

考虑拜占庭属性的 SDN 安全控制器多目标优化部署方案

王涛, 陈鸿昶

(信息工程大学, 河南 郑州 450001)

摘要: 通过赋予软件定义网络分布式控制平面拜占庭属性可以有效提高其安全性。在实现拜占庭属性过程中, 控制器部署的数量、位置, 以及交换机与控制器之间的连接关系会直接影响全局网络关键性能指标。为此, 提出了一种考虑拜占庭属性的 SDN 安全控制器多目标优化部署方案。首先, 构建了综合考量交互时延、同步时延、负载差异程度和控制器部署数量等优化指标的拜占庭控制器部署问题 (MOSBCPP) 模型; 然后, 针对该模型个性化设计了包括控制器部署策略初始化函数、变异函数, 快速非支配排序函数及精英策略选择函数等在内的 NASG-II 求解算法。相关仿真结果表明, 该部署方案能够在有效降低交互时延、同步时延、负载差异程度和控制器部署数量等性能指标的同时提高控制平面安全性。

关键词: 软件定义网络安全; 分布式控制平面; 拜占庭属性; 控制器部署; 多目标优化

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2021060

Multi-objective optimization placement strategy for SDN security controller considering Byzantine attributes

WANG Tao, CHEN Hongchang

Information Engineering University, Zhengzhou 450001, China

Abstract: By giving the software defined network distributed control plane Byzantine attributes, its security can be effectively improved. In the process of realizing Byzantine attributes, the number and location of controllers, and the connection relationship between switches and controllers can directly affect the key network performance. Therefore, a controller multi-objective optimization placement strategy for SDN security controllers considering Byzantine attributes was proposed. Firstly, a Byzantine controller placement problem (MOSBCPP) model that comprehensively considered interaction delay, synchronization delay, load difference and the number of controllers was constructed. Then, a solution algorithm based on NASG-II was designed for this model, which included the initialization function, the mutation function, the fast non-dominated sorting function and the elite strategy selection function. Simulation

收稿日期: 2020-12-11; **修回日期:** 2021-03-01

通信作者: 王涛, wangtaogenuine@163.com

基金项目: 国家重点研发计划 (2018YFB0804004); 国家自然科学基金 (62072467, 61521003)

Foundation Items: The National Key R&D Program of China (2018YFB0804004), The National Natural Science Foundation of China (62072467, 61521003)

论文引用格式: 王涛, 陈鸿昶. 考虑拜占庭属性的 SDN 安全控制器多目标优化部署方案[J]. 网络与信息安全学报, 2021, 7(3): 72-84.

WANG T, CHEN H C. Multi-objective optimization placement strategy for SDN security controller considering Byzantine attributes[J]. Chinese Journal of Network and Information Security, 2021, 7(3): 72-84.

results show that this strategy can effectively reduce interaction delay, synchronization delay, load difference and the number of controllers, while improving control plane security.

Keywords: software defined network security, distributed control plane, Byzantine attributes, controller placement, multi-objective optimization

1 引言

软件定义网络 (SDN, software defined network) [1] 将传统网络紧耦合的控制元素与转发元素分离, 通过逻辑集中的可编程控制器实现统一的网络资源管理与编排, 而转发平面的交换机单纯负责执行控制器下发的高级网络策略。这种解耦式的架构为管控灵活性和可编程性提供了便利, 加速了网络应用创新过程。

随着 SDN 技术在更大规模网络和更多场景下应用, 控制平面所面临的可扩展性需求逐渐增大, 控制平面由最初的单点式控制器向分布式控制器转变 (如 Hyperflow [2] 等)。分布式控制器能够有效改善控制平面性能和可扩展性, 很快便成为 SDN 控制平面的主流配置。但随着网络安全形势日益严峻, SDN 分布式控制器面临的安全威胁风险不断升高, 而分布式技术本身并不能增加控制平面的安全性, 这使各场景下 SDN 的安全问题较为突出。而且, 现有的一些“补丁式”SDN 安全方案 [3] 具有防御滞后性, 这更加剧了 SDN 控制平面的安全风险。为了解决这一问题, 研究人员开始尝试赋予分布式控制平面拜占庭安全属性 [4]。为了实现拜占庭安全属性, 每个交换机均需根据其安全 (容错) 需求度选择网络拓扑图中的多个控制器作为参考, 这些控制器通过运行拜占庭协议协商下发流规则。通过这种方式可以有效识别因少数控制器受到恶意攻击而被篡改的流规则 (不依赖于特定攻击特征, 从而摆脱补丁式安全方案的防御滞后性), 提高控制平面的安全性。

考虑到拜占庭控制平面结构的复杂性, 在实现上述拜占庭属性过程中, 如何设计控制器部署策略优化全局网络性能指标已成为待解决的关键问题之一。一般而言, 全局网络的开销、时延、负载均衡等性能指标直接取决于控制器的数量、位置, 以及交换机与控制器之间的连接关系。因此, 如果在部署控制器 (即确定控制器数量、位置, 以及交换机与控制器之间的连接关系) 时未

统筹考量开销、时延、负载均衡等综合网络性能指标, 就容易使控制器部署策略在多种性能指标优化方面“顾此失彼”, 造成控制器部署策略在部分指标方面短板效应显著, 降低控制器部署策略的实际参考价值。为了便于直观理解上述过程, 本文以如图 1 所示的控制器部署策略拓扑为例, 概述一般场景下不同控制器部署策略对全局网络性能指标的具体影响。为了便于描述, 假设示例中交换机 A、B 和 C 的负载分别为 l 、 l 和 $2l$ ($l > 0$); 交换机链路 AB、BC 和 AC 的传输时延分别为 t_1 、 t_2 和 t_3 ($t_1 < t_2 < t_3$); 控制器部署数量固定为 2; 如果交换机位置上部署了控制器则部署策略 (A、B、C) 对应位置元素赋值为 1, 否则为 0。在此基础上, 具体定量讨论不同控制器部署策略 (1,1,0) 和 (1,0,1) 在时延和负载均衡方面的性能差异, 如表 1 所示, 考虑到控制器与交换机一般采用最短路径进行连接, 那么上述两种策略的时延合计和负载差异指标分别为 (t_2, t_1) 和 $(2l, 0)$ 。由于 $t_1 < t_2$ 且 $l > 0$, 故不同控制器部署策略 (1,1,0) 和 (1,0,1) 会对时延和负载均衡指标产生直接影响, 且策略 (1,1,0) 各性能指标均劣于 (1,0,1)。因此, 合理确定控制器部署策略对提高网络性能指标具有重要意义。

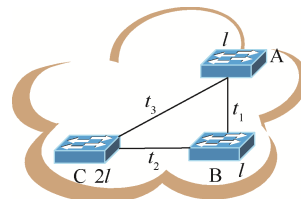


图 1 控制器部署策略拓扑示例

Figure 1 Example of controller deployment strategies topology

表 1 不同控制器部署策略下网络性能指标对比
Table 1 Comparison of network performance indicators under different controller deployment strategies

部署策略	控制器位置	管理的交换机	控制路径	控制时延	负载	时延合计	负载差异
(1,1,0)	A	A	A→A	0	l	t_2	$2l$
	B	B, C	B→B, C→B	$0, t_2$	$3l$		
(1,0,1)	A	A, B	A→A, B→A	$0, t_1$	$2l$	t_1	0
	C	C	C→C	0	$2l$		

在上述分析基础上,考虑到拜占庭控制平面结构的复杂性(涉及交换机与控制器之间交互、控制器之间同步、控制器之间负载均衡等方面)以及控制器数量不确定性因素等,使具有拜占庭属性的SDN安全控制器部署问题更加复杂,相关综合考虑多种性能优化目标的控制器部署研究也更少。为此,本文提出了一种考虑拜占庭属性的SDN安全控制器多目标部署方案,主要贡献总结如下。

(1) 首次构建了综合考量交互时延、同步时延、负载差异程度和控制器部署数量等优化指标的拜占庭控制器部署问题(MOSBCPP, multi-objective SDN Byzantine controller placement problem)模型。

(2) 在第二代非支配排序遗传算法(NSGA-II, non-dominated sorted genetic algorithm-II)思想的基础上,提出了包括基于非确定交换机序列的控制器部署策略初始化函数、基于拜占庭备选域内随机约束性映射的控制器部署策略变异函数、基于双向支配属性的快速非支配排序函数和基于非支配临界等级与拥挤度指标的精英策略选择函数等在内的MOSBCPP模型求解算法。

(3) 通过仿真验证了MOSBCPP模型在交互时延、同步时延、负载差异程度、控制器部署数量和安全性能等指标方面的有效性。

2 相关工作

目前对于软件定义网络控制器部署问题可以按照部署前控制器数量是否确定分为自适应部署方案和非自适应部署方案。其中,现有研究中大多数控制器部署方案均属于非自适应部署方案,这类方案的研究目标是在给定控制器数量的前提下,通过调整控制器部署位置以提升相关网络性能指标。而自适应部署方案则可以根据实际拓扑自适应地确定控制器部署的数量和位置,当然,这类方案在研究数量和深度方面还远未成熟。

非自适应控制器部署方案最早由Heller^[5]提出,其在给定拓扑和控制器部署的数量前提下,通过遍历手段探索不同控制器部署的位置对交换机与控制器之间交互时延的影响,并选择交互时

延指标最优的控制器位置作为最终部署方案。该研究虽然对控制器部署方案具有奠基性作用,但其仅考虑了交互时延这一单一指标,降低了该方案的通用性。随后,不断有研究人员从改善不同指标入手提出了多种不同研究方案。文献[6]从经济开销角度对控制器部署方案进行建模评估,以获得经济最优化部署方案;文献[7]在Heller所提出模型的基础上增加了控制器容量限制,从一定程度丰富了模型考虑因素;文献[8]以控制器负载均衡为优化目标,进一步增加了控制器部署模型的通用性;文献[9]应用了主备控制器机制,并将交换机与对应主备控制器的交互时延作为优化目标,增加了控制器的可靠性;文献[10-11]则通过控制路径复用程度对节点压力和控制路径丢失率等指标进行建模优化,在一定程度上提高了控制器部署的可靠性。文献[12]在预先考虑链路故障的场景下设计了一种低故障恢复开销的控制器部署方案,该方案能够有效降低链路故障情况对全局控制时延的负面影响,增加了控制器部署策略在链路故障发生时的可靠性;文献[13]针对SDN大规模广域网场景提出了一种基于网络集群划域的控制部署算法CNPA,该算法可以有效降低单个网络域内控制器和交换机之间的最大端到端时延;文献[14]提出了一种控制器静态放置与负载动态调整相结合的方案,通过在考虑时延和负载两个指标的控制部署静态方案的基础上设计了控制器负载动态调整算法(ADOA),提高了离线和在线两种状态下控制器负载均衡的能力。通过分析可知,上述非自适应部署方案与本文提出的MOSBCPP模型相比,缺少安全属性的考量,而且,相关性能指标局部特征明显。另外,非自适应部署方案中基于固定控制器数量的假设也限制了其通用性。

在自适应部署方案方面,现有研究仍较为初步,其基本思想均是通过聚类或生成树划分等方法实现子网划分,进而在每一子网内根据交互时延指标最小化来部署控制器。文献[15]根据节点密度的聚类分析方法提出了DBCP部署算法;文献[16]通过Min-Cut方法计算最大不相交路径,并以该指标为参考对网络拓扑进行分割;文献[17]利用改进的吸引力传播算法(affinity propagation)对子域进行划分;文献[18]在划分节点层次树结

果的基础上,结合负载均衡指标提出了 MCDALB 部署算法;文献[19]提出了一种基于最小容错覆盖模型的控制器部署方案,该方案能够保证数据平面容错需求的同时尽可能降低控制器部署数量。以上自适应部署方案虽然将控制器数量作为考量因素,但其本质仍以提供可行解为目标,与 MOSBCPP 相比,相关性能指标同样较为片面,安全性能等因素同样被忽视。

3 模型定义

为了更好地从数学角度描述考虑拜占庭容错属性的 SDN 安全控制器多目标部署问题,本节首先对相关模型参数进行定义说明。

模型参数描述 假设给定的网络拓扑可抽象表示为 $G=(V,E)$, 其中, V 表示网络拓扑中的节点集合, E 表示节点之间的连边集合。节点之间的最短距离存储在距离矩阵 D 中, 距离矩阵中的元素 d_{ij} 表示节点 i 和 j 之间的最短距离。最短距离的度量标准可选择时延或跳数指标, 为了便于后续直观描述, 本文选择时延作为统一距离度量标准。网络拓扑图 G 的节点和连边结构是由数据平面中交换机位置和其连接关系所决定的, 即图中的每个节点均实际对应一个交换机, 节点之间的连边对应交换机之间的通路。控制器则可以根据相关策略选择网络拓扑图中的任意节点作为其部署位置, 其中, 每个节点位置仅可部署一个控制器。控制器部署位置可通过二进制变量 y_j 表示, 当控制器部署在节点 j 时, 其值为 1, 否则为 0, 具体表达如式(1)所示。

$$y_j = \begin{cases} 1, & \text{控制器部署在节点} j \text{上} \\ 0, & \text{控制器未部署在节点} j \text{上} \end{cases} \quad (1)$$

其中, $j \in V$ 。针对交换机与控制器之间的连接关系, 本文同样以二进制变量 x_{ij} 表示, 当位于节点 i 的交换机被位于节点 j 的控制器控制时, 其值为 1, 否则为 0, 具体表达如式(2)所示。

$$x_{ij} = \begin{cases} 1, & \begin{array}{l} \text{位于节点} i \text{的交换机被位于} \\ \text{节点} j \text{的控制器控制} \end{array} \\ 0, & \begin{array}{l} \text{位于节点} i \text{的交换机未被可能} \\ \text{位于节点} j \text{的控制器控制} \end{array} \end{cases} \quad (2)$$

其中, $i, j \in V$ 。利用上述基本参数便可以对控制

器部署位置、数量及交换机与控制器之间的连接关系等结果进行符号描述。当然, 在具体确定上述结果时, 需要综合考虑基本约束及优化目标等因素, 为此, 接下来对相关基本约束条件展开讨论。

连接关系约束 在式(2)所示的连接关系中, 当位于节点 i 的交换机被位于节点 j 上的控制器所控制时 ($x_{ij}=1$), 隐含着必须在相应节点 j 上部署控制器的前提约束条件; 而当位于节点 i 的交换机未被可能位于节点 j 的控制器控制时 ($x_{ij}=0$), 则节点 j 上可能部署控制器而未与节点 i 的交换机建立连接, 也有可能节点 j 上未部署控制器。上述约束关系可由式(3)表示。

$$x_{ij} \leq y_j, \quad \forall i, j \in V \quad (3)$$

拜占庭容错约束 为了满足拜占庭容错属性, 每个交换机均需根据其容错需求度从网络拓扑图中选择满足相关条件的多个控制器作为参考, 因此, 交换机所连接的控制器数量首先需要满足相应约束, 即

$$\sum_{j \in V} x_{ij} \geq R_i, \quad \forall i \in V \quad (4)$$

其中, $\sum_{j \in V} x_{ij}$ 表示交换机 i 所连接的控制器数目之和, R_i 则表示交换机 i 的容错需求度, 为了便于说明, 本文将交换机容错需求度等价于交换机所需要连接的控制器数目。

时延相关约束 由于 SDN 交换机与控制器的交互时延直接影响全局网络性能, 因此, 在建立交换机与控制器的连接关系前, 还需要对该指标进行约束, 即

$$d_{ij} x_{ij} \leq d_{\text{interaction}}^{\max}, \quad \forall i, j \in V \quad (5)$$

$d_{ij} x_{ij}$ 涵盖了交换机 i 与控制器 j 可能存在的各种连接时延情况, $d_{\text{interaction}}^{\max}$ 则表示交换机与控制器可建立连接的最大交互时延阈值。同理, 由于每个交换机均由具有拜占庭容错属性的多个 SDN 控制器所控制, 控制器之间的同步时延会对全局网络性能产生重要影响, 因此, 在建立交换机与控制器的连接关系时也需要对控制器同步时延进行约束, 即

$$d_{j_1 j_2} x_{j_1 i} x_{j_2 i} \leq d_{\text{syn}}^{\max}, \quad \forall i \in V, j_1 \neq j_2 \in V \quad (6)$$

其中, $d_{j_1 j_2} x_{j_1} x_{j_2}$ 表示交换机 i 所连接的任意两个不同控制器 j_1 和 j_2 之间的同步时延情况, $d_{\text{syn}}^{\max}$ 则表示同一交换机所连接的多个控制器之间的最大同步时延阈值。

备选域相关约束 通过交换机与控制器交互时延约束, 以及控制器之间的同步时延约束可以初步划定针对各交换机的所能建立连接的控制器集合范围即拜占庭式控制器备选域 (BCAD, Byzantine controller alternative domain), 这一范围为提供可行解奠定了重要基础。对于交换机 i 的拜占庭式控制器备选域 BCAD_i , 其表达式如式(7)所示。

$$\text{BCAD}_i = \Gamma_i^{d_{\text{interaction}}^{\max}} \cap \Gamma_j^{d_{\text{syn}}^{\max}}, \forall i \in V, j \in \Gamma_i^{d_{\text{interaction}}^{\max}} \quad (7)$$

其中, $\Gamma_i^{d_{\text{interaction}}^{\max}}$ 表示距离交换机 i 小于最大交互时延阈值 $d_{\text{interaction}}^{\max}$ 的节点集合, $\Gamma_j^{d_{\text{syn}}^{\max}}$ 表示距离 $\Gamma_i^{d_{\text{interaction}}^{\max}}$ 内的节点 j 小于最大同步时延阈值 $d_{\text{syn}}^{\max}$ 的节点集合, 上述两集合的交集即为交换机 i 的拜占庭式控制器备选域。同时, 为了满足交换机 i 的容错需求, 其备选域 BCAD_i 内的节点数量需要大于其容错需求度, 即

$$|\text{BCAD}_i| \geq R_i, \forall i \in V \quad (8)$$

控制器容量约束 考虑到控制器计算资源的有限性, 故对于每个控制器而言, 其管理的交换机请求总负载需要小于其容量限制, 即

$$\sum_{i \in V} x_{ij} L_i \leq U_j y_j, \forall j \in V \quad (9)$$

其中, L_i 表示交换机 i 的请求负载, U_j 则表示控制器 j 的最大容量; $\sum_{i \in V} x_{ij} L_i$ 表示与控制器 j 相连接的所有交换机的负载之和, $U_j y_j$ 则表示在节点 j 上所部署的控制器 j 的容量上限。

在满足上述相关约束条件后, 便可以初步划定可行解的范围, 当然, 对最终结果的选取仍需进一步结合具体优化目标。为此, 本节选取最大交互时延、平均交互时延、最大同步时延、平均同步时延、控制器负载差异程度及控制器部署数量 6 个指标作为优化目标。

(1) 最大交互时延

网络拓扑中的任意交换机 i 与其连接的控制

器 j 的最大交互时延是反映网络性能下界的重要指标之一, 该值越小表明网络性能下界越优, 其表达如式(10)所示。

$$f_1 : \pi_{\text{max interaction latency}}(\text{CP}) = \max_{i \in V} \max_{j \in \{j \in V | x_{ij} \neq 0\}} d_{ij} \quad (10)$$

(2) 平均交互时延

网络拓扑中任意交换机 i 与其连接的所有控制器的平均交互时延是反映平均网络性能的重要指标之一, 该值越小, 表明平均网络性能越优, 其表达如式(11)所示, 其中, $|V|$ 表示网络拓扑中所有节点的数量。

$$f_2 : \pi_{\text{avg interaction latency}}(\text{CP}) = \frac{1}{|V|} \sum_{i \in V} \left(\frac{1}{\sum_{j \in V} x_{ij}} \sum_{j \in \{j \in V | x_{ij} \neq 0\}} d_{ij} \right) \quad (11)$$

(3) 最大同步时延

网络拓扑中的任意交换机 i 所连接的控制器之间的最大同步时延也是反映网络性能下界的重要指标之一, 该值越小, 同样表明网络性能下界越优, 其表达如式(12)所示。

$$f_3 : \pi_{\text{max syn latency}}(\text{CP}) = \max_{i \in V} \max_{j_1 \neq j_2 \in \{j \in V | x_{ij} \neq 0\}} d_{j_1 j_2} \quad (12)$$

(4) 平均同步时延

网络拓扑中任意交换机 i 所连接的控制器之间的平均同步时延也是反映平均网络性能的重要指标之一, 该值越小, 同样表明平均网络性能越优, 其表达如式(13)所示。

$$f_4 : \pi_{\text{avg syn latency}}(\text{CP}) = \frac{1}{|V|} \sum_{i \in V} \left(\frac{1}{C_{\sum_{j \in V} x_{ij}}^2} \sum_{j_1 \neq j_2 \in \{j \in V | x_{ij} \neq 0\}} d_{j_1 j_2} \right) \quad (13)$$

(5) 控制器负载差异程度

在 SDN 多控制器模型中, 各控制器的负载均衡程度是衡量控制器部署合理性的重要指标, 各控制器负载越均衡, 则相应地控制器部署策略合理性越高。因此, 本文将控制器负载差异程度也作为优化指标, 负载差异值越小, 表明控制器负载均衡程度越高, 全局网络性能越优, 其表达式如式(14)所示。

$$f_5 : \pi_{\text{imbalace}}(\text{CP}) = \sqrt{\frac{1}{\sum_{j \in V} y_j} \sum_{j \in \{j \in V | y_j \neq 0\}} \left(\sum_{i \in V} x_{ij} L_i - \overline{\text{CL}} \right)^2} \quad (14)$$

其中, $\overline{\text{CL}}$ 表示所有控制器的负载的平均值, 其表达如式(15)所示:

$$\overline{\text{CL}} = \frac{\sum_{i \in V} \sum_{j \in V} x_{ij} L_i}{\sum_{j \in V} y_j} \quad (15)$$

(6) 控制器部署数量

考虑到控制器资源的有限性, 本着节约资源的原则, 需要合理控制控制器部署数量。控制器部署数量越少, 相应地, 网络运维开销就越小, 其表达如式(16)所示。

$$f_6 : \pi_{\text{number}}(\text{CP}) = \sum_{j \in V} y_j \quad (16)$$

通过对上述模型参数、约束条件及优化目标分析后可知, 本文所提出的 MOSBCPP 可建模为多目标优化模型, 如式(17)所示。

$$\min F(\text{CP}) = \{f_1, f_2, f_3, f_4, f_5, f_6\} \quad (17)$$

约束条件:

式(1)~式(9)

4 算法设计

基于前文分析可知, MOSBCPP 属于多目标优化模型, 而且相关优化目标间存在竞争关系, 因此, 该模型很难求得绝对最优解。为了解决该问题, 本文设计了一种基于 NSGA-II 的 MOSBCPP 求解算法, 该算法可以有效得出 Pareto 最优前沿以供决策者使用。不同于绝对最优解, Pareto 最优前沿是由 Pareto 最优解构成的。在求解过程中, 如果可行解 CP_n 满足如下关系即可认定为 Pareto 最优解: 不存在其他可行解 CP_m 使得 $\forall l, f_l(\text{CP}_m) \leq f_l(\text{CP}_n) \wedge \exists l, f_l(\text{CP}_m) < f_l(\text{CP}_n)$, 即 CP_n 不被其他任何解支配 (上述逻辑表达式即为支配的概念)。在以上概念的基础上, 本章将介绍具体算法流程。

4.1 基于 NSGA-II 的 MOSBCPP 求解算法

基于 NSGA-II 的 MOSBCPP 求解如算法 1 所示, 其主要思路如下: 首先, 在网络拓扑图 G 及

相关约束信息 INF 的基础上, 通过初始化函数生成规模为 N 的初始种群 CP_0 , 其中, 种群内的个体即为控制器部署策略 (第 1 行); 然后, 在初始种群的基础上, 通过变异、快速非支配排序、精英策略选择等多轮进化迭代过程, 得出目标种群 $\text{CP}_{\text{target}}$ 。在每轮迭代过程中, 原始种群 CP_m 与通过变异函数生成的变异种群 $\text{CP}_m^{\text{mutational}}$ 形成合并种群 Ω_m (第 3~4 行), 极大地保证了种群多样性; 在合并种群 Ω_m 的基础上进行快速非支配排序操作, 可以将种群个体按照非支配等级 Rank 排序, 并输出分层后的种群结果 Z (第 5 行); 精英策略选择函数则针对分层后的种群 Z , 挑选出适应度较高的 N 个个体作为下一轮迭代的原始种群。在上述进化迭代过程中, 如果进化迭代前后种群结果一致 (种群结果收敛, 第 7~10 行) 或达到最大进化迭代次数 MaxIterations (第 2 行), 则算法结束并输出目标种群 $\text{CP}_{\text{target}}$ (第 8 行、第 12~14 行)。

算法 1 基于 NSGA-II 的 MOSBCPP 求解算法
输入 Topology G , Constraints INF, Size N

输出 Target solution set $\text{CP}_{\text{target}}$

- 1) $\text{CP}_0 \leftarrow \text{Initialize}(G, \text{INF}, N)$
- 2) **for** $m=0$ to MaxIterations - 1 **do**
- 3) $\text{CP}_m^{\text{mutational}} \leftarrow \text{Mutate}(\text{CP}_m, G, \text{INF}, N)$
- 4) $\Omega_m \leftarrow \text{CP}_m \cup \text{CP}_m^{\text{mutational}}$
- 5) $Z \leftarrow \text{FastNondominatedSort}(\Omega_m)$
- 6) $\text{CP}_{m+1} \leftarrow \text{EliteSelection}(Z, N)$
- 7) **if** $\text{CP}_{m+1} == \text{CP}_m$ **then**
- 8) $\text{CP}_{\text{target}} \leftarrow \text{CP}_{m+1}$
- 9) **break**
- 10) **end if**
- 11) **end for**
- 12) **if** $\text{CP}_{\text{target}} == \phi$ **then**
- 13) $\text{CP}_{\text{target}} \leftarrow \text{CP}_{\text{MaxIterations}}$
- 14) **end if**

通过上述算法分析可知, 基于 NSGA-II 的 MOSBCPP 求解算法主要包括控制器部署策略初始化函数 Initialize、变异函数 Mutate, 快速非支配排序函数 FastNondominatedSort 及精英策略选择函数 EliteSelection 等。下面将对上述函数进行

具体介绍。

4.2 基于非确定交换机序列的控制器部署策略初始化函数

为了生成可行的初始化控制器部署策略,需要令每一个交换机均满足其相关容错及时延约束条件,该过程可通过在每一个交换机的备选域中选择满足交换机容错需求数量的节点建立连接关系实现。当然,在为每一个交换机分配控制器节点的过程中,考虑到各交换机备选域的耦合性及控制器容量的有限性,交换机分配序列顺序会对最终结果产生直接影响。为此,算法 2 首先对网络拓扑图 G 内的所有节点的连通特征进行评估,并根据评估结果对交换机分配序列排序(第 1~3 行);为了描述交换机连通特征,本函数引入交换机度中心因子和交换机聚集因子等两个指标。

交换机度中心因子的表达如式(18)所示。

$$\text{DegreeCentrality}(i) = \frac{\text{Degree}(i)}{|V| - 1}, \quad i \in V \quad (18)$$

其中, $\text{Degree}(i)$ 表示图 G 中交换机 i 的度,交换机的度中心因子能够衡量该交换机在图 G 中的中心程度。

交换机聚集因子的表达如式(19)所示。

$$\text{Cluster}(i) = \frac{\text{ConnectNum}(\text{Neighbor}(i))}{\text{Degree}(i)(\text{Degree}(i) - 1) / 2}, \quad i \in V \quad (19)$$

其中,分子部分表示了交换机 i 的邻居节点之间相互连接的数量,交换机聚集因子反映了交换机 i 所在节点的局域连接属性。

通过分析上述指标可知,节点的度中心因子及聚集因子越大,备选域内可选的控制器节点越多,随机复用的概率越高。因此,在制定初始交换机分配顺序 $\text{SWSeq}[0]$ 时,本函数按照连通性指标由大到小对交换机排序(优先比较交换机度中心因子,若该指标相同,则继续比较交换机聚集因子;第 4 行),这有利于产生优势控制器部署策略。当然,为了避免后续因初始部署策略多样性不足而导致的局部最优问题,本函数采用交叉变异方法产生 N 个不同的交换机分配序列 $\text{SWSeq}[p]$ (第 5~7 行)。然后,按照交换机分配序列内的顺序 $\text{SWSeq}[p]$,在每一交换机 $\text{SWSeq}[p][q]$ 的备选

域 $\text{BCAD}_{\text{SWSeq}[p][q]}$ 中随机选择满足交换机容错数量的 $R_{\text{SWSeq}[p][q]}$ 个节点组成备选控制器集合 $Y_{\text{SWSeq}[p][q]}$ 。其中,在将该交换机连接至备选控制器集合时,若集合中存在控制器过载,则需要重新随机选择备选控制器集合,直至满足负载约束条件(第 8~12 行)后,更新连接矩阵 X_p (第 13 行)。同理,针对不同的交换机分配序列,重复上述步骤,最终返回规模为 N 的初始种群 CP (第 15~17 行)。

算法 2 基于非确定交换机序列的控制器部署策略初始化函数

```
Function Initialize( $G(V, E), INF, N$ )
1) for each switch  $i \in V$  do
2)  $\text{Attributes}[i] \leftarrow$  Calculate the attribute of node  $i$  based on  $G$ 
3) end for
4)  $\text{SWSeq}[0] \leftarrow$  Sort the switch order based on  $\text{Attributes}[i]$ 
5) for  $p = 1$  to  $N - 1$  do
6)  $\text{SWSeq}[p] \leftarrow$  Change the switch order based on cross mutation
7) end for
8) for  $p = 0$  to  $N - 1$  do
9) for  $q = 1$  to  $|V|$  do
10) do{
11) switch  $\text{SWSeq}[p][q]$  randomly selects  $R_{\text{SWSeq}[p][q]}$  nodes in  $\text{BCAD}_{\text{SWSeq}[p][q]}$  to construct set  $Y_{\text{SWSeq}[p][q]}$ 
12) } while ( $\exists$  controller in  $Y_{\text{SWSeq}[p][q]}$  exceeds its capacity)
13) Update matrix  $X_p$  based on  $\text{SWSeq}[p][q]$  and  $Y_{\text{SWSeq}[p][q]}$ 
14) end for
15)  $\text{CP}[p] \leftarrow X_p$ 
16) end for
17) return  $\text{CP} = [\text{CP}[0], \dots, \text{CP}[N - 1]]$ 
End function
```

4.3 基于拜占庭备选域内随机约束性映射的控制器部署策略变异函数

为了保证进化迭代过程中种群的多样性,在

每一轮迭代过程中，变异函数（算法 3）均需在所输入的原始种群 CP 的基础上对其中的每一个体 $CP[m]$ 进行变异处理（第 1 行）。具体而言，针对每一个体，从图中随机选择 $\lceil \text{rate} \cdot |V| \rceil$ 个交换机节点组成变异交换机集合 $SW_{\text{mutational}}$ （第 2 行）。其中，变异概率为 rate 。然后，对于变异交换机集合 $SW_{\text{mutational}}$ 内的每一交换机 $SW_{\text{mutational}}[i]$ 执行随机更改控制器连接关系操作（第 3 行），即在每一交换机 $SW_{\text{mutational}}[i]$ 的备选域 $BCAD_{SW_{\text{mutational}}[i]}$ 中随机选择满足交换机容错数量的 $R_{SW_{\text{mutational}}[i]}$ 个节点组成备选控制器集合 $Y_{SW_{\text{mutational}}[i]}$ ，其中，类似于初始化函数中的相关步骤，在将该交换机连接至备选控制器集合时，若集合中存在控制器过载，仍需要重新随机选择备选控制器集合，直至满足负载约束条件（第 4~6 行）后，更新连接矩阵 X_m （第 7 行）。最终，该变异函数针对所输入的原始种群 CP 返回规模为 N 的变异种群 $CP_{\text{mutational}}$ （第 11 行）。

算法 3 基于拜占庭备选域内随机约束性映射的控制器部署策略变异函数

```

Function Mutate( $CP, G(V, E), INF, N$ )
1) for each strategy  $CP[m] \in CP$  do
2) Randomly select  $\lceil \text{rate} \cdot |V| \rceil$  switches to
   construct set  $SW_{\text{mutational}}$ 
3) for  $i = 0$  to  $|SW_{\text{mutational}}| - 1$  do
4) do{
5) switch  $SW_{\text{mutational}}[i]$  randomly selects
    $R_{SW_{\text{mutational}}[i]}$  nodes
   in  $BCAD_{SW_{\text{mutational}}[i]}$  to construct set
    $Y_{SW_{\text{mutational}}[i]}$ 
6) } while ( $\exists$  controller in  $Y_{SW_{\text{mutational}}[i]}$  exceeds
   its capacity)
7) Update matrix  $X_m$  based on  $SW_{\text{mutational}}[i]$ 
   and  $Y_{SW_{\text{mutational}}[i]}$ 
8) end for
9)  $CP_{\text{mutational}}[m] \leftarrow X_m$ 
10) end for
11) return  $CP_{\text{mutational}} = [CP_{\text{mutational}}[0], \dots, CP_{\text{mutational}}[N-1]]$ 
End function

```

4.4 基于双向支配属性的快速非支配排序函数

快速非支配排序函数能够对输入种群内的个体进行评级，按照非支配关系将不同个体归入不同等级的策略集合中，具体步骤如算法 4 所示。

在快速非支配排序过程中，每一个策略（个体）均维护两种统计变量，即该策略被其他策略所支配的数目 DominatedNum 以及该策略支配其他策略的集合 DominateSet 。通过对输入种群 Ω 内每一个策略与种群内其他策略进行非支配比较（其中 $CP[m] \prec CP[n]$ 为 $CP[m]$ 支配 $CP[n]$ ），可以分别得出每一策略的 DominatedNum 及 DominateSet （第 1~8 行）。基于该统计属性，可以将不被其他任何策略支配的策略挑选出来，然后组成第 0 级非支配策略集合 $Z[\text{Rank}_0]$ （第 9~11 行）。在不断排除当前等级的非支配策略集合的情况下（第 16~18 行），循环选取下一级最优非支配策略集合（在排除上一级非支配策略集合影响后，现有 DominatedNum 等于 0 的策略集合为下一级最优非支配策略集合）（第 19~21 行）。针对输入种群 Ω ，快速非支配排序函数返回具有不同非支配等级的策略集合 Z （第 26 行）。

算法 4 基于双向支配属性的快速非支配排序函数

```

Function FastNondominatedSort( $\Omega$ )
1) for each strategy  $CP[m] \in \Omega$  do
2) for each strategy  $CP[n] \in \Omega$  do
3) if  $CP[m] \prec CP[n]$  then
4)  $\text{DominateSet}(CP[m]) = \text{DominateSet}(CP[m]) \cup CP[n]$ 
5) else if  $CP[n] \prec CP[m]$  then
6)  $\text{DominatedNum}(CP[m]) = \text{DominatedNum}(CP[m]) + 1$ 
7) end if
8) end for
9) if  $\text{DominatedNum}(CP[m]) = 0$  then
10)  $Z[\text{Rank}_0] \leftarrow Z[\text{Rank}_0] \cup CP[m]$ 
11) end if
12) end for
13)  $i = 0$ 
14) while  $Z[\text{Rank}_i] \neq \emptyset$  do

```

```

15)  $Z[\text{Rank}_{i+1}] = \phi$ 
16) for each strategy  $\text{CP}[m] \in Z[\text{Rank}_i]$  do
17)   for each strategy  $\text{CP}[n] \in$ 
       DominateSet( $\text{CP}[m]$ ) do
18)     DominatedNum( $\text{CP}[n]$ ) =
       DominatedNum( $\text{CP}[n]$ ) - 1
19)   if DominatedNum( $\text{CP}[n]$ ) == 0 then
20)      $Z[\text{Rank}_{i+1}] \leftarrow Z[\text{Rank}_{i+1}] \cup \text{CP}[n]$ 
21)   end if
22) end for
23) end for
24)  $i = i + 1$ 
25) end while
26) return  $Z = [Z[\text{Rank}_0], \dots, Z[\text{Rank}_{i-1}]]$ 
End function

```

4.5 基于非支配临界等级与拥挤度指标的精英策略选择函数

为了保证种群进化质量,需要在快速非支配排序函数返回的具有不同非支配等级的策略集合 Z 中选出规模大小为 N 的精英策略继续进化。通过 4.4 节分析可知,非支配等级越小的策略集合越具有精英属性。为此,如算法 5 所示的精英策略选择函数首先判断第 0 级非支配策略集合 $Z[\text{Rank}_0]$ 内的策略数目是否大于或等于 N (第 1 行),如果满足该条件,表明仅需要从该等级的非支配策略集合中选取 N 个策略即可。为了令进化过程中的种群个体更加均匀,针对同一等级非支配策略集合内的个体,本函数引入拥挤度指标,该指标越大,表明该个体作为精英策略被选取概率越高。拥挤度指标表达如式(20)所示。

$$I(\text{CP}[k]) = \sum_{i=1}^6 \frac{f_i(\text{CP}[k+1]) - f_i(\text{CP}[k-1])}{f_i^{\max} - f_i^{\min}} \quad (20)$$

其中, $f_i^{\max}$ 表示在该等级非支配策略集合下优化目标 i 的最大值, $f_i^{\min}$ 为其最小值, $f_i(\text{CP}[k+1])$ 和 $f_i(\text{CP}[k-1])$ 表示在第 i 个优化目标下,与策略 $\text{CP}[k]$ 相邻策略的优化目标值。利用式 (21) 得到每一策略的拥挤度值后,按照降序对 $Z[\text{Rank}_0]$ 内的策略排序 (第 2 行),并选择前 N 个策略组成精英集合 Elite (第 3 行)。

如果第 0 级非支配策略集合 $Z[\text{Rank}_0]$ 内的个

体数目小于 N (第 5 行),那么首先利用式 (21) 确定需要进行拥挤度排序的非支配等级策略集合的标号值 n (第 6 行)。

$$\begin{cases} \sum_{i=0}^n |Z[\text{Rank}_i]| \leq N \\ \sum_{i=0}^{n+1} |Z[\text{Rank}_i]| > N \end{cases} \quad (21)$$

然后,利用式(1)计算 $Z[\text{Rank}_{n+1}]$ 内的每一个策略的拥挤度值,按照降序对其排序后(第 7 行),选择前 $N - \sum_{i=0}^n |Z[\text{Rank}_i]|$ 个策略组成精英集合 Elite (第 8 行)。此外,在该标号值之前等级的非支配策略集合,即 $Z[\text{Rank}_0] \sim Z[\text{Rank}_n]$,与上述 Elite 集合一并作为精英策略进入下一轮进化迭代过程 (第 9 行)。

算法 5 基于非支配临界等级与拥挤度指标的精英策略选择函数

```

Function EliteSelection( $Z, N$ )
1) if  $|Z[\text{Rank}_0]| \geq N$  then
2)    $\text{SZ} \leftarrow$  Sort the strategies in  $Z[\text{Rank}_0]$  based
       on formula 20
3)   Elite  $\leftarrow$  Select the top  $N$  strategies in SZ
4)   return Elite
5) else
6)   Calculate  $n$  based on formula 21
7)    $\text{SZ} \leftarrow$  Sort the strategies in  $Z[\text{Rank}_{n+1}]$  based
       on formula 20
8)   Elite  $\leftarrow$  Select the top  $\left( N - \sum_{i=0}^n |Z[\text{Rank}_i]| \right)$ 
       strategies in SZ
9)   Elite  $\leftarrow \bigcup_{i=1}^n Z[\text{Rank}_i] \cup \text{Elite}$ 
10) return Elite
11) end if
End function

```

5 仿真评估

为了评估 MOSBCPP 算法的有效性,本节主要针对交互时延、同步时延、负载差异程度、控制器部署数量和安全性能 5 方面的指标进行仿真。仿真环境中交换机规模 $|V|$ 设置为 500~2 500 个,

在每一交换机规模下, 利用 python 2.7 networkx 库随机生成 100 种不同拓扑, 交换机之间的时延范围在 1~20 ms 内随机分布, 每个交换机冗余度需求 R 均设置为 4, 交换机请求负载 L 均设置为 1, 控制器容量 U 均设置为 50, $d_{\text{interaction}}^{\text{max}}$ 和 $d_{\text{syn}}^{\text{max}}$ 均设置为 100 ms, 算法种群空间 N 设置为 100, 变异率 rate 设置为 0.2, 最大迭代次数 MaxIterations 设置为 30。此外, 为了横向对比 MOSBCPP 算法有效性, 本文选择考虑负载均衡优化的 MCDALB 算法^[18]和考虑最小容错控制器部署的 ABC 算法^[20]与本文算法进行对比, 而且相关分析均在每一个交换机规模下的不同拓扑均值结果上进行。

5.1 交互时延指标

不同控制器部署算法的平均交互时延及最大交互时延指标如图 2 所示。

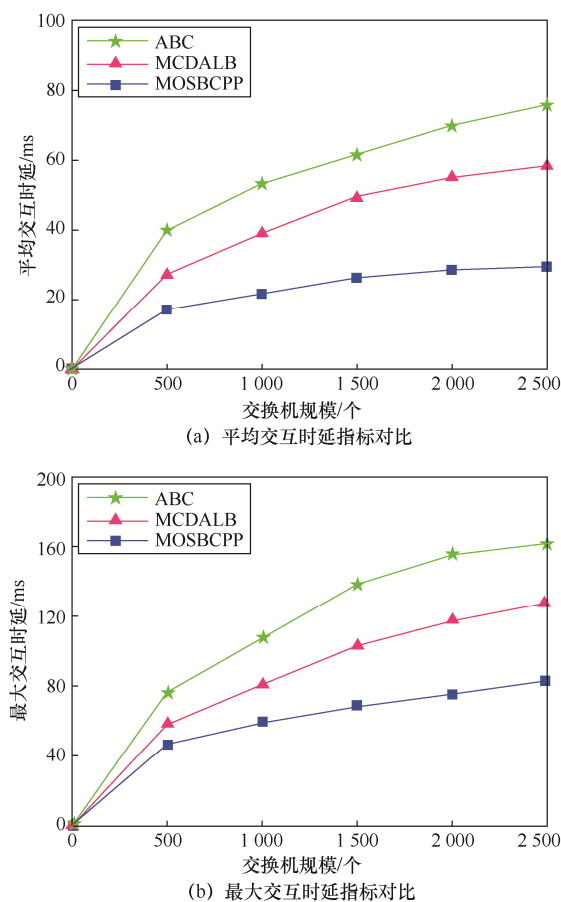


图2 不同控制器部署算法交互时延指标对比
Figure 2 Comparison of interaction delay indicators under different controller deployment algorithms

分析图 2 可知, MOSBCPP 无论在平均交互时延指标还是最大交互时延指标方面均优于

MCDALB 和 ABC 算法, 这是由于与 MOSBCPP 相比, MCDALB 和 ABC 分别以负载均衡程度和控制器部署数目为主要优化对象, 缺少对交互时延指标的优化。而且随着交换机规模的增加, MOSBCPP 交互时延上升幅度 (10 ms 左右) 低于其他两种算法, 这表明 MOSBCPP 算法具有稳定性。

5.2 同步时延指标

由于 MCDALB 不具备多控制器容错属性, 不涉及本文所考虑的同步时延指标, 因此, 本节仅对比 MOSBCPP 与 ABC 算法的平均同步时延及最大同步时延指标, 其结果如图 3 所示。分析图 3 可知, 与前述交互时延指标规律类似, MOSBCPP 无论在平均同步时延指标还是最大同步时延指标方面均优于 ABC 算法, 而且随着交换机规模的增加, MOSBCPP 同步时延波动相较于 ABC 更加稳定, 这是 MOSBCPP 综合考虑要素较为完备的优势。

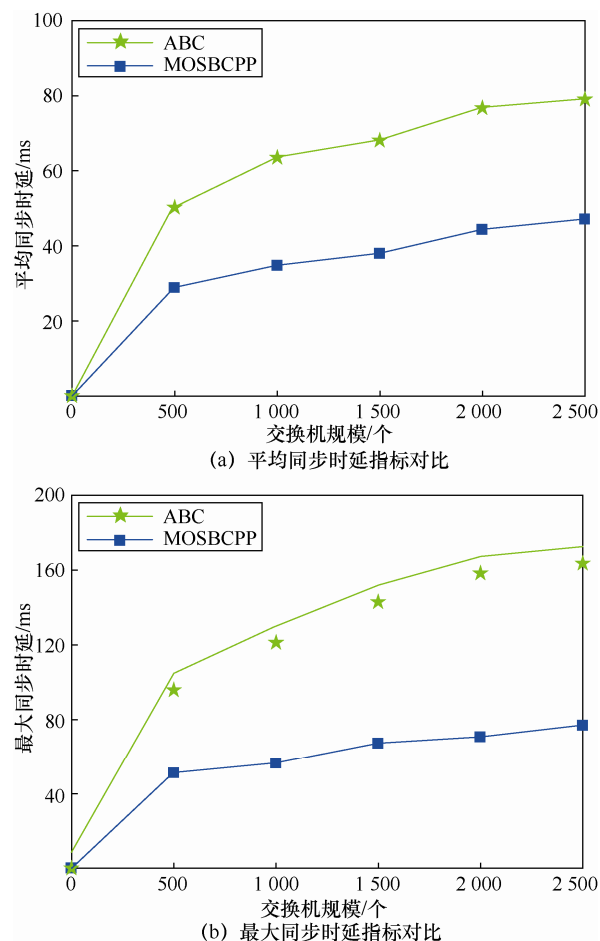


图3 不同控制器部署算法同步时延指标对比
Figure 3 Comparison of synchronization delay indicators under different controller deployment algorithms

5.3 负载差异程度指标

图4给出了不同控制器部署算法的归一化负载差异程度的CDF曲线。分析图4可知,相较于MOSBCPP和ABC算法,MCDALB控制器的负载差异程度最小,在90%的情况下负载差异度小于0.15。这是由于MCDALB以负载差异度为其优化目标,其考虑的因素要少于MOSBCPP下的多优化目标,故其在负载差异程度方面具有一定优势。虽然MOSBCPP在该指标上略劣于MCDALB,但其指标仍具有较高接受度,在90%的情况下低于0.3。而相比之下,ABC算法由于未考虑负载均衡因素,故其负载差异度最高,超过50%的情况下高于0.5。

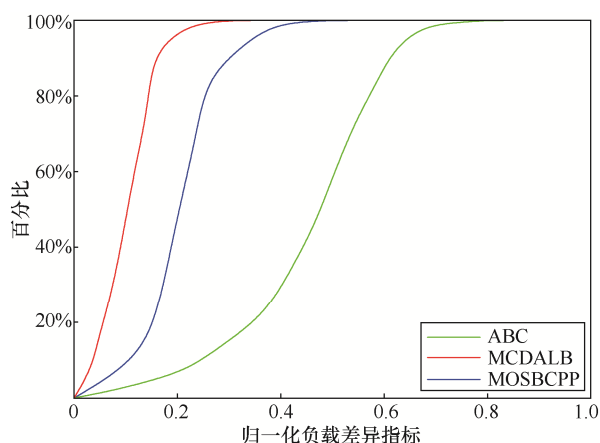


图4 不同控制器部署算法归一化负载差异指标对比
Figure 4 Comparison of normalized load difference indicators under different controller deployment algorithms

5.4 控制器部署数量指标

由于MOSBCPP与ABC算法均充分衡量了交换机的冗余属性,故在控制器部署数量指标方面,这两者与未考虑冗余容错属性的MCDALB算法不具有同等比较性。因此,本小节选择理论下界值TLB来衡量MOSBCPP与ABC算法在控制器部署数量上的绝对有效性。其中,TLB是综合考虑交换机请求负载、冗余容错需求度和控制器容量等因素而得出的控制器部署数量的理论下界。图5对比了3个算法的控制器部署数量。

从图5可知,在不同的交换机规模下,ABC的控制器部署数量均与理论下界TLB最为接近,而MOSBCPP所需要的控制器数量略多于前两者。这是由于ABC算法将控制器数量最小化作

为优化目标,而控制器部署数量指标仅是MOSBCPP众多优化目标中的一个,故ABC的控制器数量指标要优于MOSBCPP算法。但是,从绝对有效性来看,MOSBCPP算法的控制器部署数量指标与理论下界差距不大(差距15%左右),也具有较高接受程度。

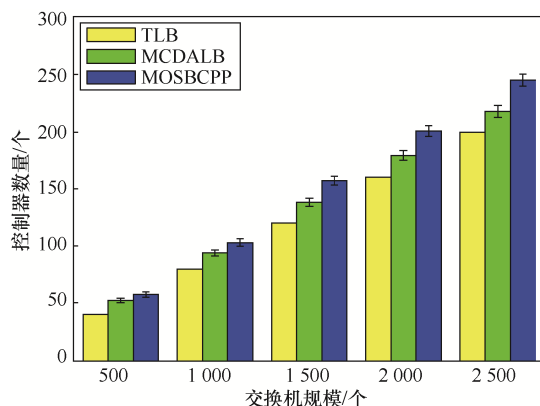


图5 不同控制器部署算法控制器数量指标对比
Figure 5 Comparison of the number of controllers under different controller deployment algorithms

5.5 安全性能指标

为了衡量不同控制器部署算法对控制平面带来的安全增益,本仿真在相应部署算法产生的结果基础上通过删除某些控制器实例来模拟控制器被劫持的场景,并在该场景中测量交换机连接失败率来衡量控制平面抗攻击的安全性能。本文仿真首先利用MOSBCPP、MCDALB和ABC算法在交换机规模为1000时的100种随机拓扑下生成控制器部署方案,然后随机删除5~10个控制器节点并测量交换机连接失败率。值得注意的是,只有在交换机没有连接任何控制器时,才将其归为交换机连接失败情况。

图6中给出了不同控制器部署算法的交换机连接失败率对比情况。分析图6可知,MCDALB算法的交换机连接失败率远高于MOSBCPP和ABC算法,这是由于MCDALB并未考虑冗余容错因素。MOSBCPP和ABC算法的交换机连接失败率则较为接近,而且随着删除控制节点数目的增加,两者交换机连接失败率均未产生明显波动,均稳定在0.1以内。当然,MOSBCPP在该指标上略微优于ABC算法,这也是控制器部署数量差异的折中体现。

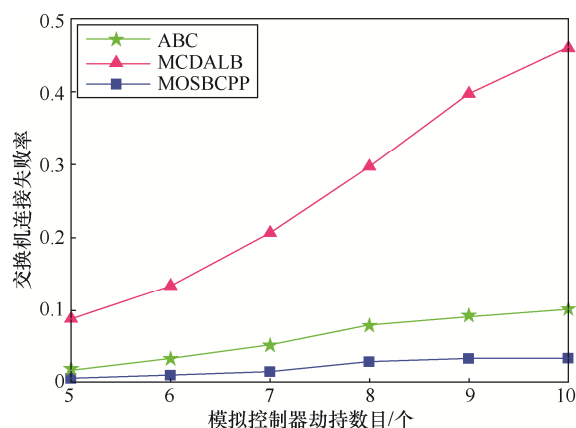


图6 不同控制器部署算法交换机连接失败率指标对比

Figure 6 Comparison of switch connection failure rate indicators under different controller deployment algorithms

6 结束语

为了提高 SDN 分布式控制平面的安全性,拜占庭式控制器受到广泛关注。在拜占庭式解决方案中,控制器部署的数量、位置,以及交换机与控制器之间的连接关系直接影响全局网络关键性能指标。为了提高控制器部署效率,本文提出了一种考虑拜占庭属性的 SDN 安全控制器多目标部署方案。通过基于 NSGA-II 思路对交互时延、同步时延、负载差异程度和控制器部署数量等多个优化指标的拜占庭控制器部署问题模型进行求解。相关仿真结果表明,该部署方案能够在有效降低交互时延、同步时延、负载差异程度和控制器部署数量等性能指标的同时,提高控制平面安全性。

参考文献:

- [1] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38(2): 69-74.
- [2] TOOTOONCHIAN A, GANJALI Y. HyperFlow: a distributed control plane for OpenFlow[C]//Internet Network Management Conference on Research on Enterprise Networking. 2011.
- [3] CHICA J, IMBACHI J, BOTERO J. Security in SDN: a comprehensive survey[J]. Journal of Network and Computer Applications, 2020, 159: 102595.
- [4] KRIIAA S, PAPILLON S, JAGADEESAN L, et al. Better safe than sorry: modeling reliability and security in replicated SDN controllers[C]//2020 16th International Conference on the Design of Reliable Communication Networks (DRCN). 2020.

- [5] HELLER B, SHERWOOD R, MCKEOWN N. The controller placement problem[J]. ACM Sigcomm Computer Communication Review, 2012, 42(4): 473-478.
- [6] SALLAHI A, ST-HILAIRE M. Optimal model for the controller placement problem in software defined networks[J]. IEEE Communications Letters, 2015, 19(1): 30-33.
- [7] YAO G, BI J, LI Y, et al. On the capacitated controller placement problem in software defined networks[J]. IEEE Communications Letters, 2014, 18(8): 1339-1342.
- [8] HU Y, LUO T, WANG W, et al. On the load balanced controller placement problem in software defined networks[C]//2016 2nd IEEE International Conference on Computer and Communications (ICCC). 2016.
- [9] BALA P. Capacitated next controller placement in software defined networks[J]. IEEE Transactions on Network and Service Management, 2017, 14(3): 514-527.
- [10] ISHIGAKI G, SHINOMIYA N. Controller placement algorithm to alleviate burdens on communication nodes[C]//International Conference on Computing. 2016.
- [11] HU Y, WANG W, GONG X, et al. On reliability-optimized controller placement for software-defined networks[J]. China Communications, 2014, 11(2): 38-54.
- [12] 吴奇, 陈鸿昶. 低故障恢复开销的软件定义网络控制器布局算法[J]. 网络与信息安全学报, 2020, 6(6): 97-104.
- [12] WU Q, CHEN H C. Low fault recovery cost controller placement strategy in software-defined networks[J]. Chinese Journal of Network and Information Security, 2020, 6(6): 97-104.
- [13] WANG G, ZHAO Y, HUANG J, et al. An effective approach to controller placement in software defined wide area networks[J]. IEEE Transactions on Network and Service Management, 2018, 15(1): 344-355.
- [14] 史久根, 谢熠君, 孙立, 等. 软件定义网络中面向时延和负载的多控制器放置策略[J]. 电子与信息学报, 2019, 41(8): 1869-1876.
- [14] SHI J G, XIE Y J, SUN L, et al. Multi-controller placement strategy based on latency and load in software defined network[J]. Journal of Electronics & Information Technology, 2019, 41(8): 1869-1876.
- [15] LIAO J, SUN H. Density cluster based approach for controller placement problem in large-scale software defined networkings[J]. Computer Networks, 2017, 112(1): 24-35.
- [16] ZHANG Y, BEHESHTI N, TATIPAMULA M. On resilience of split-architecture networks[C]//2011 IEEE Global Telecommunications Conference-GLOBECOM 2011. 2012.
- [17] ZHAO J, QU H, ZHAO J, et al. Towards controller placement problem for software-defined network using affinity propagation[J].

Electronics Letters, 2017, 53(14): 928-929.

- [18] 史久根, 郝伟, 贾坤荧, 等. 软件定义网络中基于负载均衡的多控制器部署算法[J]. 电子与信息学报, 2018, 40(2): 455-461.

SHI J G, ZHU W, JIA K Y, et al. Multi-controller deployment algorithm based on load balance in software defined network[J]. Journal of Electronics & Information Technology, 2018, 40(2): 455-461.

- [19] 吴奇, 陈鸿昶. 软件定义网络容错控制平面的最小覆盖布局方法[J]. 电子与信息学报, 2020, 42(12): 2849-2856.

WU Q, CHEN H C. Minimal coverage model for fault-tolerant controller placement in software-defined networks[J]. Journal of Electronics & Information Technology, 2020, 42(12): 2849-2856.

- [20] XIE J, GUO D, ZHU X, et al. Minimal fault-tolerant coverage of controllers in IaaS datacenters[J]. IEEE Transactions on Services Computing, 2020, 13(6): 1128-1141.

[作者简介]



王涛(1993-), 男, 山东临朐人, 信息工程大学博士生, 主要研究方向为SDN安全。



陈鸿昶(1964-), 男, 河南新密人, 信息工程大学教授、博士生导师, 主要研究方向为未来网络体系结构、人工智能。