{v})^k | (\bar{u}, \bar{v}) \in \bar{E}, k \in K \text{ and } k \geq 1\}$ is the set of augmented parallel VLinks.*

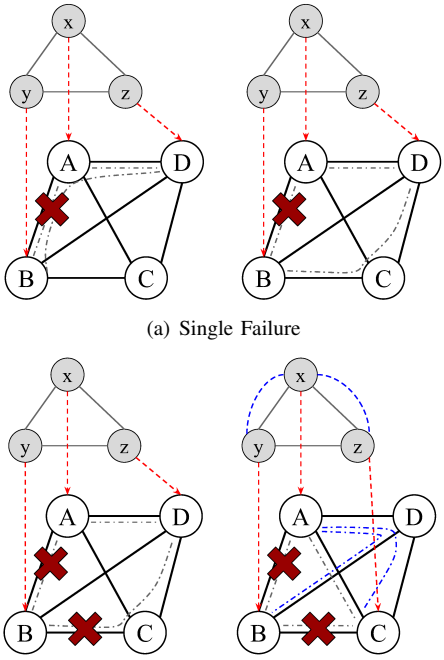
Definition 3. ***k-protected component:*** *A k-protected component of a graph $\bar{G}$ is a multi-graph $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$, where $\hat{V}_k \subseteq \bar{V}$, $\hat{E}_k = \bar{E}_k \cup \tilde{E}_k$, $\bar{E}_k \subseteq \bar{E}$, $\tilde{E}_k \subseteq \tilde{E}$ and $\tilde{E}_k$ is a set of parallel VLinks augmented in such a way that simultaneous removal of k arbitrary VLinks from $\hat{G}_k$ will not partition $\hat{G}_k$.*

Determining the bandwidth of the parallel VLinks as well as the amount of spare bandwidth to be reserved for the input VLinks (in $\bar{E}$) to survive failures is a separate problem of its own, and has been studied in [12], [52]–[54]. Here, we assume that the bandwidth of a parallel VLink will be the same as the bandwidth of the input VLink parallel to it in order to salvage full bandwidth of the VLink upon failures. In the case that a parallel VLink has lower bandwidth than the input VLink, the amount of restored bandwidth will be decreased.

D. CoViNE Example

We illustrate CoViNE examples for single and double failure scenarios in Fig. 1(a) and Fig. 1(b), respectively. In these examples, xyz is the VN and $ABCD$ is the SN. The arrow from a VNode to an SNode denotes node mapping and the dotted lines between SNodes denote link mapping. To survive single SLink failure ($k = 1$), the VN must be 2 edge-connected. Since xyz is already 2 edge-connected, no augmentation is required. Fig. 1(a) shows an un-survivable embedding (on the left) and a survivable embedding (on the right) of xyz. They differ in satisfying disjointness constraints. The embedding on the left satisfies no disjointness constraint, hence the VLinks of an edge-cut $\{(x, y), (y, z)\}$ share an SLink (A, B). Upon the failure of (A, B), both VLinks fail, and VNode y is disconnected from the rest of the VN. The embedding on the right adheres to the disjointness constraints, hence no SLinks are shared. Even though SLink (A, B) and correspondingly VLink (x, y) fail, the VN remains connected.

To survive double link failures (Fig. 1(b)), *i.e.*, for $(k = 2)$, the VN should be 3 edge-connected, which is not the case for VN xyz. Due to such lack of edge-connectivity, even an edge-disjoint embedding of all the VLinks (the embedding on the left) cannot survive two SLink failures. Hence, we transform the VN into a 2-protected one by augmenting VLinks (dashed VLinks in the figure) and embedding the resulting VN adhering to the disjointedness constraints as shown on the right. Note that, for a survivable embedding of the 2-protected VN, some SLinks can be shared (*e.g.*, (A, D)) among the mappings of some VLinks since not all the VLinks need to be embedded on mutually disjoint paths.



(a) Single Failure

(b) Double Failure

Fig. 1. CoViNE examples

IV. Optimal Solution to CoViNE

In this section, we present *CoViNE-opt*, an ILP formulation for optimally solving *CoViNE*. *CoViNE-opt* transforms an input VN $\bar{G}$ to a k-protected VN $\hat{G}$ by augmenting parallel VLinks, and minimizes the total cost of provisioning bandwidth for the VLinks of $\hat{G}$ on an SN G, while ensuring $\hat{G}$'s connectivity in the presence of k SLink failures.

A. Decision Variables

Recall from Section III-C that $\bar{G}$ needs to be augmented with additional VLinks if $\bar{G}$ is not already $k+1$ edge-connected. This implies that we must add parallel VLinks to the edge-cuts in $\bar{G}$ whose sizes are less than $k+1$, thus ensuring that there are no edge-cuts in $\bar{G}$ with size less than $k+1$. Determining which parallel VLinks should be added to $\bar{G}$ is non-trivial as augmenting one VLink parallel to $(\bar{u}, \bar{v})$ may increase the sizes of all edge-cuts that contain $(\bar{u}, \bar{v})$, resulting in a combinatorial decision making problem. To optimally decide which parallel VLinks are augmented, we introduce the following decision variable:

$$a_k^{\bar{u}\bar{v}} = \begin{cases} 1 & \text{if } k\text{-th VLink between } \bar{u} \text{ and } \bar{v} \text{ s.t. } (\bar{u}, \bar{v}) \in \bar{E} \\ & \text{is augmented,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, $a_0^{\bar{u}\bar{v}}$ denotes the input VLink between $\bar{u}$ and $\bar{v}$. Therefore, $\forall (\bar{u}, \bar{v}) \in \bar{E}, a_0^{\bar{u}\bar{v}} = 1$. Once a VLink $(\bar{u}, \bar{v})^k$ s.t. $(\bar{u}, \bar{v}) \in \bar{E} \wedge k \in K \wedge k \neq 0$ is augmented

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

by *CoViNE-opt*, $(\bar{u},\bar{v})^k$ is included in all the edge-cuts that contain $(\bar{u},\bar{v})$. We denote the set of VLinks in $\bar{C}_i$ and all the parallel VLinks augmented to the VLinks in $\bar{C}_i$ as $\bar{C}_i \cup \bar{C}_i^k$. Mathematically, $\bar{C}_i \cup \bar{C}_i^k = \{\{(\bar{u},\bar{v})^0 \text{ s.t. } (\bar{u},\bar{v}) \in \bar{C}_i\} \cup \{(\bar{u},\bar{v})^k \text{ s.t. } (\bar{u},\bar{v}) \in \bar{C}_i \wedge k \in K \wedge k \neq 0 \wedge a_k^{\bar{u}\bar{v}} = 1\}\}$.

A VLink is mapped to a set of SLinks forming a path in the SN. In this ILP formulation, we represent a bidirectional SLink by two unidirectional SLinks in opposite directions. We represent the mapping between $(\bar{u},\bar{v})^k$, the k-th VLink between adjacent VNodes $\bar{u}$ and $\bar{v}$ and an SLink $(u,v) \in E$ using the following decision variable:

$$x_{uv}^{\bar{u}\bar{v}k} = \begin{cases} 1 & \text{if } (\bar{u},\bar{v})^k \text{ s.t. } (\bar{u},\bar{v}) \in \bar{E} \text{ and } k \in K \text{ is} \\ & \text{mapped to } (u,v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The following decision variable represents the mapping between VNodes and SNodes:

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

To guarantee that a VN remains connected (*i.e.*, at least one path exists between any pair of VNodes) during any combination of k SLink failures, one of the necessary conditions discussed in Section § III-C is that for any edge-cut $\bar{C}_i \in C^{\bar{G}}$, the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ are embedded on at least $k+1$ edge-disjoint paths in the SN. *CoViNE-opt* enforces this condition by partitioning the set of VLinks in each $\bar{C}_i \cup \bar{C}_i^k$ into two mutually exclusive groups. The first group, denoted as $D_{\bar{C}_i}$, will contain the VLinks that cannot share an SLink in their mappings to ensure the existence of at least $k+1$ edge-disjoint paths among the mappings of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$. Hence, the size of $D_{\bar{C}_i}$ must be at least $k+1$. The second group will contain the rest of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ that are not in $D_{\bar{C}_i}$. The following decision variable decides the assignment of a VLink in $\bar{C}_i \cup \bar{C}_i^k$ to $D_{\bar{C}_i}$:

$$d_{\bar{C}_i}^{\bar{u}\bar{v}k} = \begin{cases} 1 & \text{if } (\bar{u},\bar{v})^k \text{ s.t. } (\bar{u},\bar{v}) \in \bar{C}_i \text{ and } k \in K \\ & \text{belongs to } D_{\bar{C}_i}, \\ 0 & \text{otherwise.} \end{cases}$$

B. Constraints

1) Augmentation Constraints: Parallel VLinks are augmented to ensure that there is no edge-cut in the VN with less than $k+1$ VLinks. (2) ensures that the size of each edge-cut is at least $k+1$ after augmentation. Note that if *CoViNE-opt* decides to augment a parallel VLink, the augmented VLink must also be mapped to a path in the SN (see § IV-B2), increasing the cost of embedding. Since the objective of *CoViNE-opt* is a minimization function (see § IV-C), *CoViNE-opt* will augment the minimum number of parallel VLinks despite not having a strict equality in (2).

$$\forall \bar{C}_i \in \bar{C}^{\bar{G}} \text{ s.t. } |\bar{C}_i| < k+1 : \sum_{\forall(\bar{u},\bar{v})\in\bar{C}_i} \sum_{k\in K} a_k^{\bar{u}\bar{v}} >= k+1 \tag{2}$$

TABLE I
NOTATION TABLE

$G=(V,E)$	Substrate Network (SN)
$\bar{G}=(\bar{V},\bar{E})$	Virtual Network (VN)
$\hat{G}=(\hat{V},\hat{E})$	k-protected VN
$\hat{G}_k=(\hat{V}_k,\hat{E}_k)$	k-protected component of a VN $\bar{G}$
$\hat{G}_k \odot \hat{v}$	An expansion of $\hat{G}_k$ towards $\hat{v}$
C_i	An edge-cut $C_i \subset E$ s.t. $\lvert C_i\rvert > 0$ in G
$\bar{C}^G$	Set of edge-cuts $\bar{C}_i \subset \bar{E}$ s.t. $\lvert C_i\rvert > 0$ in $\bar{G}$
$\bar{C}_i \cup \bar{C}_i^k$	The set of VLinks in $\bar{C}_i \in \bar{C}^G$ and all the parallel VLinks augmented to the VLinks in $\bar{C}_i \in \bar{C}^{\bar{G}}$
$(\bar{u},\bar{v})^k$	k-th VLink between $\bar{u}$ and $\bar{v}$
$\chi^{\hat{u}\hat{v}k}$	Conflicting set of a VLink $(\hat{u},\hat{v})^k$
$\chi_{\odot}^{\hat{u}\hat{v}k}$	Conflicting set of a VLink $(\hat{u},\hat{v})^k$ during expansion
$\chi^{\hat{G}}$	Conflicting set of a VN $\hat{G}$
Q^{uv}	A path between SNodes u and v in G
$P^{\hat{u}\hat{v}}$	A path between VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$\mathscr{P}^{\hat{u}\hat{v}}$	Set of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$P_i^{\hat{u}\hat{v}}$	i-th edge-disjoint path from $\hat{u}$ to $\hat{v}$ in $\hat{G}$
$\mathtt{p}_i^{\hat{u}\hat{v}}$	i-th edge-disjoint shortest path from $\hat{u}$ to $\hat{v}$ in $\hat{G}$
$\mathcal{P}^{\hat{G}_k\hat{v}}$	Set of edge-disjoint shortest paths from $\hat{v}$ to $\hat{G}_k$
K	Set of integers from 0 to k, *i.e.*, $\mathcal{Z} \cap [0,k]$

2) VLink Mapping Constraints: VLinks are mapped to substrate paths following a *Multi-commodity Unsplittable Flow* formulation [55]. Bandwidth conservation is ensured by (3) which enforces that an SLink is not assigned VLink demands that exceed the SLink's bandwidth capacity. Then, (4) ensures flow conservation by making sure that for each input $(a_0^{\bar{u}\bar{v}}, \forall(\bar{u},\bar{v}) \in \bar{E})$ and augmented $(a_k^{\bar{u}\bar{v}} = 1 \text{ s.t. } (\bar{u},\bar{v}) \in \bar{E} \wedge k \in K \wedge k \neq 0)$ VLink, the in-flow and out-flow of each SNode is equal except at the SNodes where the endpoints of a VLink are mapped. Since the decision of VLink augmentation is not known in advance, the right hand side of (4) is multiplied by $a_k^{\bar{u}\bar{v}}$, yielding a quadratic constraint.

$$\forall(u,v) \in E : \sum_{\forall(\bar{u},\bar{v})\in\bar{E}} \sum_{\forall k\in K} x_{uv}^{\bar{u}\bar{v}k} \times b_{\bar{u}\bar{v}} \leq b_{uv} \tag{3}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : \sum_{\forall v\in\mathcal{N}(u)} (x_{uv}^{\bar{u}\bar{v}k} - x_{vu}^{\bar{u}\bar{v}k}) = a_k^{\bar{u}\bar{v}} \times (y_{\bar{u}u} - y_{\bar{v}u}) \tag{4}$$

We take the following steps to linearize (4). First, we introduce two binary variables $w_u^{\bar{u}\bar{v}k}$ and $z_u^{\bar{u}\bar{v}k}$ such that:

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \leq a_k^{\bar{u}\bar{v}} \tag{5}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \leq y_{\bar{u}u} \tag{6}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \geq a_k^{\bar{u}\bar{v}} + y_{\bar{u}u} - 1 \tag{7}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \leq a_k^{\bar{u}\bar{v}} \tag{8}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \leq y_{\bar{v}u} \tag{9}$$

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \geq a_k^{\bar{u}\bar{v}} + y_{\bar{v}u} - 1 \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

Then, we rewrite (4) in a linear form using the newly introduced variables as follows:

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\bar{u}\bar{v}k} - x_{vu}^{\bar{u}\bar{v}k}) = w_u^{\bar{u}\bar{v}k} - z_u^{\bar{u}\bar{v}k} \tag{11}$$

3) Disjointedness Constraints: (12) ensures that for each edge-cut $\bar{C}_i$, the disjoint group $D_{\bar{C}_i}$ contains at least $k+1$ VLinks from $\bar{C}_i \cup \bar{C}_i^k$. To prevent the inclusion of a non-augmented VLink into $D_{\bar{C}_i}$, we multiply $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ by $a_k^{\bar{u}\bar{v}}$ in (12). An SLink (u,v) can be used at most once in the mappings of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ that are assigned to $D_{\bar{C}_i}$ to ensure their edge-disjoint embedding. We enforce this by (13). Since both (12) and (13) are quadratic, we linearize them using the same technique discussed for (4).

$$\forall \bar{C}_i \in \bar{C}^{\bar{G}} : (\sum_{\forall(\bar{u},\bar{v})\in\bar{C}_i} \sum_{k\in K} d_{\bar{C}_i}^{\bar{u}\bar{v}k} \times a_k^{\bar{u}\bar{v}}) \geq k+1 \tag{12}$$

$$\forall(u,v) \in E, \forall \bar{C}_i \in \bar{C}^{\bar{G}} : \sum_{\forall(\bar{u},\bar{v})\in\bar{C}_i} \sum_{\forall k\in K} d_{\bar{C}_i}^{\bar{u}\bar{v}k} \times (x_{uv}^{\bar{u}\bar{v}k} + x_{vu}^{\bar{u}\bar{v}k}) \leq 1 \tag{13}$$

4) VNode Mapping Constraints: (14) ensures that VNode mapping follows the given location constraint. (15) ensures that a VNode is mapped to exactly one SNode. Finally, (16) ensures that an SNode does not host more than one VNode from a VN. However, an SNode can host multiple VNodes from different VNs.

$$\forall \bar{u} \in \bar{V}, \forall u \in V : y_{\bar{u}u} \leq \ell_{\bar{u}u} \tag{14}$$

$$\forall \bar{u} \in \bar{V} : \sum_{u\in V} y_{\bar{u}u} = 1 \tag{15}$$

$$\forall u \in V : \sum_{\bar{u}\in\bar{V}} y_{\bar{u}u} \leq 1 \tag{16}$$

C. Objective Function

The objective of *CoViNE-opt* is to minimize the bandwidth provisioning cost over all the SLinks for embedding all the original and augmented VLinks of a VN, $\bar{G}$, subject to the augmentation constraints (§ IV-B1), VLink mapping constraints (§ IV-B2), disjointness constraints (§ IV-B3), and VNode mapping constraints (§ IV-B4). Given that C_{uv} is the cost of allocating unit bandwidth on SLink $(u,v) \in E$, we have the following objective function for *CoViNE-opt*:

$$\text{minimize} \left(\sum_{\forall(\bar{u},\bar{v})\in\bar{E}} \sum_{\forall k\in K} \sum_{\forall(u,v)\in E} x_{uv}^{\bar{u}\bar{v}k} \times C_{uv} \times b_{\bar{u}\bar{v}} \right) \tag{17}$$

D. Complexity Analysis

Rost *et al.*, have shown that the VNE problem with capacity constraints on the links, *i.e.*, solving *CoViNE-opt* without the augmentation constraint (2) and disjointness constraints (12) - (13), is $\mathcal{NP}$-complete and cannot be approximated under any objective unless $\mathcal{P} = \mathcal{NP}$ [51]. Furthermore, when we include the augmentation and disjointedness constraints, *CoViNE-opt* becomes even more computationally intractable. The reason is that *CoViNE-opt* generates an exponential number of variables for $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ and an exponential number of constraints for (2), (12), and (13) since the number of edge-cuts in a VN is $O(2^{|\bar{V}|})$. For instance, total number of binary variables for $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ is $|\bar{E}| \times k \times (2^{|\bar{V}|} - 2)$ and total number of constraints for (13) is $|E| \times (2^{|\bar{V}|} - 2)$. In the worst case, all binary vectors have to be enumerated and all the constraints need to be explicitly checked for each of the enumeration, yielding the time complexity of $O((|E| \times (2^{|\bar{V}|} - 2)) \times 2^{(|\bar{E}| \times k \times (2^{|\bar{V}|} - 2))})$. To reduce the size of the problem, we decompose the joint optimization, and separate augmentation and disjointness constraints computation from embedding as presented in the subsequent sections. Note that by decomposing *CoViNE-opt* into sub-problems and solving them sequentially may yield sub-optimal solution.

V. Theoretical Analysis for k Link Survivability

In this section, we first devise an efficient mechanism to compute disjointness constraints of a VN assuming that the VN is k-protected (§ V-A). We then extend this mechanism to transform an arbitrary VN to a k-protected VN (§ V-B).

A. Disjointness Constraints Computation

As discussed in § IV-D, there are exponential number of edge-cuts in $\hat{G}$ and combinatorial number of ways to assign VLinks from each edge-cut to its $k+1$ disjoint groups. To reduce the exponential number of constraints, we first define the disjointedness relationship between the VLinks of $\hat{G}$ irrespective of the edge-cuts as follows.

Definition 4. ***Conflicting VLinks:*** *Two VLinks are considered conflicting if they must be embedded on edge-disjoint paths in the SN to ensure the VN remains connected (i.e., at least one path exists between any pair of VNodes) in the presence of k SLink failures.*

Definition 5. ***Conflicting set:*** *A conflicting set of a VLink $(\hat{u},\hat{v})^k$, denoted by $\chi^{\hat{u}\hat{v}k}$, is the set of VLinks in $\hat{E}$ those are conflicting with $(\hat{u},\hat{v})^k$. A conflicting set of a VN $\hat{G} = (\hat{V},\hat{E})$, denoted by $\chi^{\hat{G}}$, is defined as $\chi^{\hat{G}} = \{\chi^{\hat{u}\hat{v}k} | \forall(\hat{u},\hat{v})^k \in \hat{E}\}$.*

A conflicting set of $\hat{G}$ imposes disjointedness constraints on VLink embedding of $\hat{G}$. The larger the size of conflicting sets of the VLinks in $\hat{G}$, the higher the number of disjointedness constraints to be satisfied, and hence the longer becomes the substrate paths used for VLink embedding. This increased number of disjointedness constraints can have a twofold impact on embedding. First, it can increase the cost of embedding due to the longer substrate paths. Second, and more importantly, it can lead to infeasible solutions due to the lack of adequate edge-disjoint paths in a moderately dense SN. Therefore, we define the notion of *minimal conflicting set* of $\hat{G}$ that ensures $k+1$ edge connectivity of $\hat{G}$ after embedding, while minimizing the requirement of having disjoint paths in the embedding. This can be obtained by finding the minimum number of partitions of the VLinks of $\hat{E}$ such that the VLinks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

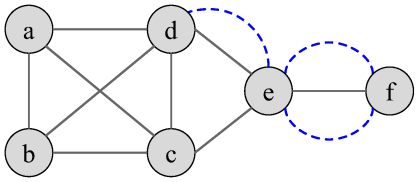


Fig. 2. The VN with only solid edges is the input VN, $\bar{G}$. The VN with both solid edges ($\bar{E}$) and dashed edges ($\tilde{E}$) is the 2-protected VN, $\hat{G}$. Any subgraph of $\hat{G}$ having 3 edge connectivity is $\hat{G}_2$.

in a partition are not conflicting with one another. Since VLinks in the same partition do not impose any disjointedness constraint, minimizing the number of partitions will yield a minimal conflicting set. However, partitioning the VLinks to yield a minimal conflicting set is non-trivial as per the following theorem.

Theorem 1. *Computing a minimal conflicting set of a VN is NP-complete.*

Proof. This problem is clearly in NP because we can verify that a given conflicting set of $\hat{G}$ ensures $k+1$ edge connectivity in polynomial time by successively removing a VLink $\hat{e}_i \in \hat{E}$ and all the VLinks in $\chi^{\hat{e}_i}$, and then checking for VN connectivity. To show that the problem is NP-complete, we reduce the NP-complete *Minimum vertex coloring* problem [56]. to computing a minimal conflicting set. Consider a graph $H = (V_H, E_H)$ as an instance of the *Minimum vertex coloring* problem. Also consider a bijection $\xi : \hat{E} \rightarrow V_H$ that maps each VLink in $\hat{E}$ to a vertex in V_H. There is an edge $(\xi(\hat{e}_i), \xi(\hat{e}_j)) \in E_H$ between the vertices $\xi(\hat{e}_i) \in V_H$ and $\xi(\hat{e}_j) \in V_H$ *if and only if* two VLinks $\hat{e}_i, \hat{e}_j \in \hat{E}$ are conflicting with each other. We can test if two VLinks are conflicting in polynomial time by removing them and checking for $k+1$ edge connectivity in $\hat{G}$. Hence, the conflicting set of $\hat{e}_i$ can be computed from H as $\chi^{\hat{e}_i} = \{\hat{e}_j | (\xi(\hat{e}_i), \xi(\hat{e}_j)) \in E_H\}$, while the conflicting set of $\hat{G}$ can be constructed as $\chi^{\hat{G}} = \{\chi^{\hat{e}_i} | \forall \hat{e}_i \in \hat{E}\}$. If two VLinks $\hat{e}_i, \hat{e}_j \in \hat{E}$ are not conflicting with each other, there is no edge between $\xi(\hat{e}_i)$ and $\xi(\hat{e}_j)$. Hence, $\xi(\hat{e}_i)$ and $\xi(\hat{e}_j)$ can be given the same color in H. Thus, finding a partition of $\hat{E}$ consisting of non-conflicting VLinks is equivalent to finding vertices with the same color in V_H. Therefore, a minimal conflicting set of $\hat{G}$ yields the minimum vertex coloring of H. Hence, *Minimum vertex coloring* $\leq_P$ computing *minimal conflicting set*. □

Note that computing the optimal conflicting set of a VN is harder than the NP-complete problem of computing a minimal conflicting set, since the former takes SN and embedding cost into account in addition to the number of disjointness constraints. Hence, we propose a heuristic algorithm in § VI-A1 to compute a conflicting set that tries to minimize number of disjointedness constraints as well as embedding cost. The following Theorem, known as Menger's Theorem [57], provide the basis for our algorithm to compute the conflicting set of a VN $\hat{G}$ within a reasonable time.

Theorem 2. *([57]) The size of the minimum edge-cut for two distinct VNodes $\hat{u}, \hat{v} \in \hat{G}$ is equal to the maximum number of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$.*

According to Theorem 2 any pair of VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$ will remain connected in the presence of k SLink failures, if at least one of the edge-disjoint paths $P_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ remains intact. This can be achieved by mapping any $k+1$ paths in $\mathscr{P}^{\hat{u}\hat{v}}$ into $k+1$ edge-disjoint paths in the SN. There are a combinatorial number of ways of choosing these $k+1$ edge-disjoint paths between $\hat{u}$ and $\hat{v}$. If $\mathscr{P}_1^{\hat{u}\hat{v}} = P_1^{\hat{u}\hat{v}}, P_2^{\hat{u}\hat{v}}, ..., P_{k+1}^{\hat{u}\hat{v}}$ is one possible combination chosen to have edge-disjoint mapping, two VLinks $(\hat{x}, \hat{y})^q \in P_i^{\hat{u}\hat{v}}$ and $(\hat{w}, \hat{z})^r \in P_j^{\hat{u}\hat{v}}$, such that $x \neq w$ and $y \neq z$, cannot share an SLink in their mappings. Therefore, a VLink $(\hat{x}, \hat{y})^q \in P_i^{\hat{u}\hat{v}}$ is conflicting with all other VLinks present in the paths in $\mathscr{P}_1^{\hat{u}\hat{v}} \setminus P_i^{\hat{u}\hat{v}}$, leading to $|\chi^{\hat{x}\hat{y}q}| = \sum_{P_i^{\hat{u}\hat{v}} \in \mathscr{P}_1^{\hat{u}\hat{v}} \wedge (\hat{x},\hat{y})^q \notin P_i^{\hat{u}\hat{v}}} |P_i^{\hat{u}\hat{v}}|$. For example, in Fig. 2, VNodes a and b will remain connected in presence of two SLink failures if the VLinks on paths $P_1^{ab} = (a,b)$, $P_2^{ab} = \{(a,d),(d,c),(c,b)\}$, and $P_3^{ab} = \{(a,c),(c,e),(e,d),(d,b)\}$ are mapped to disjoint SN paths. Hence, $\chi^{ab} = P_2^{ab} \cup P_3^{ab}$.

We now discuss some heuristics to reduce the above computation. First, we can ensure connectivity in $\hat{G}$ by ensuring connectivity in a minimum spanning tree (MST) $\hat{T}$ of $\hat{G}$. In this case, we need to compute $k+1$ edge-disjoint paths only for the $|\hat{V}|-1$ VLinks in $\hat{T}$, as opposed to considering all the VLinks in $\hat{G}$. For instance, in Fig. 2, $k+1$ edge-disjoint path computations are required for the VLinks in $\hat{T} = \{(a,b),(a,c),(c,d),(d,e),(e,f)\}$ instead of all the 12 VLinks in $\hat{G}$. Second, instead of arbitrarily selecting $k+1$ edge-disjoint paths from $\mathscr{P}^{\hat{u}\hat{v}}$, we can choose the first $k+1$ edge-disjoint shortest paths between $\hat{u}$ and $\hat{v}$. Thus, the size of the conflicting set of a VLink $(\hat{u}, \hat{v})^q \in \hat{T}$ becomes $|\chi^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$, where $\mathtt{p}_i^{\hat{u}\hat{v}}$ is the i-th edge-disjoint shortest path between two adjacent VNodes $\hat{u}$ and $\hat{v}$. This method yields smaller conflicting sets than selecting arbitrary edge-disjoint paths. For instance, the conflicting set of VLink (a,b) in Fig. 2 is $\chi^{ab} = \mathtt{p}_2^{ab} \cup \mathtt{p}_3^{ab}$ where $\mathtt{p}_2^{ab} = \{(a,c),(c,b)\}$, and $\mathtt{p}_3^{ab} = \{(a,d),(d,b)\}$. The following definitions, lemmas, and theorem formalize our heuristics and prove that they result in better conflicting sets than individual computation.

Definition 6. ***Expansion Operator*** $\odot$*: Given a k-protected component $\hat{G}_k$ of a VN $\hat{G}$ and a VNode $\hat{v}$, such that $\hat{v} \in \hat{V} \setminus \hat{V}_k$ and $\exists \hat{u} \in \hat{V}_k, \hat{v} \in \mathcal{N}(\hat{u})$, we define $\hat{G}_k \odot \hat{v}$ as an expansion of $\hat{G}_k$ generated by adding $\hat{v}$ and all the incident VLinks on $\hat{v}$ from any VNode in $\hat{G}_k$. Mathematically, $\hat{G}_k \odot \hat{v} = (\hat{V}_k \cup \{\hat{v}\}, \hat{E}_k \cup \{(\hat{u},\hat{v})^q | \hat{u} \in \hat{V}_k, \hat{u} \in \mathcal{N}(\hat{v})\})$*

Definition 7. ***EDSP*** $\mathcal{P}^{\hat{G}_k\hat{v}}$***:*** *We define EDSP as a set of Edge-Disjoint Shortest Paths $\mathcal{P}^{\hat{G}_k\hat{v}} = \{\mathtt{p}_i^{\hat{x}\hat{v}}\}$ between $\hat{G}_k$ and a VNode $\hat{v} \in \hat{V} \setminus \hat{V}_k$, such that $\hat{x} \in \hat{V}_k$ and all $\mathtt{p}_i^{\hat{x}\hat{v}}$ terminate as the first VNode $\hat{x}$ in $\hat{V}_k$ is encountered, i.e., the only VNode from $\hat{V}_k$ that is on $\mathtt{p}_i^{\hat{x}\hat{v}}$ is $\hat{x}$.*

Observation 1: Using the expansion lemma, it can be shown that $\hat{G}_k \odot \hat{v}$ is a k-protected component if and only if there exists $k+1$ edge-disjoint paths from $\hat{G}_k$ to $\hat{v}$ in $\hat{G}$.

Lemma 1. *In an expansion $\hat{G}_k \odot \hat{v}$, a VLink in $(\hat{x}, \hat{y})^q \in \hat{E}_k$ can not be present in one of the $k+1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}}$.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

Proof. This proof is based on the observation that $\hat{v}$ may not have shortest paths to some of the VNodes in $\hat{G}_k$. We consider an arbitrary VLink $(\hat{x},\hat{y})^q \in \hat{E}_k$. There can be three possibilities: i) there are two edge-disjoint paths from $\hat{x}$ and $\hat{y}$ to $\hat{v}$, $P^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$ and $P^{\hat{y}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$, respectively, such that $(\hat{x},\hat{y})^q \notin P^{\hat{x}\hat{v}} \wedge (\hat{x},\hat{y})^q \notin P^{\hat{y}\hat{v}}$. To ensure edge disjointness, $(\hat{x},\hat{y})^q$ can only be added to any of $P^{\hat{x}\hat{v}}$ or $P^{\hat{y}\hat{v}}$. Adding $(\hat{x},\hat{y})^q$ to one of $P^{\hat{x}\hat{v}}$ or $P^{\hat{y}\hat{v}}$ increases the length of respective path. ii) There is only one path from $\hat{x}$ (or $\hat{y}$) to $\hat{v}$, $P^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$ (or $P^{\hat{y}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$), such that $(\hat{x},\hat{y})^q \notin P^{\hat{x}\hat{v}}$ (or $(\hat{x},\hat{y})^q \notin P^{\hat{y}\hat{v}}$) and there is no path from $\hat{y}$ (or $\hat{x}$) to $\hat{v}$ excluding $(\hat{x},\hat{y})^q$. Again, adding $(\hat{x},\hat{y})^q$ to $P^{\hat{x}\hat{v}}$ (or $P^{\hat{y}\hat{v}}$) does not contribute in finding a new edge-disjoint path and only increases the length of the path $P^{\hat{x}\hat{v}}$ (or $P^{\hat{y}\hat{v}}$). iii) There is no edge-disjoint path from $\hat{x}$ or $\hat{y}$ to $\hat{v}$. In this case, $(\hat{x},\hat{y})^q$ can not be present in any of the $k+1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}}$. □

Lemma 2. *In an expansion $\hat{G}_k \odot \hat{v}$, the size of the conflicting set of a VLink $(\hat{u},\hat{v})^q \in \hat{E} \setminus \hat{E}_k$ is given by* $|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}} |\mathtt{p}_i^{\hat{u}\hat{v}}|$, *where $\hat{u} \in \hat{V}_k$ and $\hat{v} \in \mathcal{N}(\hat{u})$.*

Proof. For the embedding of $\hat{G}_k \odot \hat{v}$ on G to remain connected in the presence of k SLink failures, we need to satisfy two conditions: i) at least $k+1$ edge-disjoint paths from $\hat{v}$ to $\hat{G}_k$ exist (*i.e.*, $|\mathcal{P}^{\hat{G}_k\hat{v}}| \geq k+1$), and ii) all of these paths are embedded on $k+1$ edge-disjoint paths in G. Therefore, a VLink $(\hat{u},\hat{v})^q \in \mathtt{p}_i^{\hat{u}\hat{v}}$ is conflicting with all the VLinks present in the $k+1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}} \setminus \mathtt{p}_i^{\hat{u}\hat{v}}$. This leads to $|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. □

Theorem 3. *For any VLink $(\hat{u},\hat{v})^q$, the size of a conflicting set $\chi_{\odot}^{\hat{u}\hat{v}q}$ obtained through the expansion of $\hat{G}_k \odot \hat{v}$ is less than or equal to the size of any conflicting set $\chi_{I}^{\hat{u}\hat{v}q}$ of the same VLink when computed independently, i.e., $|\chi_{E}^{\hat{u}\hat{v}q}| \leq |\chi_{I}^{\hat{u}\hat{v}q}|$.*

Proof. We consider two VNodes $\hat{u} \in \hat{V}_k$ and $\hat{v} \in \hat{V} \setminus \hat{V}_k$, such that $\hat{v} \in \mathcal{N}(\hat{u})$. When computed independently, the size of the conflicting set of $(\hat{u},\hat{v})^q$ is $|\chi_{I}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. On other hand, when we construct conflicting set through the expansion, $\hat{G}_k \odot \hat{v}$, the size of the conflicting set of the VLink $(\hat{u},\hat{v})^q \in \hat{E} \setminus \hat{E}_k$ is $|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}} |\mathtt{p}_i^{\hat{u}\hat{v}}|$ (as proven in Lemma 2). In the beginning, when $\hat{G}_k$ contains only one VNode, *i.e.*, $|\hat{V}_k| = 1$, it is obvious that $|\chi_{I}^{\hat{u}\hat{v}q}| = |\chi_{\odot}^{\hat{u}\hat{v}q}|$. For $|\hat{V}_k| > 1$, consider $\hat{x} \in \hat{V}_k$ such that $\exists \mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ contains $\hat{x}$ and $\mathtt{p}_j^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$. Since $\mathtt{p}_i^{\hat{u}\hat{v}}$ contains $\hat{x}$, according to the optimal substructure property of the shortest path, we get $\mathtt{p}_i^{\hat{u}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{x}} || \mathtt{p}_j^{\hat{x}\hat{v}}$, assuming $||$ is the path concatenation operator. Thus, $|\mathtt{p}_j^{\hat{x}\hat{v}}| < |\mathtt{p}_i^{\hat{u}\hat{v}}|$ resulting into $|\chi_{\odot}^{\hat{u}\hat{v}q}| < |\chi_{I}^{\hat{u}\hat{v}q}|$. If no such $\hat{x}$ is found, we can assume $\hat{x} = \hat{u}$ and in that case $\mathtt{p}_j^{\hat{x}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{v}}$ yielding $|\chi_{\odot}^{\hat{u}\hat{v}q}| = |\chi_{I}^{\hat{u}\hat{v}q}|$. Hence, $|\chi_{\odot}^{\hat{u}\hat{v}q}| \leq |\chi_{I}^{\hat{u}\hat{v}q}|$. □

As an example of Theorem 3, let us consider the VLink (d,e) in Fig. 2 and the VN needs to survive single SLink failure. If we compute independently, we get $\chi^{de} = \{(d,c),(c,e)\}$. When we compute through the expansion $\hat{G}_1 \odot e$ where $\hat{V}_1 = \{a,b,c,d\}$, we get $\chi^{de} = \{(c,e)\}$.

B. VLink Augmentation

As described in § III-C, we may need to augment a given VN $\bar{G}$ with parallel VLinks to transform $\bar{G}$ to a k-protected VN $\hat{G}$. Since augmented parallel VLinks increase both the number of disjointness constraints and embedding cost, it is intuitive to minimize the number of parallel VLinks. Again, we use Theorem 2 to find the pair of VNodes with less than $k+1$ edge connectivity and add parallel VLinks as needed. Assume that for each pair of adjacent VNodes $\bar{u},\bar{v} \in \bar{V}$, there are at least m edge-disjoint paths in $\bar{G}$. If $m \geq k+1$, $\bar{G}$ is at least $k+1$ edge-connected, hence no augmentation is needed. If $m < k+1$, we need to add $k+1-m$ parallel VLinks between $\bar{u}$ and $\bar{v}$. In general, $max(0,k+1-m)$ VLinks are needed for each pair of adjacent VNodes. For instance, a VN should be 3-edge connected to survive 2 SLink failures. Since there are 2 edge-disjoint paths between d and e in Fig. 2, we add a parallel VLink. Similarly, we add 2 parallel VLinks between e and f to make the VN 3-edge connected. No augmentation is required for the rest of the adjacent pair of VNodes. It can be easily shown that the number of VLinks to be augmented remains the same during the expansion, $\hat{G}_k \odot \bar{v}$. In other words, if there are $\hat{m}$ edge-disjoint paths from $\hat{G}_k$ to $\bar{v}$ in $\bar{G}$, augmentation of $max(0,k+1-\hat{m})$ parallel VLinks is needed to ensure the $k+1$ edge connectivity between $\hat{u}$ and $\bar{v}$, where $\hat{u} \in \hat{V}_k$ and $\bar{v} \in \mathcal{N}(\hat{u})$.

Theorem 4. *Given a k-protected component $\hat{G}_k$ and two arbitrary VNodes in $\bar{G}$ such that $\bar{v}_1 \notin \hat{V}_k$ and $\bar{v}_2 \notin \hat{V}_k$; $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ and $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$ are the resulting k-protected components obtained by incrementally applying expansion operator $\odot$ while considering $\bar{v}_1$ and $\bar{v}_2$ as the initial nodes, respectively; A and B are the ordered set of parallel links (ordered in the sequence they were added) augmented to $\bar{G}$ in the process of obtaining $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ and $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$, respectively: $|A| = |B|$.*

Proof. To prove the theorem, we assume without loss of generality that an initial k-protected component $\hat{G}_k$ of $\bar{G}$ consists of an arbitrarily selected VNode $\hat{u} \in \bar{V}$ with no VLinks. Let's consider two arbitrary VNodes $\bar{v}_1 \in \mathcal{N}(\hat{u})$ and $\bar{v}_2 \in \mathcal{N}(\hat{u})$. The sets of edge-disjoint paths from $\hat{u}$ to $\bar{v}_1$ and $\bar{v}_2$ are $\mathscr{P}^{\hat{u}\bar{v}_1}$ and $\mathscr{P}^{\hat{u}\bar{v}_2}$, respectively. However, a path $P^{\hat{u}\bar{v}_1} \in \mathscr{P}^{\hat{u}\bar{v}_1}$ does not need to be edge-disjoint with a path $P^{\hat{u}\bar{v}_2} \in \mathscr{P}^{\hat{u}\bar{v}_2}$. Suppose, $|\mathscr{P}^{\hat{u}\bar{v}_1}| = m_1$ and $|\mathscr{P}^{\hat{u}\bar{v}_2}| = m_2$. If either or both of m_1 and m_2 are greater than or equal to $k+1$ then the theorem is trivially proved. Hence we assume that $m_1 < k+1$ and $m_2 < k+1$. We need to show that the order of including $\bar{v}_1$ and $\bar{v}_2$ to $\hat{G}_k$ has no effect on the total number of parallel VLinks needed to get either $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ or $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$.

If we expand to $\bar{v}_1$ first, $k+1-m_1$ parallel VLinks will be augmented between $\hat{u}$ and $\bar{v}_1$ to get $\hat{G}_k \odot \bar{v}_1$. The parallel VLinks between $\hat{u}$ and $\bar{v}_1$ only contribute in increasing the connectivity between $\hat{u}$ and $\bar{v}_1$, and become part of $\hat{G}_k \odot \bar{v}_1$. Now, we consider $\bar{v}_2$. There can be two possibilities based on whether $\bar{v}_1$ is on a path $P^{\hat{u}\bar{v}_2}$ or not. If $\bar{v}_1$ is on a path $P^{\hat{u}\bar{v}_2}$, $P^{\hat{u}\bar{v}_2} = \{(\bar{u},\bar{v}_1)\} || P^{\bar{v}_1\bar{v}_2}$. According to Lemma 1, VLinks between $\hat{u}$ and $\bar{v}_1$ are in the k-protected component $\hat{G}_k \odot \bar{v}_1$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

and cannot be present in any of the m_2 edge-disjoint paths from $\hat{G}_k \odot \bar{v}_1$ to $\bar{v}_2$, thus the number of edge-disjoint paths between $\bar{v}_2$ and $\hat{G}_k \odot \bar{v}_1$ will remain the same as that of between $\bar{v}_2$ and $\hat{G}_k$. For the second possibility, when $\bar{v}_1$ is not present on any $P^{\hat{u}\bar{v}_2} \in \mathscr{P}^{\hat{u}\bar{v}_2}$, the paths in $\mathscr{P}^{\hat{u}\bar{v}_2}$ will remain unaffected by expansion of $\hat{G}_k \odot \bar{v}_1$. In both cases, the number of edge-disjoint paths from $\hat{G}_k$ to $\bar{v}_2$ remains the same. In other words, the number of parallel VLinks needed to produce $\hat{G}_k \odot \bar{v}_2$ from $\hat{G}_k$ is the same as that for obtaining $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ from $\hat{G}_k \odot \bar{v}_1$. The same can be shown if we consider $\bar{v}_2$ before $\bar{v}_1$. Therefore, the order of choosing VNodes for inclusion into $\hat{G}_k$ does not have any impact on the number of parallel VLinks to be augmented needed to get either $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ or $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$. □

C. Necessary Conditions for a Feasible VN Embedding

VN augmentation satisfies only one of the two necessary conditions for *CoViNE* that is a VN must be $k+1$ edge connected. Recall from § I that the other necessary condition for *CoViNE* is to have at least $k+1$ edge-disjoint paths between every pair of VNodes after embedding the VN on the SN. Indeed, an SN that is $k+1$ edge-connected, *i.e.*, each SNode degree is at least $k+1$, satisfies this necessary condition to support the successful embedding of an augmented VN. However, an SN without $k+1$ edge-connectivity between all SNode pairs can also support the successful embedding of an augmented VN as long as there exists at least a node mapping function $f : \bar{V} \rightarrow V$ for which each pair of SNodes $f(\bar{u})$ and $f(\bar{v})$, where $\bar{u}, \bar{v} \in \bar{V} \wedge \bar{u} \neq \bar{v}$, have at least $k+1$ edge-disjoint paths in the SN with sufficient bandwidth. In other words, the SN G must have a non-empty sub-graph with at least $|\bar{V}|$ SNodes that contains at least one SNode from the location constraint sets of each VNode $\bar{u} \in \bar{V}$ and that sub-graph is $k+1$ edge-connected. A brute-force algorithm can enumerate all such sub-graphs of G to compute the optimal solution for *CoViNE*. However, as the number of sub-graphs grows exponentially with the number of nodes in either VN or SN, such brute-force approach cannot scale. In the following, we briefly describe the factors that influence the steps that should be taken to satisfy the aforementioned conditions, which we also exploit to design a scalable heuristic.

Since parallel VLinks added to a VLink $(\hat{u}, \hat{v})$ as part of the graph augmentation provide the required edge-connectivity between the two VNodes $\hat{u}$ and $\hat{v}$, VLink $(\hat{u}, \hat{v})$ and all the parallel VLinks added to $(\hat{u}, \hat{v})$ must be embedded on mutually edge-disjoint paths in the SN. In the worst case, a VLink $(\hat{u}, \hat{v})$ can be augmented with k parallel VLinks requiring $k+1$ edge-disjoint paths between the two SNodes where $\hat{u}$ and $\hat{v}$ are mapped. In addition, conflicting sets may enforce the SPaths used for embedding the augmented VLinks to be disjoint with the embedding of other VLinks of the same VN. Therefore, some of the SPaths used for embedding augmented VLinks can have a large number of SLinks to ensure disjointness with other SPaths, thus resulting in a higher embedding cost. Although the augmentation process described in § V-B ensures that the number of augmented VLinks remains constant irrespective of initial VNode choice and the subsequent order, the decision of which VLinks to augment is still a combinatorial optimization problem as discussed in § IV. This decision of which VLink to augment can have an impact on the subsequent VN embedding since one combination of augmentation may lead to an infeasible embedding due to the lack of sufficient edge-disjoint paths in the SN, while another combination may result in a higher embedding cost due to using longer edge-disjoint paths. To address these issues, we develop a heuristic algorithm in § VI-A that takes both the existence of sufficient number of edge-disjoint paths in the SN and the number of SLinks present in those edge-disjoint paths into account while expanding a k-protected component $\hat{G}_k$. The heuristic algorithm in § VI-A also computes conflicting sets for each VLink by expanding a k-protected component $\hat{G}_k$ as discussed in § V-A. These conflicting sets can then be used by any embedding algorithm to ensure the existence of $k+1$ edge-disjoint paths between every pair of VNodes in the VN embedding. We also present an ILP formulation and a heuristic algorithm in § VI-B and § VI-C, respectively, that leverage conflicting sets of VLinks to compute embedding while satisfying the conditions discussed in this section.

VI. Sequential Solutions to CoViNE

Due to the intractability of *CoViNE-opt*, in this section, we present two sequential solutions to *CoViNE*. The first solution consists of a heuristic algorithm (Alg. 1) and a simplified formulation of *CoViNE-opt*, namely *CoViNE-ILP*. In this solution, we delegate the task of transforming a VN $\bar{G}$ to a k protected VN $\hat{G}$ and computing conflicting set $\chi^{\hat{G}}$ of $\hat{G}$ to Alg. 1 as described in § VI-A. The outputs ($\hat{G}$ and $\chi^{\hat{G}}$) of Alg. 1 are then fed into CoViNE-ILP that embeds a VN on an SN using a multi-commodity flow formulation as presented in § VI-B. However, *CoViNE-ILP* cannot scale to larger VNs and SNs due to the limitations of LP solvers. Hence, we propose a heuristic algorithm (Alg. 2) in § VI-C that takes $\hat{G}$ and $\chi^{\hat{G}}$ as inputs and embeds VNodes and disjointness constrained VLinks of $\hat{G}$ in a coordinated manner. Combining Alg. 1 and Alg. 2, we get our last solution, *CoViNE-fast*, that can solve larger problem instances within a reasonable time.

A. Heuristic Algorithm for Conflicting Set and Augmentation

Alg. 1 starts with a seed k-protected component, $\hat{G}_k$, containing an arbitrary VNode $\bar{u} \in \bar{V}$ with degree at least $k+1$. Then the algorithm adds all of $\bar{u}$'s neighbors $\bar{v} \in \mathcal{N}(\bar{u})$ to $\hat{V}_k$ in the increasing order of a score computed for each VLink $(\bar{u}, \bar{v})$. The score function assigns each VLink $(\bar{u}, \bar{v})$ a value proportional to an estimated cost of embedding $(\bar{u}, \bar{v})$ along with all its augmented VLinks (if necessary) on the SN. Since $\bar{u}$ already belongs to a k-protected component $\hat{G}_k$, the number of additional VLinks that need to be augmented for a VLink $(\bar{u}, \bar{v})$ *s.t.*, $\bar{v} \in \mathcal{N}(\bar{u})$, depends on the degree of $\bar{v}$ denoted by $degree(\bar{v})$. In the worst case, the VLink $(\bar{u}, \bar{v})$ needs to be augmnted with $max((k+1-degree(\bar{v})), 0)$ parallel VLinks as $degree(\bar{v})$ imposes an upper bound on the number of EDSPs between $\bar{u}$ and $\bar{v}$. When $degree(\bar{v}) \geq k+1$, no augmentation is needed for the VLink $(\bar{u}, \bar{v})$ and the score function assigns a large value for the estimated embedding cost

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

Algorithm 1 Compute Conflicting Sets and Augmentation

1: **function** CONFLICTINGSETSAUGMENTATION($\bar{G}, k$)
2: $\forall(\bar{u},\bar{v}) \in \bar{E}$: $\chi^{\bar{u}\bar{v}} \leftarrow \phi$, $a_0^{\bar{u}\bar{v}} \leftarrow 1$, $Q \leftarrow \phi$
3: // $\bar{v}$ is an arbitrary VNode with degree$\geq k+1$
4: $\exists \bar{v} \in \bar{V} : \hat{G}_k \leftarrow (\{\bar{v}\}, \phi)$
5: ENQUEUE(Q, $\bar{v}$)
6: **while** Q is not empty **do**
7: $\bar{u} \leftarrow DEQUEUE(Q)$
8: $\Sigma \leftarrow \{(\bar{u},\bar{v}) | \bar{v} \in \mathcal{N}(\bar{u}) \wedge \bar{v} \notin \hat{V}_k\}$
9: **for all** $(\bar{u},\bar{v}) \in \Sigma$ **do**
10: **if** $degree(\bar{v}) \geq k+1$ **then** $score(\bar{u},\bar{v}) \leftarrow \infty$
11: **else**
12: **for all** $(l,m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ **do**
13: $\mathcal{P}^{lm} \leftarrow$ EDSP$(G, l, m, k+2-degree(\bar{v}))$
14: **if** $|\mathcal{P}^{lm}| < k+2-degree(\bar{v})$ **then**
15: $rank(l,m) \leftarrow \infty$
16: **else**
17: $rank(l,m) \leftarrow \sum_{p \in \mathcal{P}^{lm}} |p|$
18: $score(\bar{u},\bar{v}) \leftarrow \min_{(l,m) \in L(\hat{u}) \times L(\hat{v})} rank(l,m) b_{\bar{u}\bar{v}}$
19: $\bar{\mathcal{E}} \leftarrow$ Sort Σ in increasing order of $score(\bar{u},\bar{v})$
20: **for all** $(\bar{u},\bar{v}) \in \bar{\mathcal{E}}$ **do**
21: $\mathcal{P}^{\hat{G}_k\bar{v}} \leftarrow$ EDSP$(\bar{G}, \hat{G}_k, \bar{u}, \bar{v}, k+1)$
22: **for** $i = 1 \rightarrow (k+1-|\mathcal{P}^{\hat{G}_k\bar{v}}|)$ **do**
23: $\hat{E} \leftarrow \bar{E} \cup (\bar{u},\bar{v})^i$, $a_k^{\bar{u}\bar{v}} \leftarrow 1$
24: $\mathcal{P}^{\hat{G}_k\bar{v}} \leftarrow \mathcal{P}^{\hat{G}_k\bar{v}} \cup (\bar{u},\bar{v})^i$
25: **for all** $\mathrm{p}_i^{\hat{G}_k\bar{v}} \in \mathcal{P}^{\hat{G}_k\bar{v}}$ **do**
26: **for all** $(\bar{x},\bar{y})^q \in \mathrm{p}_i^{\hat{G}_k\bar{v}}$ **do**
27: **for all** $\mathrm{p}_j^{\hat{G}_k\bar{v}} \in \mathcal{P}^{\hat{G}_k\bar{v}} | j \neq i$ **do**
28: $\chi^{\bar{x}\bar{y}q} \leftarrow \chi^{\bar{x}\bar{y}} \cup \{\forall(\bar{s},\bar{t})^r \in \mathrm{p}_j^{\hat{G}_k\bar{v}}\}$
29: $\hat{G}_k \leftarrow \hat{G}_k \odot \bar{v}$, ENQUEUE(Q, $\bar{v}$)
30: **return** $\chi^{\bar{G}}$

of $(\bar{u},\bar{v})$ to make it appear at the end of the sorted order (Line 10). Otherwise, when $degree(\bar{v}) < k+1$, the VLink $(\bar{u},\bar{v})$ and all the $k+1-degree(\bar{v})$ parallel VLinks need to be embedded on mutually disjoint SPaths following the discussion in § V-C. Therefore, Alg. 1 computes the estimated embedding cost for $(\bar{u},\bar{v})$ by multiplying the bandwidth demand $b_{\bar{u}\bar{v}}$ with the number of SLinks present in the $k+2-degree(\bar{v})$ EDSPs in the SN that yields the lowest cost embedding for $(\bar{u},\bar{v})$ and its up to $k+1-degree(\bar{v})$ augmented VLinks (Line 18).

To estimate the lowest embedding cost for $(\bar{u},\bar{v})$ and its up to $k+1-degree(\bar{v})$ augmented parallel VLinks, Alg. 1 computes the sets of $k+2-degree(\bar{v})$ EDSPs in the SN using EDSP procedure for each possible pair of SNodes where VNodes $\bar{u}$ and $\bar{v}$ can be mapped. For a pair of such SNodes l and m, EDSP iteratively applies Dijkstra's shortest path algorithm [58] to compute a set of $k+2-degree(\bar{v})$ EDSPs as $\mathcal{P}^{lm}$ (Line 13). After computing each EDSP, all the SLinks present in the EDSP are removed from G in order to ensure the edge-disjointness of the subsequent paths. Alg. 1 then uses the set of $k+2-degree(\bar{v})$ EDSPs with the least number of SLinks over all pairs of $(l,m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ to compute the estimated embedding cost of $(\bar{u},\bar{v})$. Such sorted order of VLinks ensures that parallel VLinks are augmented to VLinks of an MST $\hat{T}$ of $\hat{G}_k$ to ensure necessary connectivity with the lowest estimated cost whenever possible. Such SN awareness also rules out the possibility of augmenting a VLink $(\bar{u},\bar{v})$ with $k+1-degree(\bar{v})$ parallel VLinks if none of the SNode pairs $(l,m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ has at least $k+2-degree(\bar{v})$ EDSPs (Line 15). The final node and link embedding are computed by Alg. 2 that neither uses the estimated embedding cost nor the mappings computed by Alg. 1.

The above process is repeated until all the VNodes of $\bar{G}$ are added to $\hat{G}_k$. For each $\bar{v}$, Alg. 1 computes $k+1$ EDSPs in the VN, $\mathcal{P}^{\hat{G}_k\bar{v}}$ between $\hat{G}_k$ and $\bar{v}$ using EDSP procedure (Line 21). EDSP initially selects the VLink, $(\bar{u},\bar{v})$ as the first shortest path $\mathrm{p}_1^{\hat{G}_k\bar{v}}$ to $\mathcal{P}^{\hat{G}_k\bar{v}}$. It then invokes *Dijkstra's shortest path* algorithm [58] k times to compute $\mathrm{p}_i^{\hat{G}_k\bar{v}}$, the i-th EDSP between $\hat{G}_k$ and $\bar{v}$. After computing each $\mathrm{p}_i^{\hat{G}_k\bar{v}}$, all VLinks present in $\mathrm{p}_i^{\hat{G}_k\bar{v}}$ are removed from $\bar{G}$ in order to ensure the edge-disjointness of the later paths. *Dijkstra's shortest path* algorithm is modified to use the bandwidth demand of a VLink as the VLink weight while computing EDSPs. Alg. 1 then proceeds to check if any augmentation is required between $\hat{G}_k$ and $\bar{v}$. To do so, it counts the number of EDSPs, $|\mathcal{P}^{\hat{G}_k\bar{v}}|$, computed in line 21. If the number of EDSPs is less than $k+1$, line 23 of Alg. 1 adds $k+1-|\mathcal{P}^{\hat{G}_k\bar{v}}|$ parallel VLinks between $\bar{u}$ and $\bar{v}$. The i-th parallel VLink, denoted by $(\bar{u},\bar{v})^i$, constitutes the $(|\mathcal{P}^{\hat{G}_k\bar{v}}|+i)$-th EDSP between $\hat{G}_k$ and $\bar{v}$. Finally, Alg. 1 updates the conflicting sets of the corresponding VLinks as described in Lemma 2 (Line 28).

1) Discussion: When augmentation is needed, Alg. 1 invokes EDSP procedure $O(|\bar{V}||\mathcal{N}(\bar{u})||L(\bar{u})|^2)$ times. EDSP invokes *Dijkstra's shortest path* algorithm on SN G $k+1$ times in the worst case. The time complexity of *Dijkstra's shortest path* algorithm on SN G based on a min-priority queue is $O(|E|+|V|\log|V|)$. Therefore, calls to EDSP requires running time of $O((k+1)|\bar{V}||\mathcal{N}(\bar{u})||L(\bar{u})|^2(|E|+|V|\log|V|))$. When augmentation is not required, the time complexity of Alg. 1 is dominated by the EDSP procedure that is invoked $O(|\bar{V}||\mathcal{N}(\bar{u})|)$ times. EDSP invokes *Dijkstra's shortest path* algorithm on VN $\bar{G}$ $k+1$ times yielding $O((k+1)(|\bar{E}|+|\bar{V}|\log|\bar{V}|))$ running time. In this case, the running time of Alg. 1 becomes $O((k+1)|\bar{V}||\mathcal{N}(\bar{u})|(|\bar{E}|+|\bar{V}|\log|\bar{V}|))$.

An alternate way for implementing EDSP is to use a maximum flow algorithm [59]. However, the iterative shortest path based method and the maximum flow algorithm have similar running time and solution quality [59]. The only scenario where maximum flow algorithms are known to be more effective is when the graph has a *trap* topology [59]. Dunn *et al.*, have shown that the formation of trap topologies require many conditions to be met simultaneously, hence, they are very rare in production networks [59]. Therefore, taking any special measures for such topology seems unnecessary. Consequently, we made an arbitrary choice and resorted to using the iterative application of Dijkstra's shortest path algorithm for implementing EDSP.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

B. CoViNE-ILP: ILP formulation for CoViNE Embedding

CoViNE-ILP, takes a k-protected VN $\hat{G}$ and its conflicting set $\chi^{\hat{G}}$ as inputs. It embeds $\hat{G}$ on an SN G while ensuring the disjointness constraints imposed by $\chi^{\hat{G}}$ and minimizing cost according to (17). *CoViNE-ILP* is similar to *CoViNE-opt*, excluding the variables and constraints related to the augmentation and disjointness requirement computation. *CoViNE-ILP* does not use the augmentation variable (*i.e.*, $a_k^{\bar{u}\bar{v}}$) and constraint (2) of *CoViNE-opt* since augmentation is performed by Alg. 1. Therefore, *CoViNE-ILP* replaces (4) by the following:

$$\forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\bar{u}\bar{v}k} - x_{vu}^{\bar{u}\bar{v}k}) = y_{\bar{u}u} - y_{\bar{v}u} \tag{18}$$

Furthermore, *CoViNE-ILP* uses the disjointness constraints imposed by pre-computed $\chi^{\hat{G}}$. Hence, it eliminates disjoint group assignment variable $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ and replaces disjointness constraints (12) and (13) from *CoViNE-opt* with the following:

$$\forall(u,v) \in E, \forall(\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall(\bar{a},\bar{b})^q \in \chi^{\bar{u}\bar{v}k} : x_{uv}^{\bar{u}\bar{v}k} + x_{vu}^{\bar{u}\bar{v}k} + x_{uv}^{\bar{a}\bar{b}q} + x_{vu}^{\bar{a}\bar{b}q} \leq 1 \tag{19}$$

(19) ensures that VLink embedding of $(\hat{u},\hat{v})^k \in \hat{E}$ will never share an SLink with the embeddings of the conflicting VLinks present in $\chi^{\hat{u}\hat{v}k}$, thus satisfying the disjointness relation among them. If the SN does not have sufficient number of EDSPs to satisfy the disjointness constraint (19), *CoViNE-ILP* becomes infeasible rendering no solution for the embedding. The total number of binary variables for $x_{uv}^{\bar{u}\bar{v}k}$ is $|\bar{E}|\times k\times|E|$ and total number of constraints generated by (19) is $k \times |E| \times |\bar{E}|^2$. Therefore, unlike *CoViNE-opt*, which has an exponential number of variables and constraints, *CoViNE-ILP* has a polynomial number of variables and constraints. Note that even with the polynomial number of variables and constraints *CoViNE-ILP* is intractable as the worst case time complexity is $O((k \times |E| \times |\bar{E}|^2) \times 2^{(|\bar{E}|\times k\times|E|)})$.

C. Heuristic Algorithm for CoViNE Embedding

Alg. 2 computes two functions, $nmap$ and $emap$, which represent the VNode and VLink mapping of $\hat{G}$ on G, respectively. Since there is no cost associated with VNode mapping, a VLink mapping that minimizes total cost determines the VNode mapping. As discussed in § V-A, disjointness constraints imposed by conflicting sets may lead to infeasible solutions. Hence, Alg. 2 prioritizes finding a feasible mapping than minimizing the embedding cost. Following this intuition, Alg. 2 first sorts the VNodes $\hat{u} \in \hat{V}$ in decreasing order of the sum of conflicting set sizes of incident VLinks. This sorted list of VNodes is represented by $\hat{\mathcal{V}}$. Since a VNode with VLinks having larger conflicting sets becomes too constrained to be mapped to a suitable SNode, Alg. 2 tries to map VNodes in the order of $\hat{\mathcal{V}}$. However, Alg. 2 maps a VNode from $\hat{\mathcal{V}}$ to the candidate SNode that satisfies the disjointness constraints and minimizes the embedding cost.

For each VNode $\hat{u} \in \hat{\mathcal{V}}$, Alg. 2 searches for an unallocated SNode in $\hat{u}$'s location constraint set, $L(\hat{u})$, which yields a feasible mapping with the minimum cost. To embed $\hat{u}$, Alg. 2 loops through each candidate SNode $l \in L(\hat{u})$ (Line $5-13$), to first temporarily map $\hat{u}$ to l (Line 6). Then the algorithm computes temporary mappings for all the VLinks incident to $\hat{u}$. The `VL-MAP` (Alg. 3) procedure is invoked to find the mapping for each such VLink (Line 9). VLinks incident to $\hat{u}$ are picked in decreasing order of their conflicting set sizes to maximize the chances of finding a feasible solution. Alg. 2 finally embeds $\hat{u}$ to the candidate l that leads to a feasible mapping for all the VLinks incident to $\hat{u}$ and yields the minimum embedding cost (Line 15). The algorithm fails, if no such feasible l is found. Once a VNode $\hat{u}$ has been finally mapped, Alg. 2 creates the final mapping for only those VLinks incident to $\hat{u}$ whose both endpoints are already finally mapped (Line 17). If we map a VLink with one unmapped endpoint (*e.g.*, , $\hat{v}$), we have to map $\hat{v}$ based on this local information. This would reduce the degrees of freedom for $\hat{v}$ when other VLinks incident to $\hat{v}$ would have been mapped,

Algorithm 2 Compute VN Embedding

1: **function** VN-EMBEDDING($G, \hat{G}, \chi^{\hat{G}}$)
2: $\hat{\mathcal{V}} \leftarrow$ Sort $\hat{u} \in \hat{V}$ in decreasing order of $\sum_{\forall \hat{v}\in\mathcal{N}(\hat{u})} \sum_{\forall k\in K} |\chi^{\hat{u}\hat{v}k}|$
3: **for all** $\hat{u} \in \hat{\mathcal{V}}$ **do**
4: $Candidate \leftarrow \phi$
5: **for all** $l \in L(\hat{u})$ **do**
6: $nmap(\hat{u}) \leftarrow l$
7: $\mathcal{E} \leftarrow$ Sort $(\hat{u},\hat{v})^q \in \hat{E}$ in dec. order of $|\chi^{\hat{u}\hat{v}q}|$
8: **for all** $(\hat{u},\hat{v})^q \in \mathcal{E}|\ emap(\hat{u},\hat{v})^q = \phi$ **do**
9: $P[(\hat{u},\hat{v})^q] \leftarrow$ VL-MAP($G, \hat{G}, \chi^{\hat{G}}, (\hat{u},\hat{v})^q$)
10: **if** $\sum_{\forall(\hat{u},\hat{v})^q\in\mathcal{E}} cost(P[(\hat{u},\hat{v})^q])$ is minimum **then**
11: $M \leftarrow P$, $Candidate \leftarrow l$
12: $nmap(\hat{u}) \leftarrow \phi$
13: $\forall(\hat{u},\hat{v})^q \in \mathcal{E}$: $emap(\hat{u},\hat{v})^q \leftarrow \phi$
14: **if** $Candidate \neq \phi$ **then**
15: Add mapping $\hat{u} \rightarrow Candidate$ to $nmap$
16: $\forall(\hat{u},\hat{v})^q \in \mathcal{E}|nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) \neq \phi$:
17: Add $(\hat{u},\hat{v})^q \rightarrow M[(\hat{u},\hat{v})^q]$ to $emap$
18: **else return** No Solution Found
19: **return** $\{nmap, emap\}$

Algorithm 3 Compute VLink Mapping

1: **function** VL-MAP($G, \hat{G}, \chi^{\hat{G}}, (\hat{u},\hat{v})^q$)
2: $p^{\hat{u}\hat{v}} \leftarrow \phi$
3: **for all** $(\hat{s},\hat{t})^r \in \chi^{\hat{u}\hat{v}q}$: **do**
4: $E \leftarrow E - \{(a,b) \in E|(\hat{s},\hat{t})^r$ is mapped to $(a,b)\}$
5: **if** $nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) \neq \phi$ **then**
6: $Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow$ MP($G, nmap(\hat{u}), nmap(\hat{v}), b_{\hat{u}\hat{v}}$)
7: **else if** $nmap(\hat{u}) = \phi \wedge nmap(\hat{v}) \neq \phi$ **then**
8: $Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow \min_{\forall l\in L(\hat{u})}$ {MP($G, l, nmap(\hat{v}), b_{\hat{u}\hat{v}}$)}
9: **else if** $nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) = \phi$ **then**
10: $Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow \min_{\forall l\in L(\hat{v})}$ {MP($G, nmap(\hat{u}), l, b_{\hat{u}\hat{v}}$)}
11: **if** $Q^{nmap(\hat{u})nmap(\hat{v})} \neq \phi$ **then**
12: Add $(\hat{u},\hat{v})^q \rightarrow Q^{nmap(\hat{u})nmap(\hat{v})}$ to $emap$
13: **return** $Q^{nmap(\hat{u})nmap(\hat{v})}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

and may lead to infeasible solution. Mappings of such VLinks incident to $\hat{u}$ are finalized when their unmapped endpoints are finally mapped.

We now describe the VL−MAP (Alg. 3) procedure for finding the mapping of a VLink, $(\hat{u},\hat{v})^q$. First we remove all the SLinks used by the mappings of all the VLinks in $\chi^{\hat{u}\hat{v}q}$ to satisfy the disjointness constraints (Line 4). Then, we compute mapping for $(\hat{u},\hat{v})^q$ by considering the following two cases: (i) both endpoints of $(\hat{u},\ \hat{v})^q$ have already been mapped to some SNodes. In this case, we find a minimum cost path between $nmap(\hat{u})$ and $nmap(\hat{v})$ with capacity at least $b_{\hat{u}\hat{v}}$ in G (Line 6); (ii) only $\hat{u}$ (or $\hat{v}$) is mapped and the other endpoint $\hat{v}$ (or $\hat{u}$) has not been mapped. In this case, we compute the minimum cost path between $nmap(\hat{u})$ (or $nmap(\hat{v})$) and all possible locations for the unmapped VNode $\hat{v}$ (or $\hat{u}$), $l \in L(\hat{v})$ (or $L(\hat{u})$) with at least $b_{\hat{u}\hat{v}}$ capacity (Line 8 (or Line 10)). The VLink $(\hat{u},\hat{v})^q$ is temporarily mapped to the computed path and the mapping is added to $emap$ (Line 12). We modified *Dijkstra's shortest path* algorithm [58] to consider SLink capacities, while computing the minimum-cost path (MP procedure call in Alg. 3). The cost of each SLink $(u,v) \in E$ is set to $C_{uv} \times b_{\hat{u}\hat{v}}$, where $b_{\hat{u}\hat{v}}$ is the bandwidth requirement of the VLink to be embedded.

1) Discussion: The most computationally expensive step of Alg. 2 is the VL−MAP procedure, which invokes *Dijkstra's shortest path* algorithm requiring $O(|E| + |V|\log|V|)$ time. Since VL−MAP is invoked $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|)$ times, the running time of Alg. 2 is $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|(|E|+|V|\log|V|))$. Unlike most of the approaches in the literature that perform VNode and VLink mapping sequentially [60], Alg. 2 addresses VNode mapping and disjointness constrained VLink mapping simultaneously. However, Alg. 2 maps VNodes of a VN (or, the VLinks incident to a VNode) one-by-one, starting from the most constrained VNode (or, VLink) to the least constrained one. Although these orders increase the chances of finding a feasible solution, they may lead to sub-optimal solutions. Meta-heuristic approaches can achieve better orders by exploring larger solution space at the cost of increased execution time [61].

VII. Evaluation

We evaluate our proposed solutions for *CoViNE* through extensive simulations. We briefly discuss the compared approaches in § VII-A, simulation setup in § VII-C followed by the performance metrics in § VII-B. Finally, we describe our evaluation results in § VII-D focusing on optimality, scalability, embedding performance, and failure restoration.

A. Compared Approaches

We compare the performance of our sub-optimal solutions (*CoViNE-ILP* and *CoViNE-fast*) to the optimal solution, *CoViNE-opt* under single ($k = 1$) and double ($k = 2$) SLink failure scenarios. We have restricted our failure scenarios to double link failures, since the possibility of more than two simultaneous failures is extremely low in practice [7], [24]. To measure how much extra resources are needed to guarantee connectivity under different failure scenarios, we compare our approaches with an optimal VNE algorithm [4] that does not ensure any connectivity during substrate failure ($k = 0$). Table II summarizes these approaches by listing the failure scenarios and names of the ILP and algorithm they use along with their worst case time complexities.

B. Performance Metrics

1) Embedding Cost: The cost of provisioning bandwidth for the VLinks in a VN, computed using (17).

2) Execution Time: The time required for an algorithm to find the solution for CoViNE.

3) Restored Bandwidth: The percentage of VLinks' bandwidth that is restored after these VLinks have been affected by one or more SLink failure(s).

C. Simulation Setup

We implement the ILP formulations of *CoViNE-opt* and *CoViNE-ILP* using IBM ILOG CPLEX C++ library and Alg. 1 and Alg. 2 using C++. The evaluation was performed on a machine with 8×10-core hyper-threaded Intel Xeon E7-8870 2.40GHz CPU and 2TB RAM. To demonstrate the scalability of our solutions, we consider both small and large network topologies summarized in Table III. For small and large cases, we vary both the size and LNR of SNs and VNs to assess the robustness of our solutions. Note that the problem instances have been selected by studying ISP network measurement research literature [62]–[64] and consulting with our industry partners. For each problem instance in Table III, with a given SN size and VN size, we generate 3 VNs for each generated SN, and execute the compared approaches in Table II to embed each VN on the SN independently. For a particular VN and SN, the source and destination of an SLink or a VLink and the location constraints of VNodes are chosen randomly, and VLink demands are set to 10% of the SLink bandwidths. For each VN, we measure the performance metrics and plot the metrics' average value with errorbar showing the maximum and the minimum values. To analyze the impact of using different algorithms for solving VN embedding (Fig. 4), we use star, ring, and randomly connected VN topologies on anonymous inter-continental network topologies with varying Link-to-Node Ratios (LNRs). However, due to huge cost of deploying inter-continental links, these SNs do not have enough LNRs to guarantee necessary edge-connectivity against double link failures. Therefore, we do not evaluate double link failure scenarios for this analysis. In addition, we demonstrate how our approaches can survive affected VLinks' bandwidth in the presence of single and double SLink failures (Fig. 6).

D. Results

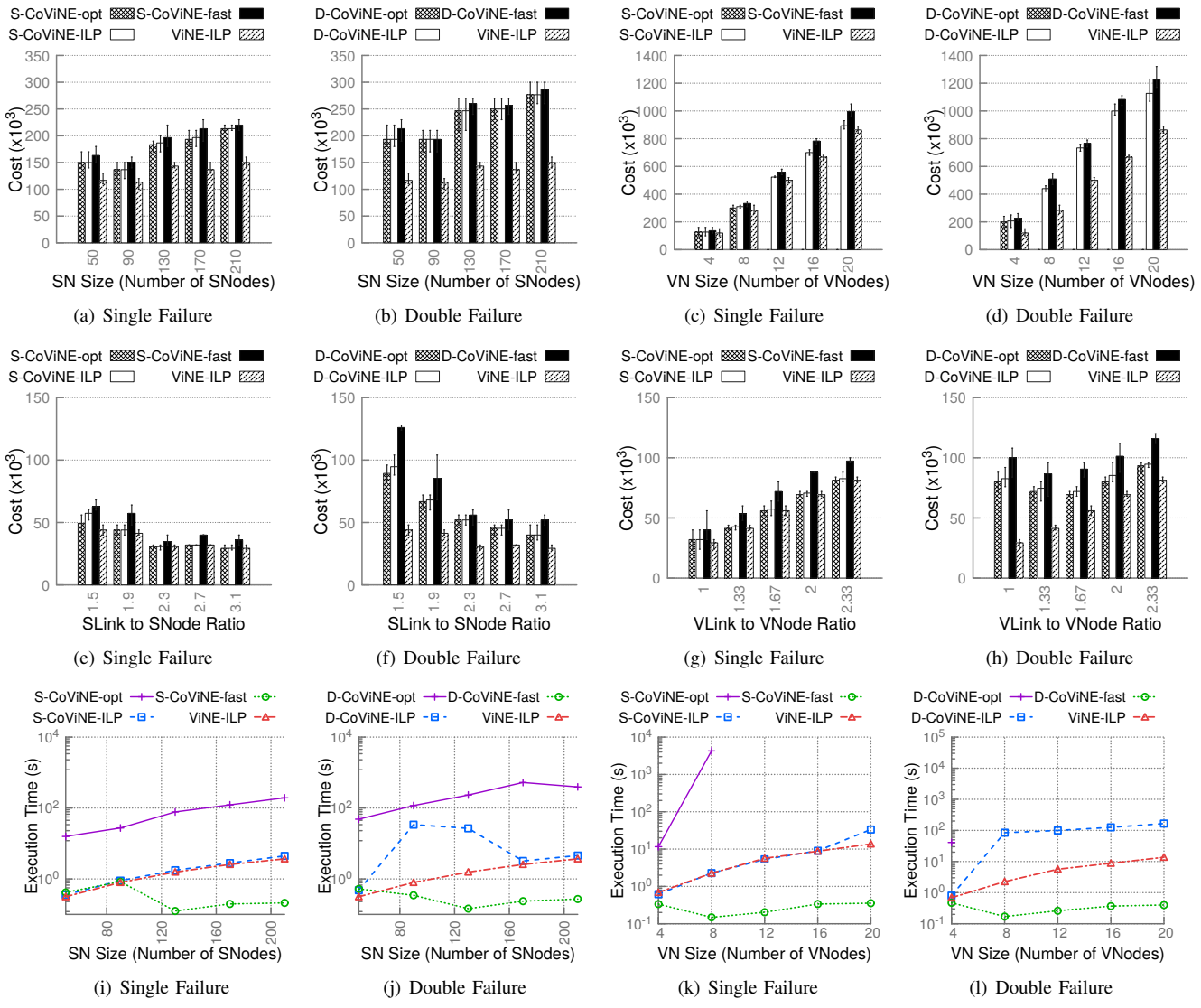
1) Small Scale Scenarios:

a) Optimality Analysis: Fig. 3(a)–3(d) compares embedding cost for guaranteeing connectivity against different failure scenarios ($k = 0, 1$, and 2) by varying SN and VN sizes, while Fig. 3(e)–3(h) presents embedding cost by varying SN and VN LNRs. Fig. 3(a) and Fig. 3(b) show that cost increases for all the compared approaches with the increase in SN size. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

TABLE II
COMPARED APPROACHES

Failure Scenario	Notation	Augmentation & conflicting set computation	Embedding	Worst case time complexity
$k=0$	ViNE-ILP [4]	None	ILP of MCUFP	$O((\lvert E\rvert \times \lvert\bar{E}\rvert) \times 2^{(\lvert E\rvert \times \lvert E\rvert)})$
$k=1$	S-CoViNE-opt	CoViNE-opt	CoViNE-opt	$O((\lvert E\rvert \times (2^{\lvert\bar{V}\rvert}-2)) \times 2^{(\lvert\bar{E}\rvert \times (2^{\lvert V\rvert}-2))})$
	S-CoViNE-ILP	Alg. 1	CoViNE-ILP	$O((\lvert E\rvert \times \lvert\bar{E}\rvert^2) \times 2^{(\lvert\bar{E}\rvert \times \lvert E\rvert)})$
	S-CoViNE-fast	Alg. 1	Alg. 2	$O(2 \times \lvert V\rvert \times \lvert\mathcal{N}(\bar{u})\rvert \times \lvert L(\bar{u})\rvert^2 \times (\lvert E\rvert + \lvert V\rvert \times \log\lvert V\rvert))$
$k=2$	D-CoViNE-opt	CoViNE-opt	CoViNE-opt	$O((\lvert E\rvert \times (2^{\lvert\bar{V}\rvert}-2)) \times 2^{(\lvert\bar{E}\rvert \times 2 \times (2^{\lvert V\rvert}-2))})$
	D-CoViNE-ILP	Alg. 1	CoViNE-ILP	$O((2 \times \lvert E\rvert \times \lvert\bar{E}\rvert^2) \times 2^{(\lvert E\rvert \times 2 \times \lvert E\rvert)})$
	D-CoViNE-fast	Alg. 1	Alg. 2	$O(3 \times \lvert V\rvert \times \lvert\mathcal{N}(\bar{u})\rvert \times \lvert L(\bar{u})\rvert^2 \times (\lvert E\rvert + \lvert V\rvert \times \log\lvert V\rvert))$



(a) Single Failure (b) Double Failure (c) Single Failure (d) Double Failure

(e) Single Failure (f) Double Failure (g) Single Failure (h) Double Failure

(i) Single Failure (j) Double Failure (k) Single Failure (l) Double Failure

Fig. 3. Embedding Cost (varying both topology size and LNR) and Execution Time Analysis for Small Scale Topologies

is due to the fact that location constraints of the VNodes of a VN are placed far apart from one another in a larger SN, thus involving more SLinks in the substrate paths for mapping VLinks of the VN and consuming more substrate resources along those SLinks. An opposite trend is observed when SN LNR is increased in Fig. 3(e) and Fig. 3(f). Incrementing SN density increases substrate path diversity with higher number of shorter and disjoint paths that result in lower embedding costs. In constrast, increasing VN size or VN LNR escalates embedding cost in general as seen in Fig. 3(c), Fig. 3(d), and Fig. 3(g) except for double link failure scenario in Fig. 3(h). This stems from the fact that a larger or denser VN has a higher number of VLinks to be embedded that increase substrate resource consumption. In contrast, sparse VNs require higher

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications

TABLE III
SUMMARY OF SIMULATION PARAMETERS

Scenario	Figure(s)	SNodes	SLinks	VNodes	VLinks
Small Scale	3(a), 3(b), 3(i), 3(j)	50-210	105-412	5	7
	3(c), 3(d), 3(k), 3(l)	100	198	4-20	5-37
	3(e), 3(f)	25	38-78	5	7
	3(g), 3(h)	25	46	6	6-14
	4	29-158	35-303	3-20	3-40
Large Scale	5(a)	500	2017	10-100	21-285
	5(d)	1000	4023	10-100	21-285
	5(b)	500	1500	10	11-31
	5(e)	1000	2000	10	11-31
	5(c)	500	1000-2000	10	21
	5(f)	1000	2000-4000	10	21
Failure Restoration	6	150	310	10	11-31

number of VLinks to be augmented (see Fig. 6(d)) to guarantee connectivity against double link failures that results in higher cost for embedding the augmented VLinks at lower VN LNRs.

Among the approaches that guarantee connectivity for single failure scenario (*i.e.*, $k = 1$) as shown in Fig. 3(a) and Fig. 3(c), *S-CoViNE-opt* generates the lowest cost of embedding. Such behavior is expected since *S-CoViNE-opt* jointly optimizes all the subproblems of *CoViNE* yielding the optimal cost, whereas *S-CoViNE-ILP* and *S-CoViNE-fast* address them sequentially. However, as can be seen in Fig. 3(c), *S-CoViNE-opt* does not scale beyond a VN of size 8 due to its dependency on the exponential number of edge-cuts in the VN. *S-CoViNE-ILP* eliminates this dependency by leveraging conflicting set of a VN and adopting a heuristic algorithm for performing augmentation and computing disjointness requirement. Despite using a heuristic for solving part of CoViNE, the embedding costs of *S-CoViNE-ILP* remain very close (within $\sim 3\%$ on average) to those of *S-CoViNE-opt*. In contrast, *S-CoViNE-fast* employs heuristic algorithms for all the sub-problems of *CoViNE* and incurs $\sim$18% and $\sim$16% additional cost compared to *S-CoViNE-opt* and *S-CoViNE-ILP*, respectively. Slightly higher optimality gaps are observed among the solutions for double failure scenarios (*i.e.*, $k = 2$) in Fig. 3(b) and Fig. 3(d) compared to single failure cases. In particular, the optimality gap between *D-CoViNE-ILP* and *D-CoViNE-opt* is $\sim 4\%$, whereas the same between *D-CoViNE-fast* and *D-CoViNE-ILP* is $\sim 19\%$. The increased optimality gaps among the solutions for $k = 2$ is due to having more augmentation and disjointness constraints than the single failure solutions.

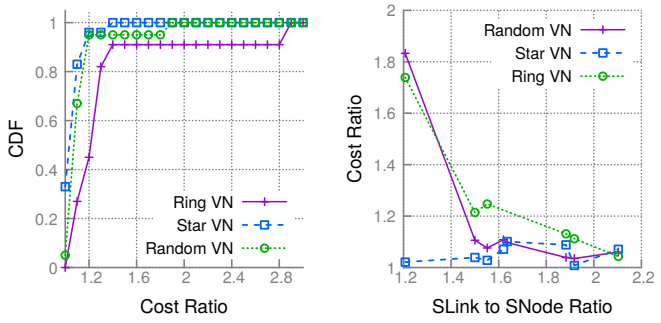
b) Embedding Cost for Different Failure Scenarios: A comparison of costs between single and double failure scenarios of any solution (*e.g.*, *S-CoViNE-opt* and *D-CoViNE-opt*) in Fig. 3(a)–3(d) reveals that ensuring connectivity against a higher degree of failure ($k = 2$) incurs a higher embedding cost. The reasons are twofold. First, *D-CoViNE-opt*, as well as *D-CoViNE-ILP* and *D-CoViNE-fast*, require more parallel VLinks to be added than their single failure counterparts to ensure the necessary edge-connectivity in the VN, consuming additional substrate resources for embedding the augmented VLinks. Second, solutions for double SLink failures require more disjointness constraints to be satisfied to preserve necessary edge-connectivity in the embedding of the VN, resulting in longer substrate paths for VLink mapping. Following these arguments, *ViNE-ILP* (with $k = 0$) produces the lowest cost of embedding, since it neither augments any VLink nor imposes any disjointness constraint. Empirically, *D-CoViNE-opt* incurs $\sim$24% and $\sim$66% more cost than *S-CoViNE-opt* and *ViNE-ILP*, respectively.

c) Scalability Analysis: Fig. 3(i)–3(l) reports execution times in logarithmic scale for guaranteeing connectivity against different failure scenarios by varying the SN and VN sizes. These figures show that the execution times of all the approaches, relying on ILP formulation for part of the problem, increase exponentially with increasing problem size. In contrast, execution times of the heuristic (*i.e.*, *S-CoViNE-fast* and *D-CoViNE-fast*) remain well within a second for similar problem instances. Furthermore, *S-CoViNE-opt* and *D-CoViNE-opt* are the slowest among the approaches for single and double failure scenarios, respectively, due to the intricacy of *CoViNE-opt* as discussed in § IV-D. In addition, *D-CoViNE-opt* is an order of magnitude slower than *S-CoViNE-opt* due to having more disjointness constraints and variables than *S-CoViNE-opt*. Finally, Fig. 3(k) and Fig. 3(l) reveal that VN size has a more profound impact on the scalability of the ILP based approaches than what SN size has. For instance, with our current hardware, *CoViNE-opt* and *CoViNE-ILP* hit a ceiling in terms of VN size of 8 and 22 nodes, respectively, on a 100 node SN.

d) Optimality Analysis of Alg. 2: We now analyze how much extra resources are allocated by *S-CoViNE-fast* compared to *S-CoViNE-ILP* for different VN topologies. This extra resource usage is measured as the ratio of their embedding costs since both *S-CoViNE-fast* and *S-CoViNE-ILP* use the same Alg. 1 for augmentation and conflicting set computation. Therefore, the cost ratio indicates the perfomance comparison between ILP formulation for VN embedding and Alg. 2. Fig. 4(a) presents the Cumulative Distribution Function (CDF) of cost ratio for different types of VNs. This plot shows that 95% VNs with star and random topologies are embedded by *S-CoViNE-fast* with at most 20% extra resources compared to *S-CoViNE-ILP*. For ring topologies, costs of 91% of the VNs incurred by *S-CoViNE-fast* remain within 35% of those of *S-CoViNE-ILP*. Ring topologies are different from star and random topologies as all the VLinks in the ring need to be embedded on mutually disjoint substrate paths to ensure connectivity against failures. Such stringent disjointness constraints affect the sequential VLink mapping of Alg. 2 in *S-CoViNE-fast*, resulting in the highest cost ratio for ring topologies. However, the higher costs of *S-CoViNE-fast*, compared to *S-CoViNE-ILP*, are compensated by their higher scalability and much faster execution time as discussed earlier.

Fig. 4(a) exhibits higher cost ratios between *S-CoViNE-fast* and *S-CoViNE-ILP* than those observed in Fig. 3(a). This is due to using real SN topologies that are sparser than synthetic SNs used in Fig. 3(a). To analyze this further, Fig. 4(b) shows how the density of the underlying SN impacts the performance of *S-CoViNE-fast*. As we observe in Fig. 4(b), the costs of *S-CoViNE-fast* and *S-CoViNE-ILP* for ring and random VNs differ by a larger margin in a ring-like SN

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications



(a) CDF of Cost Ratio (b) Impact of SN connectivity

Fig. 4. Performance Analysis of Alg. 2

(*i.e.*, LNR=1.2). In this extreme case, the penalty for missing an optimal solution by *S-CoViNE-fast* is substantial as the disjoint path becomes much longer than the shorter one due to lack of SN path diversity, resulting in a higher cost ratio. However, cost ratios between *S-CoViNE-fast* and *S-CoViNE-ILP* decrease initially with the increase in SN LNR (Fig. 4(b)), and do not change significantly for SNs with LNR$\geq$1.6. An increase in SN LNR increases SN path diversity. *S-CoViNE-fast* exploits this path diversity to find a shorter path for embedding a VLink, resulting in lower cost ratios. For star VNs, disjointness requirement is much less than denser VNs, yielding close to unit cost ratios in all cases.

2) Large Scale Scenarios:

a) Embedding Cost: Fig. 5(a) shows embedding costs of *S-CoViNE-fast* and *D-CoViNE-fast* with varying VN sizes on 500 and 1000 node SNs. As expected, cost increases with the increase in both VN size and SN size. Fig. 5(b) and Fig. 5(c) show embedding cost by varying VN and SN LNRs, respectively. In these scenarios, embedding cost is mostly influenced by disjointness constraint and parallel VLink augmentation. For *D-CoViNE-fast*, augmentation cost dominates for VNs with LNR $\leq$ 2.1 (see Fig. 6(d)). In addition, the number of augmented VLinks decreases with the increase in VN LNR, hence the initial decrease in embedding cost. However for VNs with LNR $>$ 2.1, cost for ensuring disjointness constraint dominates, which justifies the later increase in Fig. 5(b) for *D-CoViNE-fast*. On the other hand, for *S-CoViNE-fast*, disjointness constraint dominates and embedding cost increases as higher number of VLinks are embedded on the same SN for VNs with larger LNR. For the same reason discussed for small cases, higher path diversity accounts for the decrease in cost with an increase in SN LNR in Fig. 5(c).

b) Scalability Analysis: Confirming to the running time analysis in § VI-A and § VI-C, the execution times for *S-CoViNE-fast* and *D-CoViNE-fast* increase with the increase in VN and SN sizes (Fig. 5(d)), VN LNR (Fig. 5(e)), and SN LNR (Fig. 5(f)). In addition, the execution times for *S-CoViNE-fast* and *D-CoViNE-fast* are comparable with each other in the large scale scenarios.

E. Failure Restoration

To demonstrate the failure restoration capability of CoViNE, we steer three classes of traffic in a VN, labeled as Pr-1 (highest priority), Pr-2, and Pr-3 (lowest priority) demanding 20%, 30%, and 50% of each VLink's bandwidth, respectively. A controller handles failures by rerouting traffic in the affected VLinks along alternate shortest paths in the embedding of the VN. Bandwidth sharing along these paths follows fair sharing policy between traffic from the same class and weighted fair sharing across different traffic classes.

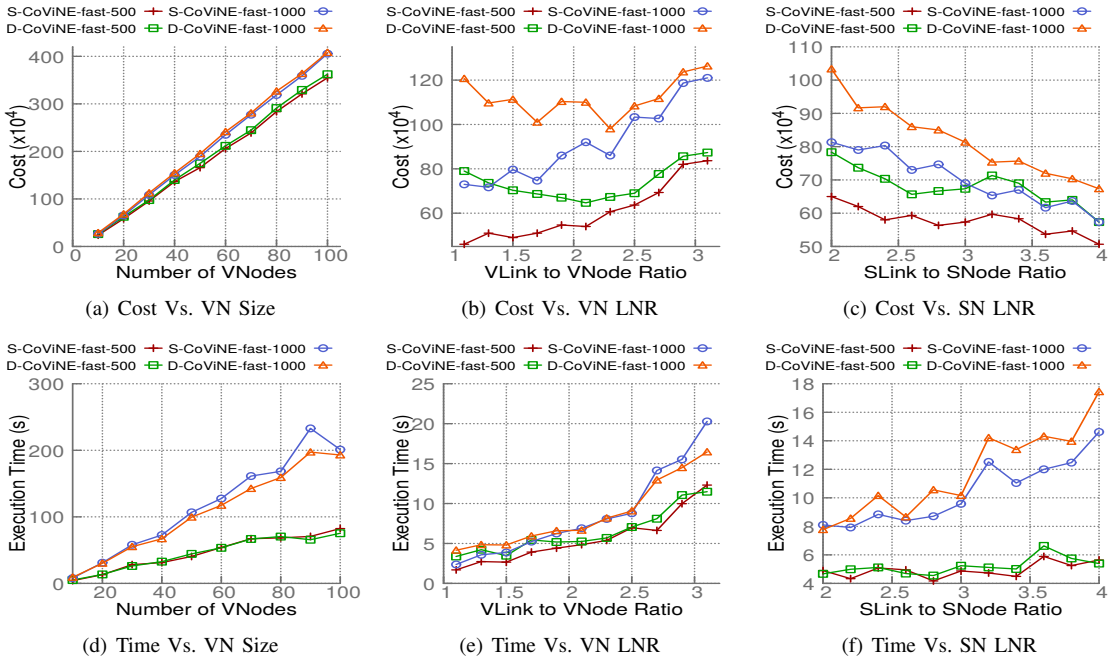
Fig. 6(a) and Fig. 6(b) show the restored bandwidth of *CoViNE-fast* and ViNE-ILP in the presence of single and double SLink failures, respectively. On the other hand, Fig. 6(c) and Fig. 6(d) present the overhead for ensuring connectivity in terms of embedding cost and number of augmented VLinks, respectively. As envisioned at the beginning of this paper, *CoViNE-fast* successfully restores almost the full bandwidth for the highest priority traffic in the presence of both single and double SLink failures as shown in Fig. 6(a) and Fig. 6(b), respectively. However, the successful restoration of the highest priority traffic achieved by *CoViNE-fast* comes at the expense of penalizing the traffic from lower priority classes. The overall decrease in restored bandwidth for *CoViNE-fast* with increasing VN LNR is counter-intuitive. This can be explained by observing the overhead shown in Fig. 6(d). As VN LNR increases in Fig. 6(d), the number of augmented VLinks and the amount of spare bandwidth decrease, thus offering less bandwidth to restore the same or possibly higher amount of bandwidth lost due to failures. Allocating spare bandwidth in the VLinks of a denser VN could help restore more bandwidth upon failure. However, spare bandwidth allocation with guaranteed connectivity is a separate problem that is out of the scope of this paper.

Fig. 6(a) and Fig. 6(b) also demonstrate that restored bandwidths of a specific traffic class achieved by *ViNE-ILP* are always lower than those achieved by both *S-CoViNE-fast* and *D-CoViNE-fast*. This is expected since *ViNE-ILP* does not take any measure to guarantee connectivity in the VN embedding against SLink failures. In addition, restored bandwidth of *ViNE-ILP* with the increase in VN LNR follows an opposite (increasing) trend compared to *CoViNE-fast*. The reasons are twofold. First, a higher LNR induces a higher path diversity in a VN that reduces the chances of VN partitioning in the presence of SLink failures. Second, a denser VN has more options for rerouting traffic in the affected VLinks along alternate paths. Despite the increasing trend, restored bandwidths of *ViNE-ILP* do not exceed those of *CoViNE-fast* even in higher LNR VNs, thanks to the disjointeness constraints of *CoViNE-fast*. Furthermore, restored bandwidths of *ViNE-ILP* are very poor in sparse VNs leaving the VNs vulnerable to failures, whereas *CoViNE-fast* offers much better restorability in such VNs.

VIII. Conclusion and Future Work

In this paper, we have studied the **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (CoViNE) problem that ensures VN connectivity in the presence of multiple substrate link failures. We have presented an ILP formulation, *CoViNE-opt*, that jointly solves three sub-problems of *CoViNE*, namely, VN augmentation, computation of disjointness constraints, and VN

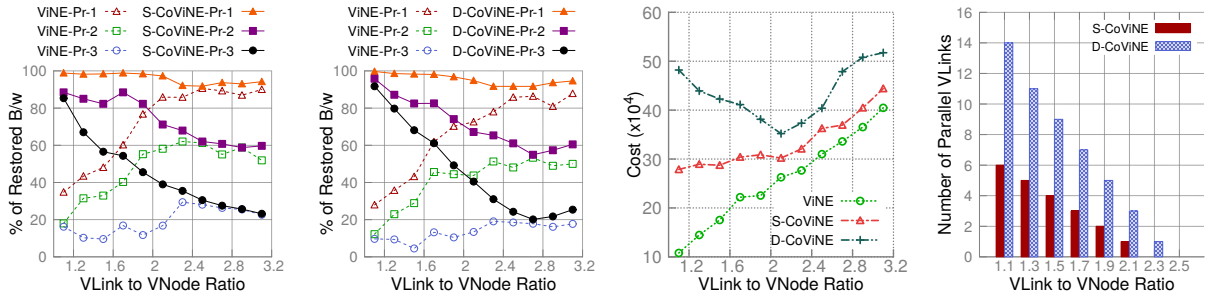
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2019.2954410, IEEE Transactions on Communications



(a) Cost Vs. VN Size (b) Cost Vs. VN LNR (c) Cost Vs. SN LNR

(d) Time Vs. VN Size (e) Time Vs. VN LNR (f) Time Vs. SN LNR

Fig. 5. Embedding cost and Execution Time Analysis for Large Scale Topologies



(a) Single Failure (b) Double Failure (c) Embedding Cost Overhead (d) VLink Aug. Overhead

Fig. 6. Restored Bandwidth and Overhead Analysis

embedding. To address the intractability of *CoViNE-opt*, we have separated VN augmentation and disjointness constraint computation from embedding and addressed them sequentially. We have introduced the concept of conflicting set of a VN to efficiently compute disjointness constraints without relying on the exponential number of edge-cuts in a VN as required by *CoViNE-opt*. Given the NP-complete nature of each of the sub-problems and their inter-dependency, we have presented a heuristic algorithm that solves both VN augmentation and conflicting set computation simultaneously. The conflicting set as well as the augmented VN, both produced by the heuristic algorithm, are then used to impose polynomial number of disjointness constraints to the sub-problem of VN embedding. Based on how we address the constrained VN embedding sub-problem, we have provided two more solutions to *CoViNE*, namely *CoViNE-ILP* and *CoViNE-fast*. *CoViNE-ILP* extends a *Multi-commodity Unsplittable Flow* based ILP formulation to address the constrained VN embedding, while *CoViNE-fast* uses a heuristic algorithm to do the same. In contrast to the state-of-the-art, our solutions are generalized to handle multiple substrate link failures for arbitrary topologies.

We have evaluated our solutions using a variety of network topologies under different failure scenarios. Evaluation results reveal that although *CoViNE-opt* can be used to benchmark the solutions to CoViNE, it cannot be used to solve practical problems due to its severely low scalability. Our evaluation results demonstrate that *CoViNE-ILP* very closely approximates *CoViNE-opt*, and can be used as a baseline to compare heuristic algorithms. In contrast, *CoViNE-fast* scales to large topologies at the cost of provisioning about 16% additional resources compared to *CoViNE-ILP*, while executing several orders of magnitude faster for the same problem instances. Evaluation results also show that *CoViNE* for single and double link failure scenarios require on average $\sim 24\%$ and $\sim 66\%$ extra resources than a VN embedding strategy that does not guarantee any connectivity. We have also demonstrated that VN connectivity can be leveraged to restore higher priority traffic in the presence of multiple substrate link failures.

We believe that *CoViNE* can set the stage for further research investigations. We intend to investigate the problem

of ensuring different levels of connectivity for the VLinks in a VN, which can empower a VN-operator to offer a wide variety of Service Level Agreements to its customers. We also want to extend our current solutions by considering SLinks' spare bandwidth allocation, SNodes' throughput, and substrate path length constraints in a coordinated manner.

REFERENCES

[1] H. Zhang *et al.*, "Network slicing based 5g and future mobile networks: mobility, resource management, and challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[2] X. Foukas *et al.*, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[3] N. M. M. K. Chowdhury *et al.*, "A Survey of Network Virtualization," *Computer Networks*, Apr 2010.

[4] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM*, 2006.

[5] B. Jaumard and H. A. Hoang, "Design and dimensioning of logical survivable topologies against multiple failures," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, no. 1, pp. 23–36, 2013.

[6] M. R. Rahman *et al.*, "Svne: Survivable virtual network embedding algorithms for network virtualization," *IEEE Trans. on Netw. and Service Management*, vol. 10, pp. 105–118, 2013.

[7] A. Markopoulou *et al.*, "Characterization of Failures in an IP Backbone," in *INFOCOM*, Mar 2004.

[8] S. Herker *et al.*, "Survey on Survivable Virtual Network Embedding Problem and Solutions," in *ICNS*, 2013.

[9] D. Stamatelakis and W. D. Grover, "Ip layer restoration and network planning based on virtual protection cycles," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1938–1949, Sep 2006.

[10] A. E. Kamal *et al.*, "Overlay protection against link failures using network coding," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1071–1084, Aug 2011.

[11] A. Haider and R. J. Harris, "Recovery techniques in next generation networks." *IEEE Communications Surveys and Tutorials*, vol. 9, no. 1-4, pp. 2–17, 2007.

[12] T. Lin *et al.*, "Unified mathematical programming frameworks for survivable logical topology routing in ip-over-wdm optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 6, no. 2, pp. 190–203, 2014.

[13] M. Kurant and P. Thiran, "On survivable routing of mesh topologies in ip-over-wdm networks," in *IEEE INFOCOM*, 2005, pp. 1106–1116.

[14] K. Thulasiraman *et al.*, "Logical topology augmentation for guaranteed survivability under multiple failures in ip-over-wdm optical networks," *Optical Switching and Networking*, vol. 7, no. 4, pp. 206–214, 2010.

[15] Z. Zhou *et al.*, "Novel survivable logical topology routing by logical protecting spanning trees in ip-over-wdm networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1673–1685, 2017.

[16] Z. Zhou *et al.*, "Survivable cloud network design against multiple failures through protecting spanning trees," *Journal of Lightwave Technology*, vol. 35, no. 2, pp. 288–298, 2017.

[17] M. Kurant *et al.*, "Survivable mapping algorithm by ring trimming (smart) for large ip-over-wdm networks," in *BroadNets*, Oct 2004, pp. 44–53.

[18] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Citeseer, 1996.

[19] T. Watanabe and A. Nakamura, "A minimum 3-connectivity augmentation of a graph," *Journal of Computer and System Sciences*, vol. 46, no. 1, pp. 91–128, 1993.

[20] E. Modiano *et al.*, "Survivable lightpath routing: a new approach to the design of wdm-based networks," *IEEE JSAC*, 2002.

[21] K. Thulasiraman *et al.*, "Circuits/cutsets duality and a unified algorithmic framework for survivable logical topology design in ip-over-wdm optical networks," in *IEEE INFOCOM*, Apr 2009, pp. 1026–1034.

[22] A. Todimala *et al.*, "A scalable approach for survivable virtual topology routing in optical wdm networks," *IEEE JSAC*, vol. 25, no. 6, pp. 63–69, 2007.

[23] N. Shahriar *et al.*, "Connectivity-aware virtual network embedding," in *IFIP Networking Conference 2016*, 2016, pp. 46–54.

[24] P. Gill *et al.*, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *ACM SIGCOMM*, Aug 2011.

[25] T. Guo *et al.*, "Shared backup network provision for virtual network embedding," in *IEEE ICC*, 2011.

[26] J. Xu *et al.*, "Survivable virtual infrastructure mapping in virtualized data centers," in *IEEE CLOUD*, 2012.

[27] M. M. A. Khan *et al.*, "Multi-path link embedding for survivability in virtual networks." *IEEE Trans. Network and Service Management*, vol. 13, no. 2, pp. 253–266, 2016.

[28] N. Shahriar *et al.*, "Virtual network survivability through joint spare capacity allocation and others," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 502–518, 2018.

[29] H. Yu *et al.*, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *IEEE GLOBECOM*, 2010, pp. 1–6.

[30] H. Yu *et al.*, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *IEEE ICC*, 2011, pp. 1–6.

[31] X. Liu *et al.*, "Disaster-prediction based virtual network mapping against multiple regional failures," in *IFIP/IEEE IM*, 2015, pp. 371–378.

[32] S. R. Chowdhury *et al.*, "Dedicated protection for survivable virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 913–926, 2016.

[33] M. A. Soares and E. R. Madeira, "A multi-agent architecture for autonomic management of virtual networks," in *IEEE/IFIP NOMS*, 2012, pp. 1183–1186.

[34] Q. Chen *et al.*, "A survivable virtual network embedding scheme based on load balancing and reconfiguration," in *IEEE/IFIP NOMS*, 2014, pp. 1–7.

[35] M. Pourvali *et al.*, "Progressive recovery for network virtualization after large-scale disasters," in *IEEE ICNC*, 2016, pp. 1–5.

[36] N. Shahriar *et al.*, "Generalized recovery from node failure in virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 261–274, 2017.

[37] I. Houidi *et al.*, "Adaptive virtual network provisioning," in *ACM SIGCOMM VISA workshop*, 2010, pp. 41–48.

[38] Q. Zhu *et al.*, "A hybrid reliable heuristic mapping method based on survivable virtual networks for network virtualization," *Discrete Dynamics in Nature and Society*, vol. 2015, 2015.

[39] A. Hmaity *et al.*, "Survivable virtual network mapping to provide content connectivity against double-link failures," in *IEEE DRCN*, 2016, pp. 160–166.

[40] K. Lee *et al.*, "Cross-layer survivability in wdm-based networks," *IEEE/ACM Trans. on Netw.*, vol. 19, no. 4, pp. 1000–1013, Aug 2011.

[41] Z. Zhou *et al.*, "Cross-layer network survivability under multiple cross-layer metrics," *IEEE/OSA J. of Optical Comm. & Net.*, 2015.

[42] K. Thulasiraman *et al.*, "Primal meets dual: A generalized theory of logical topology survivability in ip-over-wdm optical networks," in *IEEE COMSNETS*. IEEE, 2010, pp. 1–10.

[43] M. Javed *et al.*, "Lightpaths routing for single link failure survivability in ip-over-wdm networks," *Journal of Communications and Networks*, vol. 9, no. 4, pp. 394–401, 2007.

[44] P. Raghavan and C. D. Tompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[45] C. Liu *et al.*, "A new survivable mapping problem in ip-over-wdm networks," *IEEE JSAC*, vol. 25, no. 3, pp. 25–34, Apr 2007.

[46] M. Kurant *et al.*, "Survivable routing of mesh topologies in ip-over-wdm networks by recursive graph contraction," *JSAC*, vol. 25, no. 5, pp. 922–933, 2007.

[47] S. Even *et al.*, "On the complexity of time table and multi-commodity flow problems," in *IEEE FOCS*, 1975, pp. 184–193.

[48] A. Anagnostopoulos *et al.*, "A mazing 2+ ε approximation for unsplittable flow on a path," in *ACM SODA*, 2014, pp. 26–41.

[49] Z. Friggstad *et al.*, "On linear programming relaxations for unsplittable flow in trees," in *LIPIcs-Leibniz Int. Proc. in Informatics*, vol. 40, 2015.

[50] P. Bonsma *et al.*, "A constant-factor approximation algorithm for unsplittable flow on paths," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 767–799, 2014.

[51] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *Proc. of IFIP Networking*, 2018, pp. 55–63.

[52] D. D.-J. Kan *et al.*, "Lightpath routing and capacity assignment for survivable ip-over-wdm networks," in *IEEE DRCN*, 2009.

[53] Y. Liu *et al.*, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. on Netw.*, vol. 13, no. 1, pp. 198–211, 2005.

[54] Y. Liu *et al.*, "Spare capacity allocation in two-layer networks," *IEEE JSAC*, vol. 25, no. 5, pp. 974–986, 2007.

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

[56] N. Christofides, "An algorithm for the chromatic number of a graph," *The Computer Journal*, vol. 14, no. 1, pp. 38–39, 1971.

[57] Menger's theorem [online] http://math.fau.edu/locke/menger.htm.
[58] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[59] D. A. Dunn, W. D. Grover, and M. H. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, pp. 88–99, Jan 1994.
[60] A. Fischer *et al.*, "Virtual Network Embedding: A Survey," *IEEE CST*, Feb 2013.
[61] M. Baghel *et al.*, "Survey of metaheuristic algorithms for combinatorial optimization," *Int. J. of Computer Applications*, vol. 58, no. 19, 2012.
[62] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking (ToN)*, vol. 12, no. 1, pp. 2–16, 2004.
[63] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.
[64] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

Nashid Shahriar (S'16) is a Ph.D. candidate at the School of Computer Science, University of Waterloo. He received M.Sc. and B.Sc. degrees in computer science and engineering from Bangladesh University of Engineering and Technology in 2011 and 2009, respectively. He is a recipient of Ontario Graduate Scholarship, President's Graduate Scholarship, and David R. Cheriton Graduate Scholarship with the University of Waterloo. He is a co-recipient of the Best Paper Award in the IEEE/ACM/IFIP CNSM 2017 and the IEEE/ACM/IFIP CNSM 2017 and the Best Student Paper Award in the IEEE NetSoft 2019. His research interests include network virtualization, 5G network slicing, and network reliability.

Reaz Ahmed received his Ph.D. degree in computer science from the University of Waterloo, in 2007. He received M.Sc. and BSc. degrees in computer science from Bangladesh University of Engineering and Technology in 2002 and 2000, respectively. He received the IEEE Fred W. Ellersick award in 2008. 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.

Shihabur Rahman Chowdhury (S'13) is a PhD candidate at the David R. Cheriton School of Computer Science, University of Waterloo. He received his B.Sc. degree in computer science and engineering from BUET in 2009. His research interests include virtualization and softwarization of computer networks. He is co-recipient of the Best Paper Award at the IEEE/ACM/IFIP CNSM 2019, IEEE NetSoft 2019, and the IEEE/ACM/IFIP CNSM 2017 conference. He also received several other recognitions, including a MITACS Globalink Research Award, Ontario Graduate Scholarship and President's Graduate Scholarship at the University of Waterloo, among others.

Md Mashrur Alam Khan is currently working as a Software Engineer at Ceridian Canada. He received the Masters of Mathematics degree and Bachelor of Science in Computer Science and Engineering from the University of Waterloo, Canada and Bangladesh University of Engineering and Technology, Bangladesh in 2015 and 2012, respectively. His research interest include Network Virtualization and Internet of things.

Raouf Boutaba (F'12) received the M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a professor of computer science and university research chair at the University of Waterloo. He is the founding editor in chief of the IEEE Transactions on Network and Service Management (2007–2010) and on the editorial boards of other journals. He received several best paper awards and recognitions including the Premier's Research Excellence Award, the IEEE ComSoc Hal Sobol, Fred W. Ellersick, Joe LociCero, Dan Stokesbury, Salah Aidarous Awards, and the IEEE Canada McNaughton Gold Medal. He is a fellow of the Royal Society of Canada, the Institute of Electrical and Electronics Engineers (IEEE), the Engineering Institute of Canada, and the Canadian Academy of Engineering. His research interests include resource and service management in networks and distributed systems.

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 received 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 IEEE/ACM/IFIP CNSM 2017 and 2019.

PLACE PHOTO HERE

Feng Zeng received the master degree from the University of Electronics Science and Technology of China, majoring in communication and information. He worked in high performance Router PDU of H3C. In 2008, he joined Huawei and mainly focused on network plan, evaluation and optimization algorithm.