Network Planning with Bilevel Optimization to Address Attacks to Physical Infrastructure of SDN

Qian Lv[1], Jing Zhu[1], Fen Zhou[2], Zuqing Zhu[1,†]
[1]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[2]LISITE lab, Institut Superieur d'Electronique de Paris, France
[†]Email: {zqzhu}@ieee.org

***Abstract*—It is known that the design of the control plane (CP) is vital in the network planning for software-defined networking (SDN), while to improve throughput and enlarge geographical coverage, optical networks are commonly used as the physical infrastructure of SDN. However, physical-layer attacks can disrupt the operation of an optical network and thus complicate the CP design. In this work, we consider planned physical-layer attacks when designing the CP of an SDN that uses an optical network as its physical infrastructure. We first show that the CP design problem should be modeled as a bilevel optimization, where the network planner designs the CP with the minimum vulnerability to physical-layer attacks (*i.e.*, the upper-level optimization), while the attacker plans and launches attacks to disrupt the designed CP (*i.e.*, the lower-level optimization). Then, the bilevel mode is transformed into a mixed integer linear programming (MILP) model, which can solve the CP design problem exactly. We also propose a heuristic to tackle the problem time-efficiently.**

***Index Terms*—Software-defined networking (SDN), Control plane design, Physical-layer attacks, Bilevel optimization.**

I. Introduction

Over the past decade, fast-developing network services created many new demands on network control and management (NC&M) [1]. Fortunately, software-defined networking (SDN) has demonstrated its great potential in reforming the NC&M of various networks. However, the separation of control and data planes (CP and DP) in SDN complicates network planning, and thus makes it more challenging. This is because in addition to the DP, a network planner also needs to design the CP under stringent quality-of-service (QoS) constraints [2]. Specifically, the problem of CP design usually involves: 1) where to place the controller(s), 2) how to assign the switches in DP to the controller(s), and 3) how to route the control channels in DP to bridge the communications between switches and controller(s). Previously, by considering control channel latency, CP availability, and load-balancing in optimization objectives, people have tackled the whole (or partial) problem of CP design in [2–6]. Nevertheless, none of these studies have accounted the planned attacks to the physical infrastructure of SDN.

When it comes to support ultra-high throughput and/or cover a relatively large geographical area, an optical network would be the only feasible physical infrastructure of SDN [7, 8]. However, it is known that optical networks are vulnerable to various physical-layer attacks [9, 10], which can be leveraged to disrupt the operation of DP and CP in an SDN. Moreover, recent advances in flexible-grid elastic optical networks [11–13] suggest that future optical networks could use much narrower guard-bands, *i.e.*, certain physical-layer attacks (*e.g.*, those based on inter-channel crosstalk) can be launched more easily [14]. Note that, such attacks would be more devastating to an SDN if they disturb the CP [3]. More importantly, a malicious party can plan its attacks intelligently to boost their efficiency and aggravate their impacts. Hence, the existing CP design schemes that try to address random failures cannot guarantee sufficient robustness to the planned attacks.

After an attack, even though the CP can be re-established with backup components [15, 16], service disruptions cannot be avoided. Hence, it is desired to address the attacks in the phase of network planning, *i.e.*, solving the problem of CP design in consideration of planned physical-layer attacks such that the resulting disruptions can be minimized. In [17], Zhu *et al.* developed a game theoretic approach to address this problem. Nevertheless, they used some impractical assumptions, such as the physical-layer attacks can only be fiber cuts, and the attacker only has very limited strategies to disturb the CP.

Note that, the CP design taking planned physical-layer attacks into account can be better modeled with bilevel optimization [18]. More specifically, the network planner's task is the upper-level optimization, which is to design a CP that can minimize the disruption when a planned attack happens, while the attacker's task is the lower-level optimization, which is to plan and launch attacks to disturb the designed CP. In this work, we formulate this bilevel optimization, and propose two approaches to solve it. We first leverage the Karush-Kuhn-Tucker (KKT) condition [19] to transform the bilevel optimization into a normal mixed integer linear programming (MILP) model, which gives exact solutions to the original CP design problem. Then, to improve time-efficiency, we propose a tree-based heuristic. Simulation results indicate that with the bilevel optimization, our proposals can effectively reduce the disruptions on CP due to planned physical-layer attacks.

The rest of the paper is organized as follows. Section II formulates the bilevel optimization for CP design. Our proposals to solve the bilevel optimization are presented in Section III. Section IV discusses the simulations for performance evaluation. Finally, we summarize the paper in Section V.

II. CP Design Considering Physical-layer Attacks

A. Problem Description

We model the physical infrastructure of an SDN as a graph $G(V, E)$, which represents an optical network with V and

E as its sets of switch nodes and fiber links, respectively. We assume that the CP and DP of the SDN share the same physical infrastructure, which is the common case in metro or backbone networks for cost reduction [2]. The DP can use all the nodes and links in $G(V, E)$. The network planner needs to find a subgraph of $G(V, E)$ to accomplish the CP design, while taking planned physical-layer attacks into consideration.

Defintion 1: The **Problem of CP Design** is to find

- How to place a fixed number of controllers on the nodes.
- How to assign the switches in DP to the controllers.
- How to select a subset of fiber links (*i.e.*, $E' \subset E$) on which control channels can be routed.

In this work, the control channels refer to those that bridge the communications either between a controller and its switches or between two controllers. We assume that each switch has one bi-directional control channel to talk with its controller, while there is also one bi-directional control channels between each controller pair (if there are more than one controllers). When the network planning has been done, the CP design will be known by the attacker, which can then launch physical-layer attacks from any node in V to disturb the operation of CP. More specifically, the attacker leverages the CP design to calculate the routing paths of all the control channels (*i.e.*, by applying the routing algorithm used by the CP to $G'(V, E')$), and then launches an attack hoping to cause the maximized disruption to the operation of CP.

Note that, as the control latency is one of the most important QoS parameters of CP [3], this work assumes that the CP routes control channels with the shortest routing paths. According to the studies in [9, 14], the impact of a physical-layer attack to an optical network can be quantified with the number of fiber links shared by the malicious and legitimated lightpaths. This is because the more fiber links shared by the two lightpaths, the larger signal degradation can be caused on the legitimated one when the malicious one leverages common physical-layer attacking scenarios (*e.g.*, interference injection and power jamming). Then, we can see that if the CP design results in more control channels share a same link, the attacker would have a better chance to launch a more effective attack.

Defintion 2: In a designed CP, the most vulnerable path is the control channel that shares link(s) with most other control channels. Hence, the **Vulnerability of a CP Design** (*i.e.*, $\mathcal{V}_{\max}$) is defined as one plus the number of control channels that share links with the one with the most vulnerable path.

Fig. 1 shows an illustrative example on the CP design considered in this work and how to calculate a CP design's vulnerability. It can be seen that in the designed CP, the most vulnerable path is 3-4-5, which shares links with two other control channels. Therefore, if the attacker plans its attack with the path 3-4-5, it can impact the most control channels (*i.e.*, three). This explains the rationale behind our definition of the vulnerability of a CP design. The example also clarifies why the CP design taking planned physical-layer attacks into account should be modeled with bilevel optimization.

Specifically, the problem involves two independent entities, *i.e.*, the network planner and the attacker. The network planner performs the upper-level optimization to design a CP whose vulnerability should be minimized to protect the SDN from planned attacks, while the attack's task is the lower-level optimization to find the most vulnerable path in the designed CP to maximize the impact of its attack. In other words, different CP designs from the network planner make the attacker select different paths for amplifying the effectiveness of its attack, while the paths considered by the attacker would in turn affect the network planner's CP design.

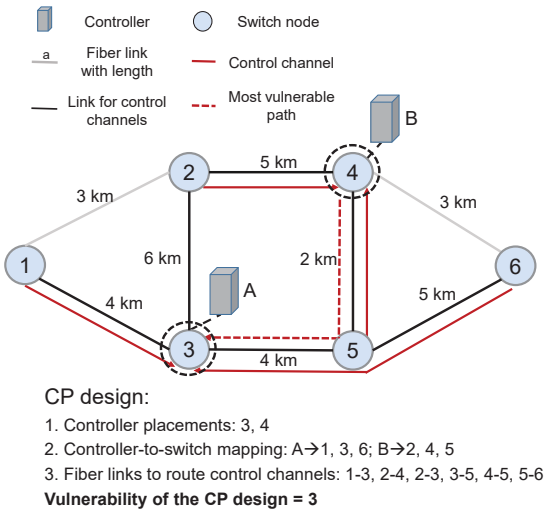

Fig. 1. Example on CP design and its vulnerability.

B. Formulation of Bilevel Optimization

We model the problem of CP design as the following bilevel optimization. The common parameters of both levels are

Common Parameters:

- $G(V, E)$: the topology of the underlying optical network.
- $L_{(u,v)}$: the length of link $(u, v) \in E$.
- $\mathcal{N}_c$: the number of controllers to be allocated in the SDN.
- $\mathcal{C}$: a controller's capacity in number of managed switches.

The **Upper-level Optimization** is for the network planner to design the CP with minimized vulnerability.

Parameters:

- $y^{z,w}_{(u,v)}$: the boolean indicator that equals 1 if link $(u, v) \in E$ will be used for the control channel between nodes z and w, and 0 otherwise.

Variables:

- $x_{(u,v)}$: the boolean variable that equals 1 if link (u, v) is selected in the CP design, and 0 otherwise.
- c_u: the boolean variable that equals 1 if a controller is placed on node $u \in V$, and 0 otherwise.
- $k_{u,v}$: the boolean variable that equals 1 if the switch at node u is assigned to the controller placed at node v, and 0 otherwise.
- $b_{u,v}$: the boolean variable that equals 1 if there is one control channel that uses the shortest path for $u \rightarrow v$, and 0 otherwise.
- $o_{u,v}$: the boolean variable that equals 1 if both nodes u and v have controller placements, and 0 otherwise.

- $s_{(u,v)}^{z,w,p,q}$: the boolean variable that equals 1 if the shortest paths for $z \to w$ and $p \to q$ share (u, v), and 0 otherwise.
- $S^{z,w,p,q}$: the boolean variable that equals 1 if the shortest paths for $z \to w$ and $p \to q$ have shared link(s), and 0 otherwise.
- $n^{u,v}$: the integer variable that indicates the number of shortest paths sharing link(s) with the one for $u \to v$.
- $\mathcal{V}_{\max}$: the integer variable that indicates the vulnerability of the CP design.
- $d_{u,v}$, $t_{u,v}$, $w_{u,v}$, and $r_{u,v}$: the auxiliary boolean variables introduced to linearize constraints.

Objective:

The objective of the upper-level optimization is to minimize the vulnerability of the CP design. This optimization is related to the lower-level one through $\{y_{(u,v)}^{z,w}\}$, which are parameters here but get optimized in the lower-level optimization.

$$\text{Minimize} \quad \mathcal{V}_{\max}. \tag{1}$$

Constraints:

$$x_{(u,v)} = x_{(v,u)}, \quad \forall (u,v) \in E. \tag{2}$$

Eq. (2) ensures that the bidirectional links are selected to build duplex control channels.

$$\sum_{u \in V} c_u = \mathcal{N}_c. \tag{3}$$

Eq. (3) ensures that enough controllers are placed in the CP.

$$k_{u,v} \le c_v, \quad \forall u, v \in V, \tag{4}$$

$$\sum_{v \in V} k_{u,v} = 1, \quad \forall u \in V, \tag{5}$$

$$\sum_{u \in V} k_{u,v} \le \mathcal{C}, \quad \forall v \in V. \tag{6}$$

Eqs. (4)-(6) determine the mapping between controllers and switches.

$$\begin{cases} y_{(u,v)}^{z,w} + y_{(u,v)}^{p,q} - 1 \le s_{(u,v)}^{z,w,p,q}, \\ y_{(u,v)}^{z,w} \ge s_{(u,v)}^{z,w,p,q}, \\ y_{(u,v)}^{p,q} \ge s_{(u,v)}^{z,w,p,q}, \end{cases} \tag{7}$$
$$\forall (u,v) \in E, \ \{z, w, p, q \in V : z \ne w, p \ne q\},$$

$$s_{(u,v)}^{z,w,p,q} \le S^{z,w,p,q} \le \sum_{(u,v) \in E} s_{(u,v)}^{z,w,p,q}, \tag{8}$$
$$\forall (u,v) \in E, \ \{z, w, p, q \in V : z \ne w, p \ne q\},$$

$$n^{z,w} + 1 = \sum_{\{p,q \in V : p \ne q\}} S^{z,w,p,q}, \quad \{z, w \in V : z \ne w\}, \tag{9}$$

$$n^{u,v} + 1 \le \mathcal{V}_{\max}, \quad \{u, v \in V : u \ne v\}. \tag{10}$$

Eqs. (7)-(10) ensure that the vulnerability of the CP design (*i.e.*, $\mathcal{V}_{\max}$) is calculated correctly.

$$\begin{cases} b_{u,v} = k_{u,v} + k_{v,u} + o_{u,v} - d_{u,v} - t_{u,v} - w_{u,v} + r_{u,v}, \\ o_{u,v} \le c_u, \quad o_{u,v} \le c_v, \quad o_{u,v} \ge c_v + c_u - 1, \\ d_{u,v} \le k_{u,v}, \quad d_{u,v} \le k_{v,u}, \quad d_{u,v} \ge k_{u,v} + k_{v,u} - 1, \\ t_{u,v} \le k_{u,v}, \quad t_{u,v} \le o_{u,v}, \quad t_{u,v} \ge k_{u,v} + o_{u,v} - 1, \\ w_{u,v} \le k_{v,u} \quad d_{u,v} \le o_{u,v}, \quad w_{u,v} \ge k_{v,u} + o_{u,v} - 1, \\ r_{u,v} \le d_{u,v} \quad r_{u,v} \le o_{u,v}, \quad r_{u,v} \ge d_{u,v} + o_{u,v} - 1, \end{cases} \tag{11}$$
$$\{u, v \in V : u \ne v\}.$$

Eq. (11) determines the values of $\{b_{u,v}\}$ and $\{o_{u,v}\}$.

The **Lower-level Optimization** is for the attacker to find the most effective attacking path based on the designed CP. Hence, the solution from the upper-level optimization is the precondition, which means that the variables $\{x_{(u,v)}\}$ and $\{b_{u,v}\}$ have become parameters.

Variables:

- $y_{(u,v)}^{z,w}$: the boolean variable that equals 1 if link $(u, v) \in E$ will be used for the control channel between nodes z and w, and 0 otherwise.

Objective:

The objective of the lower-level optimization is to find the shortest paths in the designed CP to launch attacks.

$$\text{Minimize} \quad \sum_{\{z,w \in V : z \ne w\}} \sum_{(u,v) \in E} L_{(u,v)} \cdot y_{(u,v)}^{z,w}. \tag{12}$$

Constraints:

$$\sum_{(u,v) \in E} y_{(u,v)}^{z,w} - \sum_{(v,u) \in E} y_{(v,u)}^{z,w} = \begin{cases} b_{z,w}, & u = z, \\ -b_{z,w}, & u = w, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$
$$\{z, w \in V : z \ne w\}.$$

Eq. (13) ensures the flow conservation conditions.

$$y_{(u,v)}^{z,w} \le b_{z,w}, \quad \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}. \tag{14}$$

Eq. (14) ensures that $y_{(u,v)}^{z,w}$ identifies a link used by control channel(s).

$$y_{(u,v)}^{z,w} = y_{(v,u)}^{w,z}, \quad \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}. \tag{15}$$

Eq. (15) ensures that the bi-directional control channels of a switch-controller pair use the same path.

$$y_{(u,v)}^{z,w} \le x_{(u,v)}, \quad \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}. \tag{16}$$

Eq. (16) ensures that the shortest paths for control channels can only use the links selected in the CP design.

It is worth noting that if the number of controllers to be allocated is one, the control channels only involve those between the controller and all the switches. Otherwise, the CP design needs to set up control channels that support not only the communications between controller-switch pairs but also those between controller-controller pair(s).

III. Algorithm Design

A. MILP based on KKT Condition

The bilevel optimization in Section II-B can be solved by leveraging the Karush-Kuhn-Tucker (KKT) condition to represent the lower-level optimization and transform it into a standard MILP [19]. More specifically, since the lower-level optimization is solved based on the solution from the upper-level one, the decision variables of the lower-level optimization will be unimodular when the decision variables of the upper-level optimization have been determined. This enables us to represent the lower-level optimization with the KKT condition of its linear programming (LP) relaxation, which introduces real KKT multipliers $\{\gamma_{(u,v)}^{z,w}, \ \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}, \ \{\lambda_{(u,v)}^{z,w}, \ \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}\}$, $\{\nu_{(u,v)}^{z,w}, \ \forall (u,v) \in E, \ \{z, w \in V : z \ne w\}\}$, and $\{\mu_u^{z,w}, \ \forall u \in V, \ \{z, w \in V : z \ne w\}\}$. Then, we can obtain the optimal solution of the lower-level optimization by solving it with the constraints in Eqs. (13)-(16) and following ones.

$$y_{(u,v)}^{z,w} \ge 0, \quad \forall (u,v) \in E, \ \{z, w \in V : z \ne w\} \tag{17}$$

$$L_{(u,v)} - \mu_u^{z,w} + \mu_v^{z,w} - \gamma_{(u,v)}^{z,w} + \lambda_{(u,v)}^{z,w} + \nu_{(u,v)}^{z,w} = 0 \tag{18}$$
$$\forall (u,v) \in E, \ \{z, w \in V : z \ne w\}$$

$$\begin{cases} \gamma^{z,w}_{(u,v)} \cdot y^{z,w}_{(u,v)} = 0, \\ \lambda^{z,w}_{(u,v)} \cdot \left(y^{z,w}_{(u,v)} - x_{(u,v)} \right) = 0, \\ \nu^{z,w}_{(u,v)} \cdot \left(y^{z,w}_{(u,v)} - b_{z,w} \right) = 0, \\ \forall (u,v) \in E,\ \{z,w \in V : z \neq w\} \end{cases} \tag{19}$$

$$\begin{aligned} &\gamma^{z,w}_{(u,v)} \geq 0, \lambda^{z,w}_{(u,v)} \geq 0, \nu^{z,w}_{(u,v)} \geq 0, \\ &\forall (u,v) \in E,\ \{z,w \in V : z \neq w\}. \end{aligned} \tag{20}$$

Here, the nonlinear equations in Eq. (19) can be linearized by introducing a large positive number M to transform them into

$$\begin{aligned} &\nu^{z,w}_{(u,v)} \leq M \cdot \left[1 - \left(b_{z,w} - y^{z,w}_{(u,v)} \right) \right], \\ &\lambda^{z,w}_{(u,v)} \leq M \cdot \left[1 - \left(x_{(u,v)} - y^{z,w}_{(u,v)} \right) \right], \\ &\gamma^{z,w}_{(u,v)} \leq M \cdot \left(1 - y^{z,w}_{(u,v)} \right), \\ &\forall (u,v) \in E,\ \{z,w \in V : z \neq w\}, \end{aligned} \tag{21}$$

Finally, we obtain the following MILP model whose solution is also the optimal solution of the bilevel optimization.

$$\text{Minimize} \quad \mathcal{V}_{\max},$$
$$s.t. \quad \text{Eqs. (2)-(11), (13)-(16), (18) and (20)-(21).}$$

B. Heuristic Algorithm

The MILP model will become intractable for problems with relatively large sizes. Hence, we design a heuristic to solve the CP design time-efficiently. The heuristic consists of two phases. In the first phase, we get a fixed number of mappings (*i.e.*, $\mathcal{M}$) between controllers and switches as the candidates to be considered in the next phase. *Algorithm* 1 explains the procedure. *Lines* 1-5 are for the initialization. *Line* 1 preselects the $\mathcal{M}$ combinations, where the i-th one assigns the controllers to manage $\{k^i_1, \cdots, k^i_{\mathcal{N}_c}\}$ switches, respectively. We have

$$\begin{cases} k^i_j \leq \mathcal{C}, \quad \forall i \in [1, \mathcal{M}],\ j \in [1, N_c], \\ \sum_{j=1}^{\mathcal{N}_c} k^i_j = |V|, \quad \forall i \in [1, \mathcal{M}]. \end{cases}$$

Then, *Lines* 3-4 select $\mathcal{N}_c$ nodes whose degrees are the largest to place controllers. The outer for-loop (*Lines* 6-13) obtains $\mathcal{M}$ feasible controller-to-switch mappings according to the preselected combinations, and stores them in set $\mathbb{M}$. Specifically, as shown in the inner for-loop (*Lines* 8-11), for each combination, the j-th controller gets assigned to the k^i_j unassigned switches whose shortest paths to it are the shortest.

The second phase figures out the final CP design based on the mapping candidates stored in $\mathbb{M}$, by leveraging minimum spanning trees (MSTs), as explained in *Algorithm* 2. *Line* 1 calculates the MST in $G(V, E)$ to start with, which is the initial $\mathcal{T}_1$ (*Line* 4), and we initialize $\tilde{\mathcal{V}}_{\max}$ to its upper-bound and the final CP design $\mathbb{D}$ as an empty set in *Line* 2. Then, the outer for-loop (*Lines* 3-37) checks each mapping candidates to finally get the CP design that provides the smallest vulnerability. Specifically, the while-loop of *Lines* 7-20 tries to replace a link in $\mathcal{T}_1$ until the resulting vulnerability is minimized, and the while-loop of *Lines* 24-33 then tries to expand the obtained tree by adding shortest paths to it, until the total length of all the control channels is minimized. Through this process, we balance the tradeoff between the vulnerability and control latency of the CP design. Finally, in *Lines* 34-36, we select the CP design that has the smallest vulnerability to store in $\mathbb{D}$, which includes the controller-to-switch mapping and the subgraph for routing control channels (*i.e.*, $\mathcal{T}_1$).

The time complexity of the heuristic can be analyzed as follows. In *Algorithm* 1, the complexity to calculate the shortest paths between node pairs is $O(|V|^3)$, the complexity of the outer for-loop is $O(\mathcal{N}_c)$, and that of the inner for-loop is $O(\mathcal{M})$. Therefore, the overall time complexity of *Algorithm* 1 is $O(|V|^3 + \mathcal{M} \cdot \mathcal{N}_c)$. In *Algorithm* 2, the complexity to obtain the MST $\mathcal{T}_1$ is $O(|E| \cdot \log(|V|))$. The outer for-loop runs for $\mathcal{M}$ times, the while-loop of *Lines* 7-20 has a complexity of $O(|V| \cdot (|E| - |V|))$, and the complexity of the while-loop for tree-expanding (*Lines* 24-33) is $O(|V| + \mathcal{N}_c)$. Hence, the overall time complexity of *Algorithm* 2 is $O(\mathcal{M} \cdot (|V| \cdot |E| - |V|^2 + \mathcal{N}_c))$.

Algorithm 1: CP Design to Map Controllers to Switches

Input: $G(V, E)$, $\mathcal{C}$, and $\mathcal{N}_c$
Output: Mappings between controllers and switches ($\mathbb{M}$)
1 preselect $\mathcal{M}$ combinations where the i-th one assigns the number of switches to the controllers as $\{k^i_1, \cdots, k^i_{\mathcal{N}_c}\}$;
2 compute the shortest path for each node pair in $G(V, E)$;
3 sort nodes in V in descending order of their degrees;
4 select the first $\mathcal{N}_c$ nodes to place controllers;
5 $\mathbb{M} = \emptyset$;
6 **for** $i = 1$ *to* $\mathcal{M}$ **do**
7 $m = \emptyset$;
8 **for** $j = 1$ *to* $\mathcal{N}_c$ **do**
9 take the j-th controller in sorted order;
10 assign the controller to k^i_j unassigned switches whose shortest paths to it are the shortest;
11 **end**
12 store the obtained controller-to-switch mapping in m and insert m into $\mathbb{M}$;
13 **end**
14 **return** $\mathbb{M}$;

IV. Performance Evaluation

A. Benchmarking with Single-level Optimization

The CP design with the bilevel optimization model basically balance the tradeoff between the security and latency of a CP. Then, to verify its necessity and effectiveness, we need to benchmark it with single-level optimization models. More specifically, we modify the bilevel optimization model to two single-level models, *i.e.*, the security- and latency-prioritized ones. For the security-prioritized one, we only consider the upper-level optimization subject to the constraints in Eqs. (3)-(11) and (13)-(15) and those for preventing cycles in path computation. While for the latency-prioritized one, we only take the lower-level optimization into account and make it subject to the constraints Eqs. (11) and (13)-(15). Here, our simulations using the 7-node, 8-node and NSFNET topologies in Fig. 2, and we set the capacity of each controller as $\mathcal{C} = 7$, $\mathcal{C} = 5$, and $\mathcal{C} = 7$, respectively.

Algorithm 2: CP Design with Minimum Spanning Trees

Input: $G(V, E)$ and $\mathbb{M}$
Output: Final CP design

calculate the minimum spanning tree $\mathcal{T}$ in $G(V, E)$;
$\tilde{\mathcal{V}}_{\max} = |V| \cdot (|V| - 1)$, $\mathbb{D} = \emptyset$;
for $i = 1$ *to* $\mathcal{M}$ **do**
  $\mathcal{T}_1 = \mathcal{T}$;
  take the i-th controller-to-switch mapping m_i in $\mathbb{M}$;
  calculate the vulnerability $\mathcal{V}_{\max}$ with $\mathcal{T}_1$ and m_i;
  while *there are unchecked links on* $\mathcal{T}_1$ **do**
    delete an unchecked link from $\mathcal{T}_1$ to get $\mathcal{T}_2$;
    mark the deleted link as a checked one;
    for *each link that is not on* $\mathcal{T}_1$ **do**
      add the link to $\mathcal{T}_2$ to get $\mathcal{T}_3$;
      if $\mathcal{T}_3$ *is a connected graph* **then**
        calculate the vulnerability $\mathcal{V}'_{\max}$ with $\mathcal{T}_3$ and m_i;
        if $\mathcal{V}'_{\max} \leq \mathcal{V}_{\max}$ **then**
          $\mathcal{T}_1 = \mathcal{T}_3$, $\mathcal{V}_{\max} = \mathcal{V}'_{\max}$;
          mark the newly-added link as a checked one in $\mathcal{T}_1$;
        end
      end
    end
  end
  $flag = 1$;
  route control channels in $\mathcal{T}_1$ with shortest paths;
  get the total length of all control channels as $\mathcal{L}$;
  while $flag = 1$ **do**
    find the shortest path in $G(V, E)$ (for a node pair in m_i) to add on $\mathcal{T}_1$ such that the resulting $\mathcal{T}'_1$ provides the minimum $\mathcal{V}'_{\max}$ for m_i;
    route control channels in $\mathcal{T}'_1$ with shortest paths;
    get the total length of all control channels as $\mathcal{L}'$;
    if $\mathcal{L}' < \mathcal{L}$ *OR* $\mathcal{V}'_{\max} < \mathcal{V}_{\max}$ **then**
      $\mathcal{T}_1 = \mathcal{T}'_1$, $\mathcal{L} = \mathcal{L}'$, $\mathcal{V}_{\max} = \mathcal{V}'_{\max}$;
    else
      $flag = 0$;
    end
  end
  if $\mathcal{V}_{\max} < \tilde{\mathcal{V}}_{\max}$ **then**
    $\tilde{\mathcal{V}}_{\max} = \mathcal{V}_{\max}$, $\mathbb{D} = \{m_i, \mathcal{T}_1\}$;
  end
end

Table I summarizes the comparisons between the bilevel model and the single-level ones, when they are applied to design CPs for the SDN topologies in Figs. 2(a) and 2(b). Here, both topologies are modified from realistic ones. We can see that compared with the single-level model prioritized for latency, the CP design obtained by the bilevel model provides smaller vulnerability, while it outperforms the security-prioritized single-level model in terms of average latency. This verifies the effectiveness of our bilevel model. We also notice that the bilevel model may provide a larger vulnerability than the security-prioritized single-level model. This is because the bilevel model considers the vulnerability and average latency of CP jointly, and to minimize the vulnerability, it might not be able to route the control channels with the shortest paths.

TABLE I
COMPARISONS OF SINGLE-LEVEL AND BILEVEL OPTIMIZATION MODELS

Topology/$\mathcal{N}_c$		Single-level Model		Bilevel Model
		Latency-prioritized	Security-prioritized	
7-Node/1	Latency (km)	16255.9	35697.1	16536.7
	$\mathcal{V}_{\max}$	3	2	2
8-Node/2	Latency (km)	1530	1948	1600
	$\mathcal{V}_{\max}$	3	1	2

B. Comparison between Bilevel Model and Heuristic

As for relatively large instances of the CP design problem, using the MILP to solve the bilevel optimization model would become intractable. Hence, we compare the performance of the MILP and the heuristic with *Algorithms* 1 and 2 using the small-sized topologies in Figs. 2(a) and 2(b). The results are listed in Table II, which indicate that the heuristic provides very similar results as the MILP but consumes much shorter computation time. This confirms the heuristic's performance.

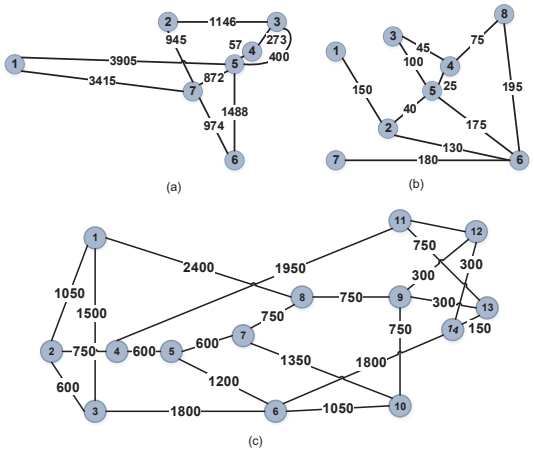

Fig. 2. Topologies used in simulations (link lengths in kilometers), (a) 7-node, (b) 8-node, and (c) NSFNET.

TABLE II
COMPARISONS BETWEEN MILP AND HEURISTIC

Topology	$\mathcal{N}_c$	Algorithm	$\mathcal{V}_{\max}$	Latency (km)	Running Time (s)
7-Node	1	MILP	2	16536.7	1120
		Heuristic	2	16938.4	0.032
8-Node	2	MILP	1	1600	41423.1
		Heuristic	1	1790	0.26

Finally, in order to further study the CP design problem, we apply the heuristic to the relatively large topology in Fig. 2(c), and change $\mathcal{N}_c \in [2, 6]$. The specific CP design for $\mathcal{N}_c = 3$ is presented in Table III, where a "C-S Path" refers to the routing path of a control channel between a controller and a switch and a "C-C Path" is the path of a control channel between two controllers. It can be seen that in the designed CP, certain fiber links are not selected to diverge the control channels from

sharing a large number of links, and thus the vulnerability can be reduced. Meanwhile, the controllers placed at *Nodes* 9 and 6 manage more switches than the one at *Node* 1, which suggests that placing a controller on the node with a larger degree and letting it manage more switches would help to reduce the vulnerability of the designed CP.

TABLE III
CP DESIGN IN NSFNET TOPOLOGY WITH $\mathcal{N}_c = 3$

Controller	#1	#2	#3
Location	6	9	1
Managed Switches	6, 10, 5, 3, 4	9, 12, 13, 14, 8, 11	1, 2, 7
C-S Paths (Duplex)	6-10 6-5 6-3 6-5-4	9-12 9-13 9-8 9-13-14 9-12-11	1-2 1-2-4-5-7
C-C Paths (Duplex)	6-10-9 6-3-1	9-10-6 9-8-1	1-3-6 1-8-9
CP Topology (E')	1-2, 1-3, 1-8, 2-4, 3-6, 4-5, 5-7, 5-6, 6-10 7-8, 8-9, 9-10, 9-12, 9-13, 11-12, 13-14		
$\mathcal{V}_{\max}$	2		
Latency (km)	41700		

The average latency and vulnerability of the designed CPs for $\mathcal{N}_c \in [2, 6]$ are plotted in Fig. 3. It is interesting to see that the values of the latency and vulnerability first decreases and then increase with $\mathcal{N}_c$, with the turning point at $\mathcal{N}_c = 3$. This suggests that for each topology, there might be a best setting for $\mathcal{N}_c$, and placing too many controllers in a CP might not help to reduce the average latency of its control channels and its vulnerability. Specifically, for the NSFNET topology in Fig. 2(c), the best setting places $\mathcal{N}_c = 3$ controllers in the CP.

This phenomenon can be explained as follows. When $\mathcal{N}_c$ is few, each controller covers many switches, and thus the control channels between controllers and switches might be routed over large hop-count paths. This will inevitably push up the latency and vulnerability of the designed CP. As $\mathcal{N}_c$ increases, each controller will have a smaller coverage in switches, but the control channels for mutual interactions among the controllers will increase. The joint effect will make the latency and vulnerability first decrease and then increase with $\mathcal{N}_c$.

V. CONCLUSION

This paper studied the CP design that takes planned physical-layer attacks into account. We first formulated the problem as a bilevel optimization, where the network planner tackles the upper-level optimization to design the CP with the minimum vulnerability to physical-layer attacks, while the attacker uses the lower-level optimization to plan and launch attacks to disturb the designed CP. Then, we transformed the bilevel optimization into an MILP and also proposed a heuristic to solve the problem time-efficiently. Simulation results verified that by reducing its vulnerability, our proposals can effectively protect a CP from planned physical-layer attacks, and they also successfully balance the tradeoff between the security and latency of a designed CP.

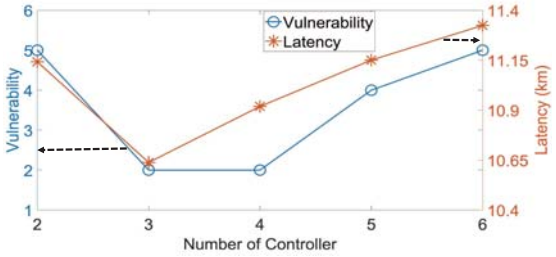

Fig. 3. Properties of CPs designed with NSFNET topology.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC projects 61871357 and 61771445, CAS Key Project (QYZDY-SSW-JSC003), and SPR Program of CAS (XDC02070300).

REFERENCES

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] B. Zhao, X. Chen, J. Zhu, and Z. Zhu, "Survivable control plane establishment with live control service backup and migration in SD-EONs," *J. Opt. Commun. Netw.*, vol. 8, pp. 371–381, Jun. 2016.

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Prof. of HotSDN 2012*, pp. 7–12, Aug. 2012.

[4] M. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. of CNSM 2013*, pp. 18–25, Oct. 2013.

[5] X. Chen *et al.*, "Leveraging master-slave OpenFlow controller arrangement to improve control plane resiliency in SD-EONs," *Opt. Express*, vol. 23, pp. 7550–7558, Mar. 2015.

[6] H. Huang *et al.*, "Realizing highly-available, scalable and protocol-independent vSDN slicing with a distributed network hypervisor system," *IEEE Access*, vol. 6, pp. 13 513–13 522, 2018.

[7] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[8] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.

[9] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska, "Physical-layer security in evolving optical networks," *IEEE Commun. Mag.*, vol. 54, pp. 110–117, Aug. 2016.

[10] J. Zhu, B. Zhao, and Z. Zhu, "Leveraging game theory to achieve efficient attack-aware service provisioning in EONs," *J. Lightw. Technol.*, vol. 35, pp. 1785–1796, May 2017.

[11] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[12] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[13] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[14] J. Zhu, B. Zhao, and Z. Zhu, "Attack-aware service provisioning to enhance physical-layer security in multi-domain EONs," *J. Lightw. Technol.*, vol. 34, pp. 2645–2655, Jun. 2016.

[15] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Trans. Cloud Comput.*, vol. 2, pp. 436–447, Oct. 2014.

[16] L. Müller *et al.*, "Survivor: An enhanced controller placement strategy for improving SDN survivability," *in Proc. of GLOBECOM 2014*, pp. 1909–1915, Dec. 2014.

[17] J. Zhu *et al.*, "How to survive targeted fiber cuts: A game theoretic approach for resilient SDON control plane design," *in Proc. of ONDM 2019*, pp. 1–6, May 2019.

[18] B. Colson, P. Marcotte, and G. Savard, "Bilevel programming: A survey," *4OR*, vol. 3, pp. 87–107, Jun. 2015.

[19] B. Kara and V. Verter, "Designing a road network for hazardous materials transportation," *Transport. Sci.*, vol. 38, pp. 188–196, May 2004.