Journal of Network and Computer Applications 178 (2021) 102969

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Secure and cost-effective controller deployment in multi-domain SDN with Baguette

Wendi Feng, Chuanchang Liu *, Bo Cheng, Junliang Chen

Beijing University of Posts and Telecommunications, 10 Xitucheng RD, 100876, Beijing, China

ARTICLE INFO

Keywords:
Multi-domain SDN
Controller security
Attack mitigation

ABSTRACT

Software-Defined Networking (SDN) is becoming prevalently in recent years. Practical SDN (*e.g.*, production Software-defined Wide Area Network) deployments leverage multiple commercial controllers, which partitions the network into multiple domains, and each domain uses a dedicated controller. Commercial controllers are usually used for reliability and fully post-sales supports. However, using a single type of SDN controllers can compromise the whole network if the attacker can exploit its vulnerabilities. In this paper, we consider this security issue and present the *Secure and Cost-effective Controller Deployment (SCCD)* problem. The SCCD problem aims to replace a few controllers with different types of commercial SDN controllers, which satisfies the security requirement at a minimal cost. The complexity of the SCCD problem comes from *common vulnerabilities* shared among different types of SDN controllers and *attack propagations* among network domains. We prove the non-deterministic polynomial-time hardness (NP-hardness) of the problem and propose the Baguette algorithm to efficiently solve the problem. Baguette judiciously chooses and replaces controllers for *critical domains* with selected types of commercial SDN controllers. Simulation results show that Baguette can achieve comparable performance to the Optimal solution and can stably achieve up to 12.6x security enhancement compared with the single controller type deployment and reduce to 11.1% cost of the securest deployment.

1. Introduction

Software-defined Networking (SDN) (Kreutz et al., 2015) is a prevalent networking technology, which decouples the control plane and data plane of network devices (*i.e.*, SDN switches) and allows innovations to be easily applied. It also simplifies network management with the fine-grained network controlling and monitoring mechanisms. Beside, it provides open interfaces that allow customized functionality and new network wide applications to be quickly deployed (Arashloo et al., 2016). Owing to the advantages, SDN plays an important role in Cloud Computing (Hayes, 2008), Edge Computing (Shi et al., 2016), 5G (Boccardi et al., 2014), and Fog Computing (Bonomi et al., 2012). Many industrial companies, such as Google (Jain et al., 2013) and Facebook (Choi et al., 2018), have started deploying SDN in their production environments. AT&T, as one of the biggest Internet Service Providers (ISP), also aims to increase the SDN deployment to 75% by the end of 2020 (Fuetsch, 2020).

Large scale commercial SDNs (*e.g.*, production Software-defined Wide Area Networks) leverage the *multi-domain* deployment as the ever-growing demand expands the network that requires scalability, reliable, and high performance. In multi-domain SDNs, each domain is deployed with a *controller*.[1] Fig. 1 depicts a network that is partitioned into three domains, and each controller connects to its domain's SDN switches. The controller and switches are connected with the *in-band* mode (Jain et al., 2013; Hong et al., 2013; Yap et al., 2017), that control messages share the communication channel with data packets. Production networks usually utilize commercial controllers as they are more reliable and provide fully post-sales supports (Teo et al., 2016).

Existing work consider data communications (Phemius et al., 2014), controller placement (Guo et al., 2019), and reliability enhancement (Song et al., 2017) in multi-controllers of multi-domain SDNs. They use only one type of SDN controller and fail to consider the security benefits of deploying different types of SDN controllers. If the attacker compromises one controller by leveraging its vulnerabilities, other controllers in other domains with the same type can also be compromised, and thus the whole network can be compromised. We call it the *multi-domain controller attack*. For example, the attack first uses an end-host and captures the "switch – controller" connection or disconnection packets.

* Corresponding author.
E-mail address: lcc3265@bupt.edu.cn (C. Liu).

[1] We use "controller" and "SDN controller" interchangeably in this paper.

https://doi.org/10.1016/j.jnca.2020.102969
Received 29 September 2020; Received in revised form 8 December 2020; Accepted 25 December 2020
Available online 7 January 2021

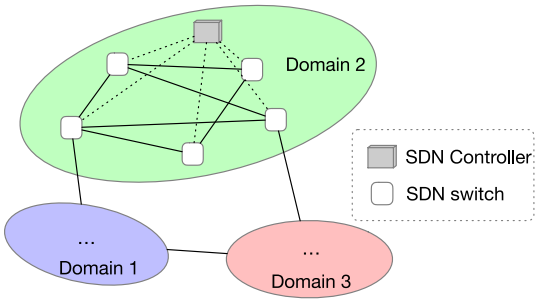


Fig. 1. The multi-domain SDN deployment. Each domain has a dedicated controller. " …" in Domains 1 and 3 represent the omitted SDN devices and connections.

The attacker then counterfeits a disconnection packet and exploits the controller's control message validation vulnerability to disconnect the switches from the "real" controller and then re-connect to the "fake" one (the controlled end host). This domain is thus compromised. Next, the attacker can further propagate the attack to adjacent domains if their controllers have the *common vulnerability* that is exploited by the attacker. Finally, the whole network can be compromised. (Scott-Hayward et al., 2016; Chica et al., 2020; Scott-Hayward et al., 2013).

The question is whether we can mitigate attacks from "spreading" out to other domains and make the network resilient to attacks by using different types of controllers in the commercial SDN deployment? The answer is yes. Different types of controllers may potentially have vulnerabilities,[2] but an attacker may only be able to exploit one or a few specific vulnerabilities. When using a device whose vulnerabilities cannot be exploited, the attack cannot compromise the controller and the domain.

Simply using different types of controllers at different domains can be either unnecessary or insufficient. This is because i) practical SDN networks use commercial SDN controllers for better services, the deployment cost will be exorbitant because this often uses more types than needed and increases the expense when the security level has already been satisfied (see Section 3.2.2). ii) Security level may fail to be provided as *common vulnerabilities* reside in multiple switch types. Hence, the attack can still propagate and compromise the network.

Inspired by the above observation, we present the *Secure and Cost-effective Controller Deployment (SCCD)* problem in this paper, which jointly considers security and the controller deployment cost in the commercial multi-domain SDN controller deployment. In a nutshell, SCCD aims to identify few critical domains in the multi-domain SDN and replaces their existing controllers with specific types to satisfy the security requirement at a minimum cost. The network functionality will not be influenced due to the openness (see Section 2.3). The complexity of the problem results from *common vulnerabilities* of controller types and *attack propagations* among domains. We prove the SCCD problem is Non-deterministic Polynomial-time Hard (NP-hard), and hence, we propose an efficient heuristic algorithm called Baguette to solve it. Baguette judiciously chooses a small portion of the critical domains and deploy smartly selected different types of controllers for them. Our simulation results show that Baguette can stably achieve near-optimal performance with up to 12.6x security enhancement and down to 11.1% cost of the most secure method.

In summary, our contribution is three-fold:

- We identify the multi-domain controller attack in commercial multi-domain SDN deployments, and mathematically formulate the SCCD problem.
- We prove the NP-hardness of the SCCD problem and propose Baguette to efficiently solve it. Baguette can satisfy the security requirement with a minimum deployment cost.
- We evaluate Baguette under various real-world topologies and compare them with baseline algorithms. Simulation results show that Baguette can stably achieve near-optimal performance.

The remainder of this paper is organized as follows. Section 2 introduces the fundamental background knowledge for SDN and multi-domain SDN. Section 3 first presents the considered attack model and real-world attack examples and then motivates the SCCD problem with examples. Section 4 defines the security requirement and mathematically formulates the SCCD problem. In Section 5, we prove the NP-hard complexity for the SCCD problem and propose Baguette to efficiently solve it. Section 6 describes the simulation setup and compares Baguette with baseline algorithms. Section 7 surveys the most relevant related works, and Section 8 concludes the paper.

2. Background

Before presenting the attack model, we first briefly introduce the SDN and multi-domain SDN network as well as their benefits.

2.1. Software-defined networks

SDN is a state-of-the-art network architecture that decouples the *control plane* and the *data plane* by removing the control plane from the network device and leaving the basic forwarding functionality. This differs from traditional networks that couple both planes in the same device. As shown in Fig. 2, SDN uses a (logically) central controller to get the "global view" of the network and to guide the data plane (controlled devices) forwarding packets. The communications between the controller and SDN switches can be either *in-band* or *out-of-band* using SDN protocols (*e.g.*, OpenFlow (McKeown et al., 2008)). The former transmits the control messages by sharing the links with the data plane, while the latter uses dedicated "controller – switch" links.

2.2. Multi-domain SDN

As the network enlarges (*e.g.*, the number of SDN switches increases), a single controller becomes insufficient in handling the large demand of controlling the devices, and thus multiple controllers are needed. In which, each controller can only control a small number of SDN switches and synchronize the topology information with others. The control plane is still logically centralized but becomes more powerful. Practical SDN (especially software-defined wide area networks) deployments employ this scheme, which partition the whole network into many domains (sites). Each domain contains many SDN switches controlled by one (or more for resiliency) controller, and the control messages are

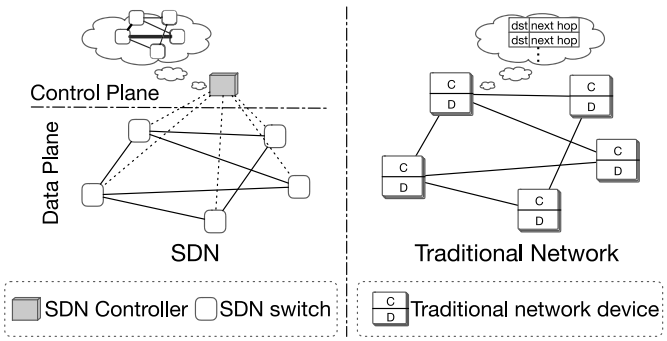


Fig. 2. Software-defined networks versus traditional networks. "C" and "D" in the traditional network device icon represent the control plane and the data plane, respectively.

[2] The controller's vendor may report vulnerabilities to its customer, or the network management team may identify the vulnerabilities.

passed in-band (see Fig. 1).

2.3. Openness of SDN brings new opportunities

One of the greatest advantages provided by SDN is that the architecture is *open*. This allows new, customized, or 3rd-party network applications to be quickly implemented and deployed on the network with the same set of underlying network topology and infrastructure. The openness is achieved by decoupling the control plane and data plane and adopting unified *south-bound* (between SDN switches and the SDN controller) and flexible *north-bound* APIs (between SDN controller and the network management applications). The network management applications can run at the controller to instruct the forwarding behavior under different requirements. This openness gives us the flexibility that functionality can be easily realized using different types of controllers.

3. Attack model and motivation

In this section, we first present our attack model. Base on the attack model, we then present examples to motivate the SCCD problem.

3.1. Attack model and real-world examples

Our attack model is based on the 0day vulnerabilities (Bilge and Dumitraş, 2012), which have not yet been patched. We consider the case where the *defender side* (controller vendors and network operators) masters more vulnerability information than that of the *attacker side* (attackers). This is because controller vendors possess the full design and implementation details of the controller, and vulnerable information is prone for them to get. They can further report this information to their users (network operators). Besides, their users may have a dedicated security team to identify possible software flaws and to defend against attacks.

In this section, we present the attack model and two possible real-world examples. As shown in Fig. 3, our attack model considers compromising the *whole* multi-domain SDN network by incrementally compromising one domain of the SDN network by leveraging the vulnerability on the controller and then propagates the attack to other domains. The attack propagation is conducted by exploiting *common vulnerabilities* (Li et al., 2016) shared across controllers since designing bug-free computer systems is notoriously hard (Vizarreta et al., 2020).

Common vulnerabilities usually happen due to the reuse of codes. Since modern software development uses existing libraries to avoid "building the wheels", when the used library contains vulnerabilities, all software using the library can suffer from security threats. Hence, if two controllers use the same flawed library, they will share common vulnerabilities, and thus, even using different types of controllers, the attack can still attack both of them.

The multi-domain controller attack has the following characteristics. i) The attacker compromises a domain by leveraging the vulnerabilities of its controller. ii) The attacker can propagate to adjacent domains if the current domain's controller is compromised, and iii) if the domain can mitigate the attack, the attack will terminate. Many attack schemes in the real-world have these characteristics. We list two attack examples as follows.

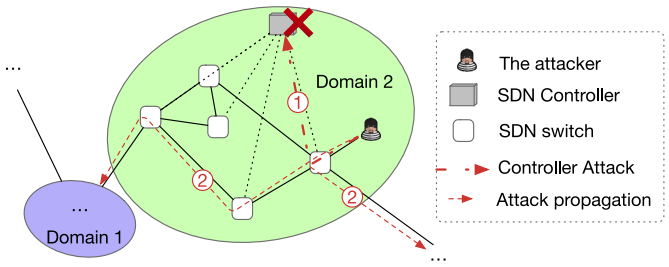


Fig. 3. The multi-domain controller attack demonstration. The attacker ① first compromise one domain by attacking the controller, and it ② then controls the controller and propagates the attack to compromise other domains. " …" represent the omitted entities.

3.1.1. Unauthorized access attack

Authentication is the most critical defense of attacks, but authentication mechanisms are complex and prone to flawed design. This is because the in-band "controller – switch" communication mode uses the same links as transmitting data, and hence, the attacker may access the controller by attacking its authentication system (*e.g.*, password guessing). After compromising the controller, the attacker can modify the flow tables and change the forwarding behavior to conduct further attacks.

3.1.2. Controller hijacking attack

Another possible method of gaining access to the controller is by hijacking the "controller – switch" communication with carefully impersonated packets. If the controller fails to provide essential protections, the attacker can capture these control packets (*e.g.*, the "controller – switch" connection and disconnection packets) with an end-host E in the domain and then counterfeit the packet with the E's information to "mislead" SDN switches to connect E. The attacker can then change the routing policies and conduct the same kind of controller hijacking in adjacent domains.

3.2. Motivation

This sub-section presents the SCCD problem examples to show that the single SDN controller type fails to mitigate attack propagations, and thus multiple controller types are needed to enhance security. However, solely considering the security factor results in exorbitant cost, but concerning the cost alone fails to guarantee the security requirement. Thus, an intelligent multi-type controller deployment mechanism by jointly premeditating security and cost to achieve the *Secure and Cost-effective Controller Deployment (SCCD)* in multi-domain commercial SDNs is needed. For brevity, we represent a domain as a node.

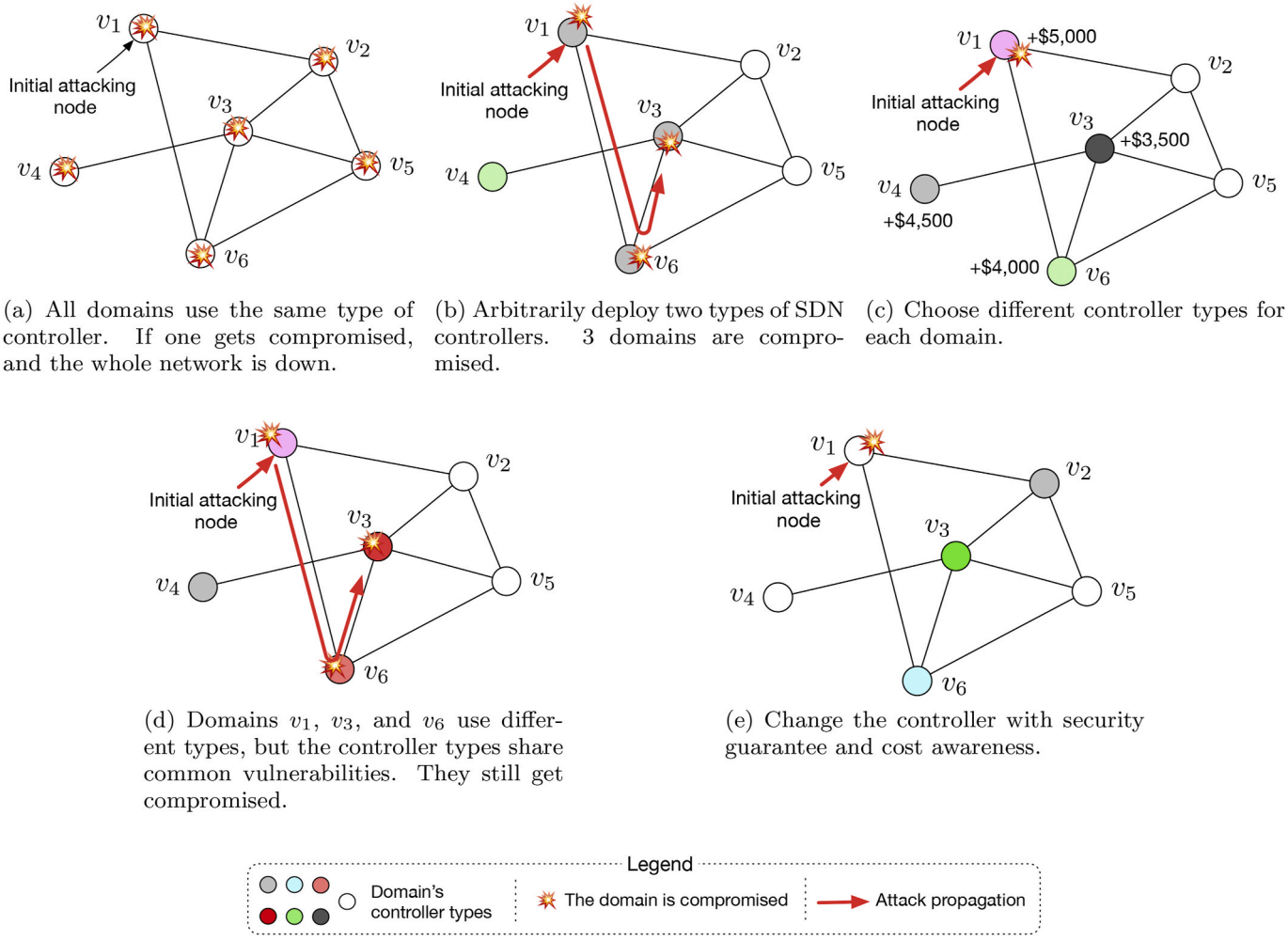
3.2.1. Insecure single controller type deployment

Fig. 4a demonstrates that merely using one type of controller to deploy all domains can lead to vulnerabilities being shared across domains, and hence if any domain (*e.g.*, domain v_1) is compromised by an attack, the attack can propagate to all other domains and compromise the whole network. This shows that a single controller type deployment is insecure, and multiple types should be employed for security enhancement.

3.2.2. Curse of arbitrary deployment

Generally, multiple controller types deployment can enhance security because the vulnerabilities can be different on different types of controllers. However, arbitrarily changing controllers with multiple types would also fail to protect the network. In Fig. 4b, when domains v_1, v_3, v_4, and v_6 are randomly selected to change to another type of controller, where v_1, v_3, and v_6 use one controller type (shown in gray color), and v_4 uses another type (shown in light green color). The gray and light green types do not share common vulnerabilities. Suppose the attack arrives at v_1, and the attack can propagate to domain v_6, then to v_3, and finally to v_4. Thus v_1, v_3, and v_6 are compromised. When the attack propagates to v_4, it cannot compromise v_4, and the attack terminates. However, half of the network domains are compromised.

As depicted in Fig. 4c, using different types for each domain may mitigate the attack. However, the overall cost would be exorbitant because the security requirement can be satisfied by changing the controllers of a few numbers of domains. Since each type of commercial controller has different costs (Cisco Systems, Inc; Huawei Technologies Co., Ltd; Juniper Networks, Inc; Telefonaktiebolaget LM Ericsson), one



(a) All domains use the same type of controller. If one gets compromised, and the whole network is down.

(b) Arbitrarily deploy two types of SDN controllers. 3 domains are compromised.

(c) Choose different controller types for each domain.

(d) Domains v_1, v_3, and v_6 use different types, but the controller types share common vulnerabilities. They still get compromised.

(e) Change the controller with security guarantee and cost awareness.

Fig. 4. Motivation examples demonstration. Each node represents a network domain that has one SDN controller. Unsuccessful attack propagation is omitted for a clearer illustration, and we suppose the first entry domain can always be compromised to demonstrate attack propagations.

possible way to lower the cost would be simply sorting the controller types based on their cost from low to high and change the type of the node using the sorted types one by one to minimize the cost. However, different controller types may contain *common vulnerabilities* due to the use of shared libraries. Hence, attacks can leverage these common vulnerabilities to compromise the network. As shown in Fig. 4d, even domains v_1, v_3, and v_6 use different types, the attack can still propagate from v_1 to v_6 and v_3, and then, compromise all of them. Consequently, arbitrarily deploying the controller types is unreliable.

3.2.3. Security and cost-effective deployment

By selecting vulnerability-distinct controller types adjacent to the attack entry domain, the attack can be mitigated at the initial stage. In Fig. 4e, the controller type deployment can efficiently mitigate the attack on its propagation path and achieve the minimum overall cost through the SCCD intelligent SDN controller type deployment scheme. This paper presents the problem of finding the SCCD scheme that guarantees the security requirements at a minimum cost.

4. Problem formulation

In this section, we first mathematically present the network system description and then propose security metrics to describe the possibility of compromising the network. We further introduce constraints and the objective function of the SCCD problem. Finally, we formulate the problem as an optimization problem.

4.1. System description

We mathematically formulate the network in this subsection. All notation definitions can be found in Table 1. A multi-domain SDN network can be divided into multiple domains and represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \dots\}$ is the set of network domains, and v_i represents the ith domain in the network. $\mathcal{E}$ is the links set, and

Table 1
Notation definitions.

Notation	Description
$\mathcal{V}$	The network domain set. $\mathcal{V} = \{v_1, v_2, \dots\}$.
$\mathcal{S}$	The controller type set. $S = \{s_1, s_2, \dots\}$.
$\mathcal{C}$	The cost set. Costs of each type of controller. $C = \{c_1, c_2, \dots\}$.
x_{ij}	Domain v_i uses type s_j.
$\mathcal{F}_j$	Controller type s_j's vulnerabilities. $\mathcal{F}_j = \left\{f^j_1, f^j_2, \dots, f^j_{\lvert\mathcal{F}_j\rvert}\right\}$.
$\mathcal{A}_j$	The attack set of the controller type s_j. $\mathcal{A} = \{a_1, a_2, \dots\}$.
P_j	The attack probability of controller type s_j.
X	The controller type mapping scheme. $x_{ij} \in X, \forall v_i \in \mathcal{V}, \forall s_j \in \mathcal{S}$.
$e(X)$	The compromising expectation of the whole network under controller deployment X.
E^{max}	The maximum compromising expectation provided by network operator.
$\mathcal{V}_i^{*a_k}$	The compromising sub-graph domains set.
$r_i^{a_k}(X)$	The compromised ratio under the deployment scheme X and attacked by attack a_k when domain v_i is the entry domain.

each link connects two domains. Each domain in $\mathcal{V}$ can choose multiple controller types to deploy. Let $\mathcal{S} = \{s_1, s_2, \dots\}$ represents the controller type set. In the default setting, all domains are deployed with the same type of controller. Different controller types have different costs, and the cost of the existing controller type is 0 as no further expenses are needed for the deployed controller. Let $\mathcal{C} = \{c_1, c_2, \dots,\}$ represents the cost of purchasing each controller type. We use $x_{ij} = 1$ to denote network domain v_i is deployed with controller type s_j, and otherwise $x_{ij} = 0$.

4.2. Attack metrics

In this subsection, We introduce probability-based security metrics to measure the security of the network. Specifically, we use *attack probability* to measure the possibility of compromising a specific controller type by an attack. We then consider the influence of attack propagation with the *controller compromised ratio*. Finally, we present the *compromising expectation* to evaluate the overall compromising expectation of the network, and the security requirement is evaluated by the compromising expectation.

4.2.1. Attack probability

Each type of controller may have multiple vulnerabilities. Let $\mathcal{F}_j = \left\{f_1^j, f_2^j, \dots, f_{|\mathcal{F}_j|}^j\right\}$ be the set of all vulnerabilities of type s_j, where $|\mathcal{F}_j|$ is the number of vulnerabilities of type s_j. Let $\mathcal{F} = \bigcup_j \mathcal{F}_j$ be the total vulnerability set, and let $\mathcal{A} = \{a_1, a_2, \dots\}$ be the attack set. The attack set contains all possible attacks based on the reports of vendors or the network operator's previous experiences. Each attack can exploit one or more vulnerabilities, so that each attack a_k is a subset of the vulnerability set $\mathcal{F}$ (excluding $\varnothing$). Thus, the total number of attacks is $2^{|\mathcal{F}|} - 1$. If attack a_k can exploit the vulnerability that type s_j has, type s_j can be compromised by attack a_k.

The attack probability of controller type s_j is the ratio of the number of attacks that can compromise the type to the total number of attacks. It is formulated as

$$P_j = \frac{\left|\bigcup_{a_k \cap \mathcal{F}_j \neq \varnothing} \{a_k\}\right|}{|\mathcal{A}|}. \tag{1}$$

4.2.2. Domain compromised ratio

An attack can propagate from one domain to another if these domains are physically adjacency, and have the same type or common vulnerability sharing controllers deployed. We present the concept of the *compromising sub-graph* to identify the possible compromised domains if given entry network domain v_i and attack a_k.

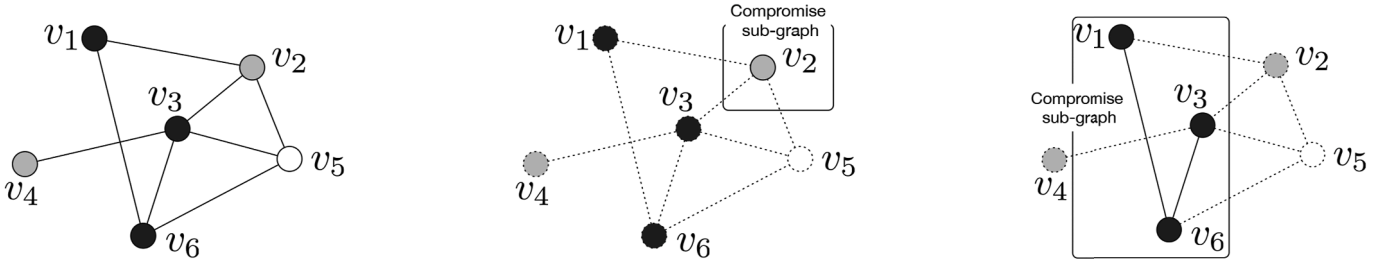
As shown in Algorithm 1, the compromising sub-graph is generated by 2 steps. i) Adjusting the adjacency matrix of the network graph, by removing the node (the domain) in the adjacency matrix whose type's vulnerabilities have no intersection with attack a_k; and ii) traversing the graph with the adjusted adjacency matrix by using breadth-first search (BFS) to get all the domains that can be reached in the graph traversal. Then, we get $\mathcal{G}_i'^{a_k} = (\mathcal{V}_i'^{a_k}, \mathcal{E}_i'^{a_k})$, which is the generated compromising sub-graph. Fig. 5a depicts a network with 6 domains where v_1, v_3, and v_6 use the "black" type, v_2, v_4 use the "gray" type, and v_5 uses the "white" type. In Fig. 5b, v_2 is the only element in $\mathcal{V}_2'^{a_1}$, and in Fig. 5c, $\mathcal{V}_1'^{a_2}$ contains v_1, v_3, v_6. The domain compromised ratio is formulated as

$$r_i^{a_k}(X) = \frac{|\mathcal{V}_i'^{a_k}|}{|\mathcal{V}|}. \tag{2}$$

Algorithm 1
Ratio sub-graph generation.

Algorithm 1: Ratio sub-graph generation.

Input: $Matrix_{\mathcal{G}}$, v_i, a, X
Output: $\mathcal{G}_i'^a$
/* Generate the adjusted adjacent matrix $Matrix_{\mathcal{G}'}$ */
/* Initialize $Matrix_{\mathcal{G}'}$ */
$Matrix_{\mathcal{G}'} \leftarrow Matrix_{\mathcal{G}}$;
/* Remove the adjacent between nodes which is not in a */
for $i \leftarrow 0$ **to** $|\mathcal{V}|$ **do**
  if $\sum_j x_{ij} s_j \in a$ **then**
    $Matrix_{\mathcal{G}'}[i] \leftarrow \vec{0}$
    continue
  for $k \leftarrow 0$ **to** $|\mathcal{V}|$ **do**
    if $\sum_j x_{kj} s_j \in a$ **then**
      $Matrix_{\mathcal{G}'}[i][k] \leftarrow 0$
Traverse the $Matrix_{\mathcal{G}'}$ graph with breadth-first search (BFS), and saves all the reached vertices into $\mathcal{V}_i'^a$ and all the reached links into $\mathcal{E}_i'^a$.
$\mathcal{G}_i'^a = (\mathcal{V}_i'^a, \mathcal{E}_i'^a)$



(a) The network has 3 controllers types (shown as black, gray, and white).

(b) An attack compromises gray type domain and enters at domain v_2.

(c) An attack compromises black type domain, and enters domain v_1.

Fig. 5. Compromising sub-graph demonstration. In Fig. 5b, the compromising sub-graph only contains v_2, and the compromised ratio is $\frac{1}{6}$. In Fig. 5c, the compromising sub-graph contains v_1, v_3, and v_6, and the compromised ratio is $\frac{3}{6} = \frac{1}{2}$.

4.2.3. Compromising expectation

In this subsection, we propose the *compromising expectation* to measure the overall compromising possibility of the network. The compromising expectation is formulated as a mathematical expectation shown below

$$e(X) = \frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\sum_{a_k\in\mathcal{A}}\sum_{s_j\in\mathcal{S}} r_i^{a_k}(X)P_j x_{ij}, \tag{3}$$

where $r_i^{a_k}(X)$ is the domain compromise ratio of domain v_i under attack a_k, and P_j is the attack probability of controller type s_j under all attacks.

4.3. Constraints

4.3.1. Maximum compromising possibility

Security requirements may vary between network to network. Therefore, a minimum security level (or put another way, a maximum compromising possibility) can be set by the network operator to guarantee the least security requirement of the network. Thus we have

$$\frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\sum_{a\in\mathcal{A}}\sum_{s_j\in\mathcal{S}} r_i^{a}(X)P_j x_{ij} \le E^{max}, \tag{4}$$

where E^{max} is the maximum required compromising possibility of the network.

4.3.2. Single controller type constraint

Only one controller type can be used for each domain in the network. This is written as

$$\sum_{s_j\in\mathcal{S}} x_{ij} = 1, \forall v_i \in \mathcal{V}. \tag{5}$$

4.4. Objective function

Our objective is to minimize the overall cost of controller deployment for each domain in the network. Therefore, the objective function is written as follows

$$obj = \sum_{v_i\in\mathcal{V}}\sum_{s_j\in\mathcal{S}} x_{ij}c_j. \tag{6}$$

4.5. Problem formulation

The goal of the SCCD problem is to find an optimal controller type deployment scheme between domains in $\mathcal{V}$ and types in $\mathcal{S}$ by judiciously placing the *suitable* type to the domain, which reaches the target of minimizing the overall cost under the security requirement. Consequently, we formulate the SCCD problem as follows:

$$\begin{aligned} \min_x \quad & \sum_{v_i\in\mathcal{V}}\sum_{s_j\in\mathcal{S}} x_{ij}c_j \\ \text{s.t.} \quad & (4)(5), \\ & x_{ij} \in \{0,1\}, \\ & v_i \in \mathcal{V}, s_j \in \mathcal{S}, \end{aligned} \tag{P}$$

where $\{c_j\}$ are constants, and $\{x_{ij}\}$ are designed variables. In the SCCD problem, the objective function is linear, and variables are binary integers. Thus, this problem is an Integer Linear Programming (ILP) problem.

5. Solution

In this section, we first propose the analysis on the complexity of the SCCD problem and then present a heuristic algorithm called Baguette to solve the problem.

5.1. Complexity analysis

In this subsection, we reduce a special case of the SCCD problem to the *Graph Coloring Problem (GCP)* (Jensen and Toft, 2011) and prove the NP-hardness of the SCCD problem.

Theorem 1. *For a special case with the following four conditions, the Secure and Cost-effective Controller Deployment is NP-hard.*

(1) *Each network domain can only be deployed with one type of controller.*
(2) *Vulnerabilities of each type are different.*
(3) *The compromising expectation of the network is zero.*
(4) *The costs of all types of controllers are the same.*

Proof 1. *We first introduce the GCP problem. The GCP problem aims to minimize the number of colors used for a graph node, where each node has one color, and adjacent nodes have different colors. A typical formulation of the GCP problem is shown as follows.*

$$\begin{aligned} \min_x \quad & \sum_j w_j \\ \text{s.t.} \quad & \sum_j x_{ij} = 1, \forall i \in V, \\ & x_{ij} \in \{0,1\}, \\ & \forall x_{uj} + x_{vj} \le 1, (u,v) \in E, j \in C, \\ & x_{ij} \le w_j, \forall i \in V, j \in C, \\ & x_{uj} + x_{vj} \le w_j, \forall (u,v) \in E, j \in C, \end{aligned} \tag{7}$$

where $G = (V, E)$ is a graph. V and E are vertex and edge sets, respectively. C is the color set. $x_{ij} = 1$ denotes color j is mapped on vertex i, and 0 otherwise. (u, v) denotes an edge in the edge set E. $w_j = 1$ if at least one vertex is mapped with color j, and 0 otherwise. It has been proved that the GCP is NP-hard (Jensen and Toft, 2011).

We then prove for a special case under conditions (1)–(4), problem P and the GCP are equivalent. Given condition (1), only one version can be mapped to one network node. Thus, we have

$$\begin{aligned} & x_{ij} \le w_j, \quad && \forall v_i \in \mathcal{V}, s_j \in \mathcal{S}, \\ & x_{v_1 j} + x_{v_2 j} \le 1, \quad && \forall (v_1, v_2) \in \mathcal{E}, s_j, \in \mathcal{S}, \end{aligned} \tag{8}$$

where $w_j = 1$ denotes that at least one domain uses type s_j, and otherwise $w_j = 0$. Given condition (2), $\forall s_{j_1}, s_{j_2} \in \mathcal{S}$, we have $F_{j_1} \cap F_{j_2} = \emptyset$. Thus, given condition (3), the system has the maximum security and the minimum compromising expectation. Each attack can only compromise one domain's controller because if an attack compromises a domain's controller, it cannot propagate to the succeeding domains. Thus, we have

$$e(X)=\frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\sum_{a\in\mathcal{A}}\sum_{s_j\in\mathcal{S}} r_i^a(X)P_j x_{ij}=\frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\left(\frac{\left|\bigcup_{a_k\cap\mathcal{F}_j\neq\emptyset}\{a_k\}\right|}{|\mathcal{V}|}\times P_j\times x_{ij}\right)$$
$$=\frac{\left|\bigcup_{a_k\cap\mathcal{F}_j\neq\emptyset}\{a_k\}\right|^2}{|\mathcal{V}|^2|\mathcal{A}|}, \tag{9}$$

is a constant. Therefore, given an edge $(v_1,v_2)\in\mathcal{E}$*, we have*

$$\forall s_j\in\mathcal{S}, x_{v_1 j}+x_{v_2 j}\le w_j. \tag{10}$$

For condition (4), let the cost of controllers be a constant c*, thus problem P can be reformulated as*

$$\begin{aligned}\min_w\quad & \sum_{s_j\in\mathcal{S}} c\times w_j\\ \text{s.t.}\quad & (5)(8)(10),\\ & w_j\in\{0,1\}\end{aligned} \tag{P'}$$

Problem P′ aims to minimizing the total number of types at the maximum security requirement ($\frac{1}{|\mathcal{V}|}$)*, which is the adjacent domains having different controller types.* **Problem P' is a special case of the GCP.** Since the GCP is NP-hard, therefore, we can conclude that:

Theorem 2. *The Secure and Cost-effective Controller Deployment problem is NP-hard.*

Algorithm 2. The BAGUETTE algorithm.

5.2. The Baguette algorithm

The complexity of the SCCD problem comes from both the attack propagations among network domains, and vulnerabilities can share between controller types. Due to the NP-hard complexity, we present an efficient heuristic algorithm called Baguette to solve the SCCD problem.

As shown in Fig. 6, the idea behind the Baguette algorithm is to replace the controller types of *critical domain* for mitigating attacks. When critical domains are adjacent to each other, they should use different controller types with the highest *vulnerability differentiation* to prevent the attack from propagating. Baguette follows three steps to solve the problem as follows.

(1) **Preparing critical network domains.** As depicted in Fig. 6a, Baguette identifies critical network domains based on their degrees and stores the critical domains in an array in the order of each domain should be processed. This is achieved by sorting the domain nodes based on their degrees in the descendent order.
(2) **Preparing controller type candidates.** The objective of the SCCD problem is to minimize the deployment cost. Hence, Baguette sorts the types based on their costs from low to high, which ensures cheaper controller types can be prioritized considered in the mapping procedure (see Fig. 6a).
(3) **Mapping controller types.** Baguette repeatably picks a critical domain from the sorted critical domain array and selects a type of the controller for it until the whole network satisfies the security requirement (shown in Fig. 6b–d). Baguette maintains a current global minimum security mapping, and whenever a domain candidate is mapped with a controller type, there is a new mapping X. Baguette compares the current minimum compromising expectation (the security requirement) with X's expectation in the process to get the minimum compromise expectation of all tested mappings if Baguette cannot satisfy the security requirement.

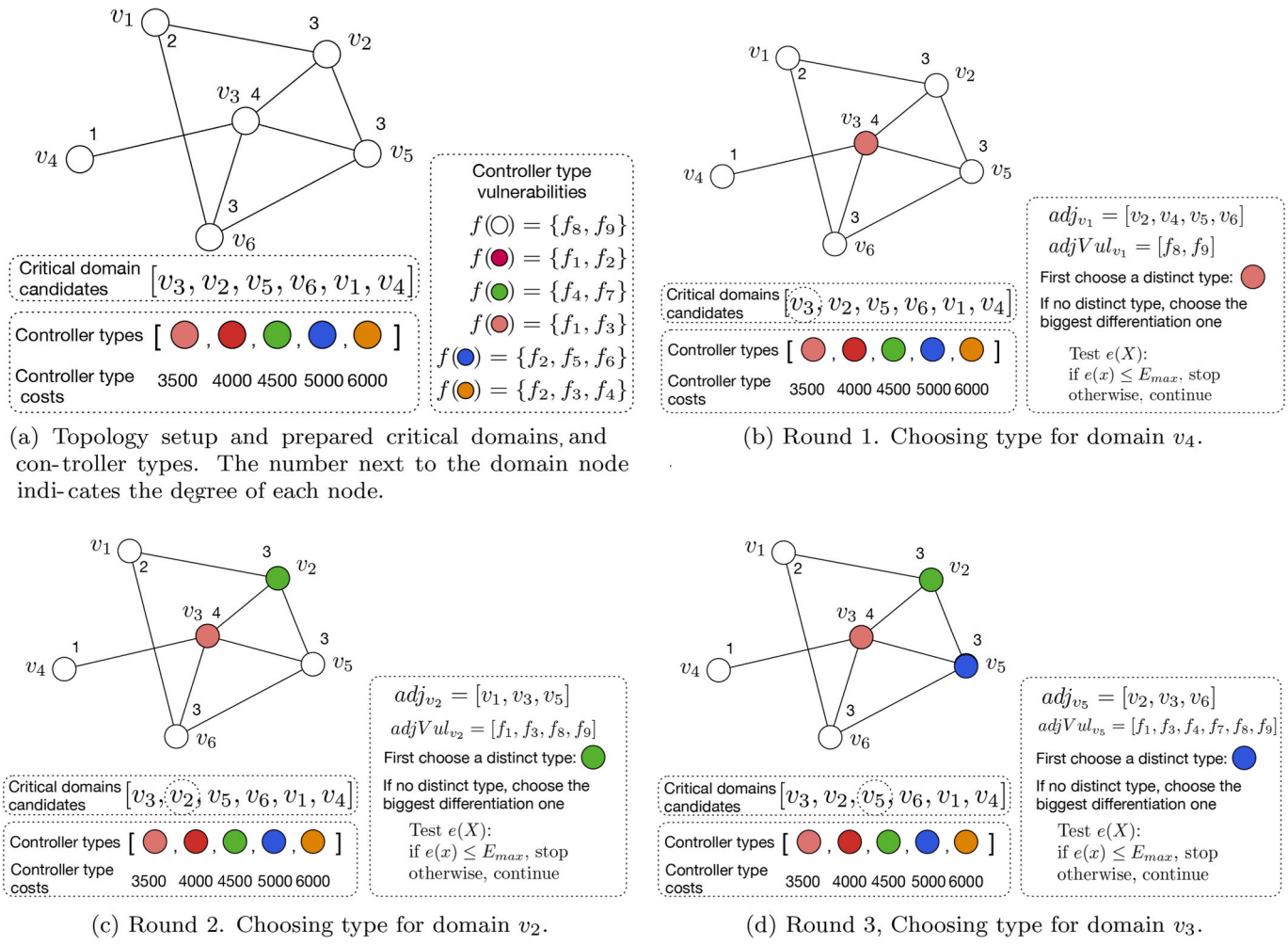
Algorithm 2: The Baguette algorithm.

Input: $\mathcal{G}$, $\mathcal{S}$
Output: X
/* Critical domain candidates Preparation. */
1 $N\leftarrow|\mathcal{V}|$;
2 $Matrix\leftarrow$ `getAdjacencyMatrix`($\mathcal{G}$);
3 $D\leftarrow\emptyset$;
4 $X_{min}\leftarrow X$, $expect_{min}\leftarrow\infty$;
5 **for** $i\leftarrow 1$ **to** N **do**
6 $\quad D\leftarrow D\cup(\sum_{i_1}^{N} Matrix_{i,i_1})$;
7 Generating vector $\mathcal{V}'=\{v_i, i\in[1,N]\}$ by sorting the degree of each domain D_i in descending order;
/* Preparing controller types */
8 $C\leftarrow$`getControllerCost`($\mathcal{V}$);
9 Generating vector $\mathcal{S}'=\{s_j, j\in[1,|\mathcal{S}|]\}$ by sorting the cost of each type C_j in ascending order;
/* Mapping controller types */
10 **for** $v_i\in\mathcal{V}'$ **do**
11 $\quad Adj\leftarrow$ `getAdjacent`(v_i);
12 $\quad\mathcal{F}'\leftarrow\emptyset$, $S_1\leftarrow\emptyset$;
13 $\quad$**for** $v_{i_1}\in Adj$ **do**
14 $\quad\quad\mathcal{F}'\leftarrow\mathcal{F}'\cup$ `getVulnerability`(v_{i_1});
15 $\quad\quad S_1\leftarrow S_1\cup$ `getControllerType`(v_{i_1});
16 $\quad\mathcal{S}''\leftarrow\mathcal{S}'\setminus S_1$;
/* If all versions are used. */
17 $\quad$**if** $S''=\emptyset$ **then**
18 $\quad\quad S''\leftarrow S'$;
19 $\quad min\_ratio\leftarrow 1$, $j_{min}\leftarrow 1$;
20 $\quad$**for** $s_j\in\mathcal{S}''$ **do**
21 $\quad\quad ratio\leftarrow\frac{\mathcal{F}_j\cap\mathcal{F}'}{\mathcal{F}'}$;
22 $\quad\quad$**if** $min\_ratio>ratio$ **then**
23 $\quad\quad\quad min\_ratio\leftarrow ratio$, $j_{min}\leftarrow j$;
24 $\quad x_{i,j_{min}}\leftarrow 1$;
25 $\quad expect\leftarrow$`compromiseExpectation`(X);
26 $\quad$**if** $expect\le E^{max}$ **then**
27 $\quad\quad$**break**;
28 $\quad$**else**
29 $\quad\quad$**if** $expect_{min}>expect$ **then**
30 $\quad\quad\quad X_{min}\leftarrow X$;
31 $\quad\quad\quad expect_{min}\leftarrow expect$;
32 **return** X_{min}

The controller type selection procedure has the following 6 subroutines. i) Get all types of the current selected domain's adjacent domains. ii) Calculate a new array of controller types, by removing the adjacent types as $\mathcal{S}'$. iii) Get the common vulnerabilities of the adjacent domains as $\mathcal{F}'$. iv) For each type s_j in $\mathcal{S}'$, calculate the *vulnerability variation* that is the percentage of the number of common vulnerabilities between $\mathcal{F}_j$ and $\mathcal{F}'$, over the number of vulnerabilities of $\mathcal{F}_j$. v) Map the type with the lowest vulnerability variation calculated in Step iv) to the critical domain. vi) Calculate the compromising expectation of the network, and if the value satisfies the requirement, stop. Otherwise, pick the next critical domain, and go to Step i).

5.3. Analysis of baguette

The Baguette algorithm is detailed in Algorithm 2. Lines 1–7 generate the order of domains whose controller types to be changed. The time complexity of this sub-procedure is $O(|N|\log|N|)$ as it leverages sorting. In Lines 2–6, it first generates the degree vector D based on the adjacency matrix of the network, and it then sorts the domains based on the degree in descending order. Lines 8–9 prepare the different controller types by sorting the types based on their costs. The time complexity of preparing controller types candidates is also $O(|\mathcal{S}|\log|\mathcal{S}|)$ because of the use of sorting. Lines 10–31 map types to domains. The key idea is to



(a) Topology setup and prepared critical domains, and con-troller types. The number next to the domain node indi-cates the degree of each node.

(b) Round 1. Choosing type for domain v_4.

(c) Round 2. Choosing type for domain v_2.

(d) Round 3, Choosing type for domain v_3.

Fig. 6. The Baguette algorithm demonstration.

choose the type that has the biggest vulnerability differentiation of all adjacent domains. The cost of the deployment is considered when multiple type candidates have the same vulnerability variation, and we choose the one with a smaller cost. Lines 11–15 get all the types and vulnerabilities of adjacent domains. Lines 18–23 calculate the ratio of *common vulnerabilities* of the current type and the types of adjacent domains' controllers, to the *total vulnerabilities* of the type of adjacent domains' controllers. We choose the type with the minimum ratio. Finally, we calculate the compromise expectation based on the current mapping X. If the mapping satisfies the requirement, the algorithm stops. Otherwise, it keeps mapping the next domain until all domains are processed. The compromise expectation is calculated with Equation (3), and domain compromised ratios are calculated with Algorithm 1 that traverse the network. Hence, the time complexity of calculating the compromise expectation is $O((|\mathcal{V}| + |\mathcal{E}|)|\mathcal{V}||\mathcal{A}|)$, and the time complexity of Algorithm 2 is $O((|\mathcal{V}| + |\mathcal{E}|)|\mathcal{V}|^2|\mathcal{A}|)$, which is a polynomial time. Online algorithms are not necessary for the SCCD problem as it is in the network deployment procedure and will not influence the running performance. Hence, we believe Baguette is adequate to solve the problem.

6. Simulation

In this section, we present the simulation of Baguette to evaluate its performance. We first introduce the simulation setup information and comparison algorithms. We then compare the performances of different algorithms under various real-world topologies and evaluate Baguette's stability under different sizes of topologies. Simulations results show that the Baguette algorithm achieves near-optimal performance under non-full mesh topologies and performs stably under 80% of the topologies. We do not evaluate the time consumption due to the non-polynomial time complexities of optimal and SecureMost solutions.

6.1. Simulation setup

We first use Arpanet (Topology Zoo), GlobalCenter (The Center, LLC), and HEAnet (HEAnet) from Topology Zoo (Knight et al., 2011) to conduct the simulation. Topology Zoo is a collection of 262 real-world backbone network topologies, and each topology is provided with a `gml` file. We use a popular python graph library `python-igraph` (Csardi Nepuszet al., 2006) to read `gml` files. Both Arpanet and GlobalCenter have 9 nodes, and GlobalCenter is a full-meshed network with 36 links. HEAnet has 7 nodes. In the simulation, each node in the topology represents a domain, which deployed with one controller. There are 4 controller types in total. Originally, controllers of each domain use type s_1. Each controller type contains multiple vulnerabilities in the simulation, we randomly generate vulnerabilities for each type, and the number of vulnerabilities is random in the range of (0, 6). Besides, we have surveyed many commercial SDN controllers (Cisco Systems et al., Cisc; Huawei Technologies Co.; Juniper Networks et al., N; TelefonaktiebolagetEr), and the prices of popular commercial SDN controllers range from $1000 to $5000. Thus, we randomly generate a cost for

every controller type in the range of (1000, 5000). We generate attacks by calculating the subset of total vulnerability set $\mathcal{F}$, and the total number of attacks is $2^6 - 1$ (the empty set is removed). We use Python to implement the simulation.

6.2. Compared algorithms

We compare the following 4 algorithms. The reason we do not compare Baguette with existing GCP solutions is that the GCP solutions are insufficient to solve Baguette (see Section 5.1).

- Legacy: this is the default controller type deployment where all controllers use the default type.
- Optimal: this is the optimal solution of the SCCD problem, which minimizes the overall controller deployment cost under certain network security requirements. Since Equation (2) requires graph traversal, common ILP solvers like GUROBI (Gurobi) cannot help. We first pre-generate possible mappings and then choose the best one to reduce the calculation time.
- SecurityMost: this algorithm calculates the controller deployment scheme with the minimum compromise expectation. We also pre-generate all mappings and choose the one with the minimum compromise expectation.
- Baguette: this algorithm is shown in Algorithm 2.

6.3. Simulation results

In this subsection, we first use three smaller topologies to show that Baguette achieves near-optimal performance. We then evaluate the performance on different topologies. We do not conduct experiments on bigger topologies (number of nodes greater than 10) due to the NP-hardness of Optimal and SecurityMost algorithms.

6.3.1. Algorithm comparison

We compare the security and overall cost performances of Legacy, Optimal, Baguette, and SecurityMost algorithms under different security (compromise expectation) requirements ranging from 0.1 to 0.9. We do not use larger topology since they are time-consuming due to the complexity of Optimal and SecurityMost. While Baguette is tolerable to all scale of topologies thanks to the polynomial time complexity.

Fig. 7 shows the security and cost performances of the 4 algorithms under the 3 topologies. In a nutshell, Baguette achieves up to **12.6x security enhancement** compared with the legacy setup and **as low as 11.1% cost of SecurityMost**. Legacy has no security guarantee, and SecurityMost can stop most attacks while introduces an exorbitant cost. In Fig. 7a–c, Optimal satisfies all the security requirements in each tested topology. **Baguette satisfies 100% tests on HEAnet and approximately 80% tests on Arpanet**. The results also indicate that Baguette does not perform well on GlobalCenter when the security requirement is strict (low compromise expectation). Fig. 7c shows that Baguette fails to satisfy the security requirement when the required compromise expectation is below 0.7. This is because Baguette prefers to choose nodes with the most number of degrees to replace controller types, but GlobalCenter network is a full-meshed network, and the number of degrees of each node is the same. To this end, Baguette can only sequentially replace types of nodes one by one until all 4 types of SDN controllers are utilized.

To better understand the performance between Optimal and Baguette, we define the *Performance Likelihood (PL)* metric. The PL metrics measures the portion of the performance difference between Baguette and the optimal solution over the total performance improvements (*e.g.*, from no security enhancements to the most secure solution and from the most secure solution that has the maximum costs to the no replacement solution). It is the absolute difference between two algorithms over the difference of the minimum and maximum performances, as follows.

$$\mathrm{PL} = \frac{|p_{A_1} - p_{A_2}|}{p_{\max} - p_{\min}}, \tag{11}$$

where p_{A_1} and p_{A_2} are the performances of algorithms A_1 and A_2. $p_{\max}$ and $p_{\min}$ are the maximum and minimum performances. $p_{A_1}, p_{A_2}, p_{\max}$ and $p_{\min}$ are at the same compromise expectation requirement. For example, the security PL of Baguette and Optimal is represented as the absolute difference of compromise expectations of Baguette and Optimal over the absolute difference of compromise expectations of Legacy and SecurityMost. A PL value is a floating number between 0 and 1. When the performances are similar, the value approaches to 0.

Fig. 7 shows that the average security PL is 0.12 in all tests, and Baguette and Optimal have the same security performance in almost half (46.7%) of all the tests. In Fig. 7e, f, and d, the cost PL between Optimal and Baguette is 0.11 of all tests, and Baguette and Optimal have the same cost performance in 42.8% of all the tests. We can conclude that **Baguette achieves near-optimal performance** under non-full mesh topologies. Although both Arpanet and GlobalCenter have 9 nodes (domains), their security and cost performances are different. This is because different topological structures contribute to different SDN controller deployment schemes that employ different types of controllers.

6.3.2. Tests on different topology sizes

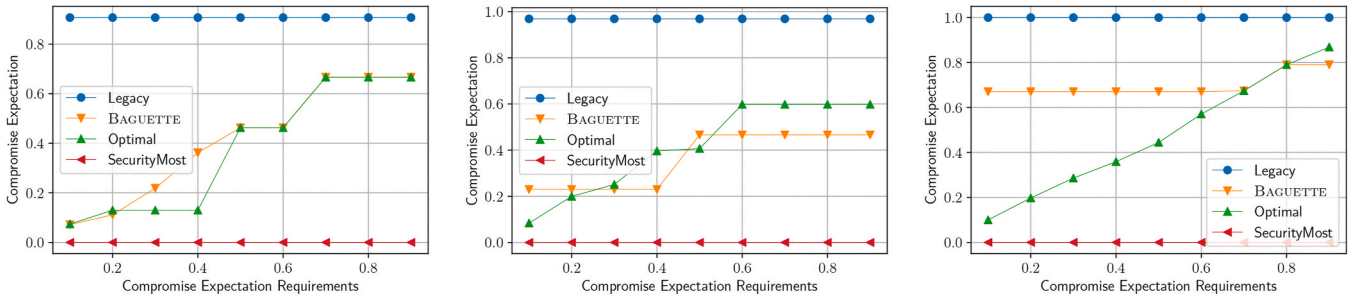
We run Baguette on 69 topologies whose numbers of nodes ranging from 5 to 92. We use different security requirements to conduct the simulations. We run 20 trials for each topology and security requirement ratio combination. Each trial has distinct versions and vulnerabilities setup since they are generated randomly.

Fig. 8 shows the experimental results, and Fig. 8a depicts the average compromise expectation for all test topologies under security requirements ranging from 0.1 to 0.9. **Baguette satisfies 100% of the average security performance** under security requirements 0.5 and above in all topologies, 85% of the topologies under security requirement 0.4, 50% of the topologies under security requirement 0.3, 40% for security requirement 0.2, and 25% for security requirement 0.1. There are some "hills" in the figure (*e.g.*, when the number of nodes is 30, 36, 40, 59, and 88). We analyze the corresponding topologies and find that these topologies are *star topologies* (or variations, *e.g.*, several connected star networks). Since Baguette tends to replace types for the "critical" nodes who have the most degrees, it will first replace the controller type for the "hub" nodes in the network. However, we find that the number of "hub" nodes is usually smaller than 5. Hence, after replacing types for all "hubs" in the network, further replacements of other nodes have little benefits on the security improvement.

Fig. 8b shows the randomly chosen topology sizes due to space limitation. The results indicate that Baguette performs stably on 80% of the cases with under 0.1 standard deviations. Hence, we believe **Baguette achieves stable performance**. As shown in Fig. 8c, the average cost of 0.5 security requirement is only half of that of security requirement 0.1 setup, and stricter security requirements result in a higher cost. Standard deviations of costs are bigger on large topologies because they require more network domains to change the type of controllers, but Baguette can reduce the cost of distinct controller type deployment for each domain. For example, in the Viatel topology (Metter et al., 2015) (with 92 nodes), even the security requirement 0.1 can reduce the cost to only 20% of completely using distinct controller types for each domain.

7. Related works

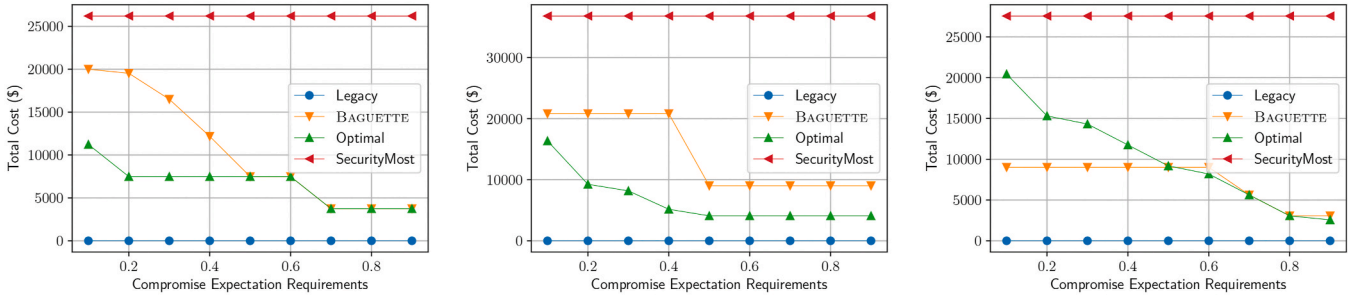
To the best of our knowledge, this is the first work that focuses on leveraging the benefit of node heterogeneity, which uses controller types to mitigate attacks propagating to other domains. However, existing Multi-domain SDN work focusing on multi-domain controller



(a) The security comparison of 4 algorithms under Heanet (non-full mesh topology). The lower the better.

(b) The security comparison of 4 algorithms under Arpanet (non-full mesh topology). The lower the better.

(c) The security comparison of 4 algorithms under GlobalCenter (full mesh topology). The lower the better.



(d) The cost comparison of 4 algorithms under Heanet (non-full mesh topology). The lower the better.

(e) The cost comparison of 4 algorithms under Arpanet (non-full mesh topology). The lower the better.

(f) The cost comparison of 4 algorithms under GlobalCenter (full mesh topology). The lower the better.

Fig. 7. Legacy, Optimal, BAGUETTE, and SecurityMost performances under different security requirements. BAGUETTE achieves near-optimal performance under non-full mesh topologies.

communication, traffic engineering, and controller placement. In this section, we study these multi-domain SDN work and SDN oriented security researches.

7.1. Multi-domain SDN controllers communications

Onix (Koponen et al., 2010) is the first production-level distributed SDN controller. Each Onix replica controls many (a domain of) SDN switches and generates the topology of the domain. Onix replicas then share the local domain topology information to build a global view of the network across all controllers. ElastiCon (Dixit et al., 2013) addresses the load unbalances problem of controllers in the multi-domain scenario as the fixed "controller – switch" mapping cannot adjust to the changing traffic load. ElastiCon employs an elastic controller pool with a distributed data store that collects the network status. The controller pool can dynamically grow or shrink based on the traffic condition, and it manages the switch migration to guarantee the liveness and safety.

7.2. Multi-domain SDN traffic engineering and resiliency

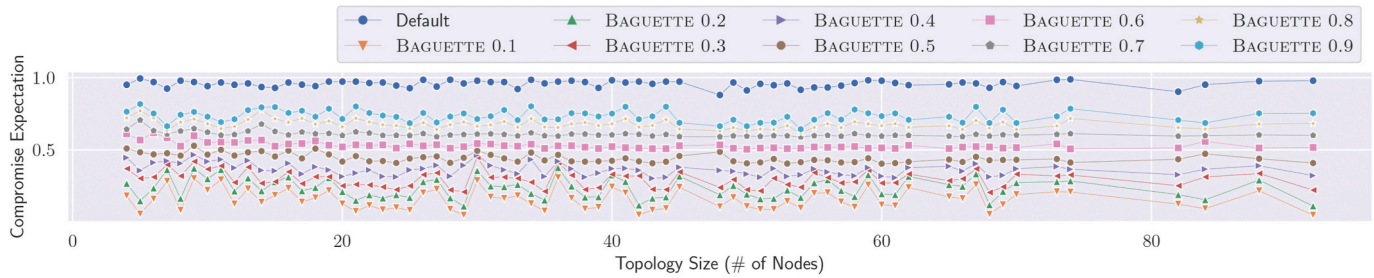
B4 (Jain et al., 2013), SWAN (Hong et al., 2013), and RetroFlow (Guo et al., 2019) are SD-WAN schemes, which leverages the multi-domain SDN. B4 and SWAN are built for traffic engineering. B4 can achieve near 100% link utilization with the global decision and fine-grained traffic class control that prioritizes the critical traffic to be successfully transmitted and uses the less important traffic to fill the "gaps". SWAN can reach 70% link utilization without blocking critical control messages by reserving the bandwidth for the critical control messages. RetroFlow considers the resilience of the multi-domain SDN network, and when one controller fails, how to remap the SDN switches to existing controllers in other domains with the minimal performance overhead.

7.3. Controller placement in multi-domain SDN

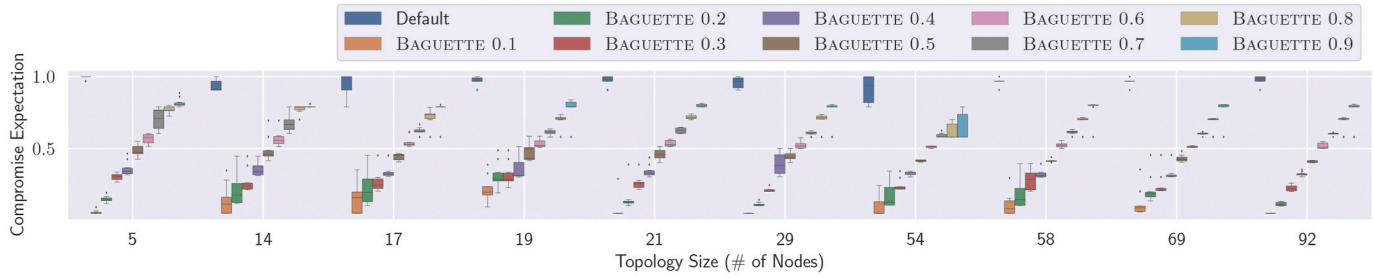
The key to controller placement is to partition the network and find the optimal mapping between switches and controllers under certain conditions. Heller et al. (2012) first propose the controller placement problem and answers the questions that given a network topology, how many controllers are needed, and how to place them. They leverage the "switch – controller" latency as the key metric and formulate the problem as an ILP problem. Xu et al. (2019) propose SDN switch migration schemes to achieve load balance among SDN controllers with small migration costs because loads of controllers may become uneven as time goes by.

7.4. Attack mitigation in SDN

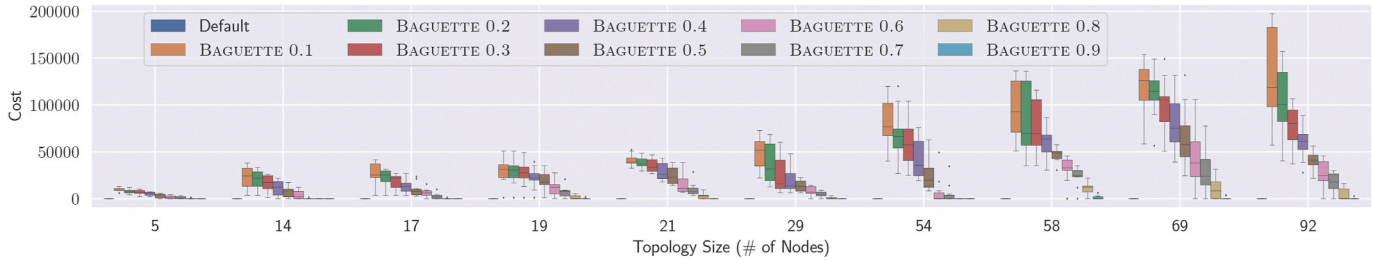
Many SDN security researches have been conducted. DASON (Vizarreta et al., 2020) studies bugs in open source SDN controller systems and outages resulted from the bugs. CLÉ (Feng et al., 2019) proposes to virtualize the SDN devices as security middleboxes in the hybrid SDN deployment to mitigate attacks. Xu et al. (2017) present the flow table overflow attack in SDN, and they identify the attack pattern and then use the token bucket model to mitigate the attacks. SAFETY (Kumar et al., 2018) is a novel entropy-based TCP flood attack detection system that targets on "SYN-flood" from the data plane to the control plane. Antidose (Simpson et al., 2018) proposes the Autonomous System (AS) level Distributed Denial of Service (DDoS) filtering mechanism to avoid ASes being exposed to additional attacks. However, none of the work considers the relationship between controller attacks and controller types as well as multi-domain attack propagations.



(a) Average compromise expectations for different topologies under different security requirements.



(b) Compromising expectation boxplot under different topologies with different security requirements. The lower the better.



(c) Cost boxplot under different topologies with different security requirements. The lower the better.

Fig. 8. Baguette performance under different size of topologies and different security requirements.

8. Conclusion and future work

In this paper, we have identified the SCCD problem in multi-domain commercial SDN deployment, which aims to achieve the security requirement with a minimum cost. We have proved the NP-hardness of the SCCD problem and have proposed the Baguette algorithm to efficiently solve it. Baguette judiciously chooses critical domains and deploys selected types of SDN controllers for them to mitigate the attack propagations. We have conducted simulations using real-world topologies to evaluate Baguette. Experimental results have shown that Baguette can stably achieve near-optimal performance under non-full mesh topologies with up to 12.6x security enhancement and down to 11.1% cost of the most secure solution. By presenting Baguette, we hope it can motivate the network community to consider and utilize the benefits of node heterogeneity.

To advance the SCCD problem and the Baguette algorithm, our future work consists of three parts: i) improving Baguette for achieving better performance on full-mesh topologies and spine-leaf topologies that have identical degrees of nodes, ii) identify different scenarios and applications that can apply Baguette to solve (*e.g.*, considering performance overhead introduced by different types of controllers), and iii) validating controller attack and attack propagations on real-world network testbeds.

Credit author statement

Wendi Feng: Conceptualization, Methodology, Software, Validation, Writing- original draft. Chuanchang Liu: Resources, Project administration. Bo Cheng: Supervision. Junliang Chen: Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1003804, Natural Science Foundation of China under Grant 61921003, 61972043. We thank Prof. Zhi-Li Zhang, Dr. Zehua Guo, and Dr. Gang Wang for their valuable comments and recommendations. We also thank the anonymous reviewers for their constructive comments. Wendi Feng gratefully acknowledge the financial support from China Scholarship Council.

References

Arashloo, M.T., Koral, Y., Greenberg, M., Rexford, J., Walker, D., 2016. Snap: stateful network-wide abstractions for packet processing. In: Proceedings of the 2016 ACM SIGCOMM Conference, pp. 29–43.

Bilge, L., Dumitraş, T., 2012. Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12. ACM, Raleigh, North Carolina, USA, pp. 833–844.

Boccardi, F., Heath, R.W., Lozano, A., Marzetta, T.L., Popovski, P., 2014. Five disruptive technology directions for 5g. IEEE Commun. Mag. 52 (2), 74–80.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. ACM, pp. 13–16.

Chica, J.C.C., Imbachi, J.C., Vega, J.F.B., 2020. Security in sdn: a comprehensive survey. J. Netw. Comput. Appl. 159, 102595. https://doi.org/10.1016/j.jnca.2020.102595.

Choi, S., Burkov, B., Eckert, A., Fang, T., Kazemkhani, S., Sherwood, R., Zhang, Y., Zeng, H., 2018. Fboss: building switch software at scale. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. SIGCOMM '18, ACM, New York, NY, USA, pp. 342–356.

Cisco Systems, Inc.. Cisco open SDN controller. https://www.cisco.com/c/en/us/products/cloud-systems-management/open-sdn-controller/index.html.

Csardi, G., Nepusz, T., et al., 2006. The igraph software package for complex network research. InterJournal, Complex Syst. 1695 (5), 1–9.

Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R., 2013. Towards an elastic distributed sdn controller. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. Association for Computing Machinery, pp. 7–12. https://doi.org/10.1145/2491185.2491193.

Feng, W., Zhang, Z.-L., Liu, C., Chen, J., 2019. Clé: enhancing security with programmable dataplane enabled hybrid SDN. In: Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '19. Association for Computing Machinery, pp. 76–77. https://doi.org/10.1145/3360468.3368185.

Fuetsch, A.. The future of AT&T 5G. https://about.att.com/innovationblog/2020/09/future_att_5g.html.

Guo, Z., Feng, W., Liu, S., Jiang, W., Xu, Y., Zhang, Z., 2019. Retroflow: maintaining control resiliency and flow programmability for software-defined wans. In: Proceedings of the International Symposium on Quality of Service, IWQoS 2019, Phoenix, AZ, USA, June 24-25, 2019., pp. 1:1–1:10.

Gurobi. Gurobi optimizer, gurobi. http://www.gurobi.com.

Hayes, B., 2008. Cloud computing. Commun. ACM 51 (7), 9–11.

HEAnet, HEAnet - Ireland's national research & education network, https://www.heanet.ie.

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. SIGCOMM Comput. Commun. Rev. 42 (4), 473–478. https://doi.org/10.1145/2377677.2377767.

Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., Wattenhofer, R., 2013. Achieving high utilization with software-driven wan. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13. Association for Computing Machinery, pp. 15–26. https://doi.org/10.1145/2486001.2486012.

Huawei Technologies Co., Ltd.. Agile controller. https://e.huawei.com/uk/products/enterprise-networking/sdn-controller/agile-controller.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A., 2013. B4: experience with a globally-deployed software defined wan. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13. ACM, New York, NY, USA, pp. 3–14.

Jensen, T.R., Toft, B., 2011. Graph Coloring Problems, vol. 39. John Wiley & Sons.

Juniper Networks, Inc., NorthStar controller, https://www.juniper.net/us/en/products-services/sdn/northstar-network-controller/.

Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M., 2011. The internet topology zoo. IEEE J. Sel. Area. Commun. 29 (9), 1765–1775.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., NEC, Y.I., NEC, H.I., NEC, T.H., Shenker, S., 2010. Onix: a distributed control platform for large-scale production networks. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, Berkeley, CA, USA, pp. 351–364.

Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103 (1), 14–76.

Kumar, P., Tripathi, M., Nehra, A., Conti, M., Lal, C., 2018. Safety: early detection and mitigation of tcp syn flood utilizing entropy in sdn. IEEE Trans. Netw. Serv. Manag. 15 (4), 1545–1559. https://doi.org/10.1109/TNSM.2018.2861741.

Li, Z., Zou, D., Xu, S., Jin, H., Qi, H., Hu, J., 2016. Vulpecker: an automated vulnerability detection system based on code similarity analysis. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC '16. ACM, pp. 201–213.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. Comput. Commun. Rev. 38 (2), 69–74.

Metter, C., Gebert, S., Lange, S., Zinner, T., Tran-Gia, P., Jarschel, M., 2015. Investigating the impact of network topology on the processing times of sdn controllers. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1214–1219.

Phemius, K., Bouet, M., Leguay, J., 2014. DISCO: distributed multi-domain SDN controllers. In: 2014 IEEE Network Operations and Management Symposium. (NOMS), pp. 1–4.

Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. Sdn security: a survey. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–7.

Scott-Hayward, S., Natarajan, S., Sezer, S., 2016. A survey of security in software defined networks. IEEE Commun. Surv. Tutorials 18 (1), 623–654.

Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: vision and challenges. IEEE Internet Things J. 3 (5), 637–646.

Simpson, S., Shirazi, S.N., Marnerides, A., Jouet, S., Pezaros, D., Hutchison, D., 2018. An inter-domain collaboration scheme to remedy ddos attacks in computer networks. IEEE Trans. Netw. Serv. Manag. 15 (3), 879–893. https://doi.org/10.1109/TNSM.2018.2828938.

Song, S., Park, H., Choi, B., Choi, T., Zhu, H., 2017. Control path management framework for enhancing software-defined network (SDN) reliability. IEEE Trans. Netw. Serv. Manag. 14 (2), 302–316.

Telefonaktiebolaget LM Ericsson, Ericsson Cloud SDN, https://www.ericsson.com/en/portfolio/digital-services/cloud-infrastructure/cloud-sdn.

Teo, Z., Birman, K., Renesse, R.V., 2016. Experience with 3 sdn controllers in an enterprise setting. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop. DSN-W, pp. 97–104.

The Center, LLC, The global center for nonprofit excellence, https://www.theglobalcenter.net.

Topology Zoo, Arpanet 1970-6, http://topology-zoo.org/maps/Arpanet19706.jpg.

Vizarreta, P., Trivedi, K., Mendiratta, V., Kellerer, W., Machuca, C.M., 2020. DASON: Dependability Assessment Framework for Imperfect Distributed SDN Implementations. IEEE Transactions on Network and Service Management, p. 1, 1.

Xu, T., Gao, D., Dong, P., Foh, C.H., Zhang, H., 2017. Mitigating the table-overflow attack in software-defined networking. IEEE Trans. Netw. Serv. Manag. 14 (4), 1086–1097. https://doi.org/10.1109/TNSM.2017.2758796.

Xu, Y., Cello, M., Wang, I., Walid, A., Wilfong, G., Wen, C.H., Marchese, M., Chao, H.J., 2019. Dynamic switch migration in distributed software-defined networks to achieve controller load balance. IEEE J. Sel. Area. Commun. 37 (3), 515–529.

Yap, K.-K., Motiwala, M., Rahe, J., Padgett, S., Holliman, M., Baldus, G., Hines, M., Kim, T., Narayanan, A., Jain, A., Lin, V., Rice, C., Rogan, B., Singh, A., Tanaka, B., Verma, M., Sood, P., Tariq, M., Tierney, M., Trumic, D., Valancius, V., Ying, C., Kallahalla, M., Koley, B., Vahdat, A., 2017. Taking the edge off with espresso: scale, reliability and programmability for global internet peering. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17. Association for Computing Machinery, pp. 432–445. https://doi.org/10.1145/3098822.3098854.

Wendi Feng is a PhD candidate with Professor Junliang Chen at State Key Laboratory of Network and Switching Technology at Beijing University of Posts and Telecommunications. He is co-advised by Prof. Zhi-Li Zhang at the University of Minnesota - Twin Cities. His research interests include mobile computing, cloud computing, computer networking, software-defined network, and network function virtualization.

Chuanchang Liu is currently an Associate Professor with the State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include mobile device security, cloud computing, and oriented-service computing.

Bo Cheng is currently a Professor and vice director with the State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include mobile device security, cloud computing, internet of things and big data analysis, network service and intelligence.

Junliang Chen is currently a Professor and the Academic Leader with the State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He is a member of the Chinese Academy of Science and the Chinese Academy of Engineering, and a fellow of the China Computer Federation. His current research interests include service-oriented computing and service generation system.