软件定义网络容错控制平面的最小覆盖布局方法

吴 奇* 陈鸿昶

(中国人民解放军战略支援部队信息工程大学 郑州 450001)

摘 要：容错控制平面通过将多个控制器部署在不同的网络设备上进而增强网络的可靠性，但是大量的控制器部署带来了巨大的布局成本，严重地限制了容错控制平面在实际网络中的部署与应用。为了解决上述问题，该文首先构造了容错控制平面的最小覆盖布局模型，然后设计了一种基于局部搜索策略的启发式控制器布局算法，避免搜索结果陷入局部最优解。在不同规模网络中的仿真结果表明，相对于其他算法，所提算法可以在保证网络容错需求的同时，降低网络中部署控制器的数量。

关键词：软件定义网络；错误容忍；控制器布局；最小覆盖

中图分类号：TN919; TP393 **文献标识码**：A **文章编号**：1009-5896(2020)12-2849-08

DOI: 10.11999/JEIT190972

Minimal Coverage Model for Fault-Tolerant Controller Placement in Software Defined Networks

WU Qi CHEN Hongchang

(*PLA Strategic Support Force Information Engineering University*, *Zhengzhou* 450001, *China*)

Abstract: In order to deploy fault-tolerant Software-Defined Networks(SDN), many controllers must be physically distributed among different network devices. However, a large number of controllers bring huge costs, which limits severely the application of the fault-tolerant control plane to the real networks. In order to solve the above problems, the fault-tolerant control plane is analyzed and a mathematical model that covers all switches using the least number of controllers is constructed. Then, a heuristic controller placement algorithm based on the local search strategy is proposed to avoid the local optimal solution. The experimental results show that compared with other algorithms, the proposed algorithm can effectively reduce the number of required controllers while ensuring network fault tolerance requirements in different scale networks.

Key words: Software Defined Networks(SDN); Fault-tolerant; Controller placement; Minimal coverage model

1 引言

软件定义网络(Software Defined Networks, SDN)[1]是一种实现了控制平面和数据平面分离的新型网络结构，它可以通过一个集中化的控制器，运用可编程技术，将整个网络抽象成一个统一化视图进行管理，增强了网络管理的灵活性和可扩展性。

虽然控制平面和数据平面的解耦给网络管理带来了很大的灵活性，但是也带来了很多的安全风险。单点故障问题就是SDN中的一个典型问题：一旦控制器出现故障，整个SDN网络都无法正常工作。为了避免这一问题，基于多控制器的控制平面被提出用以取代单控制器的控制平面，典型的多控制器的控制平面基于Openflow1.2版本设计，包含主备冗余的控制平面和错误容忍的控制器平面。在主备控制平面中[2]，只有当Master控制器故障后，Slave控制器才可以取代Master控制器，进而对交换机进行控制。该类架构的优点在于可以降低控制器故障对网络的影响，其缺点在于仅可以防御控制器宕机等良性错误，对于实际场景中常发生的拜占庭错误无法进行有效防御，故容错控制平面[3]被提出。在容错控制平面中每一个交换机同时被1个Master控制器和多个Equal控制器控制，只要控制器的故障数不超过故障容忍上限，任何控制器的故障都对网络的工作没有影响。

虽然容错控制平面可以大幅度提高网络的可靠性，但是由于通信协议和控制器冗余度等要求，容错控制平面的部署成本往往非常高昂，研究者为了降低容错控制平面的部署成分设计了不同的方法[4,5]。

收稿日期：2019-12-09；改回日期：2020-05-27；网络出版：2020-06-22
*通信作者： 吴奇 wqstudyy@126.com
基金项目：国家重点研发计划(2018YFB0804004)
Foundation Item: The National Key Research and Development Program of China (2018YFB0804004)

但是上述机制大多以时延为目标进行设计，忽视了控制器部署数量对网络的花销的影响，事实上，对于SDN而言，越多的控制器代表越多的资源(CPU，内存、带宽等)消耗[6]，如何在满足实际容错需求的情况下，最小化网络中控制器的部署数量，以降低控制平面的部署成本是容错控制平面需要解决的问题。

本文针对容错控制平面中的多控制器布局问题进行研究，着重分析控制平面的最小容错覆盖问题(Minimal Fault-Tolerant Coverage Problem, MFTCP)，优化目标在于寻找一种控制器布局方案，使得网络在满足网络容错需求等指标的同时，最小化控制器的部署数量。本文的贡献和创新工作主要如下：

(1) 将网络故障容忍能力、交换机控制器之间的通信时延、控制器之间的通信时延及控制器的负载能力纳入到容错控制平面的布局模型中，构造了控制平面的最小容错覆盖模型。

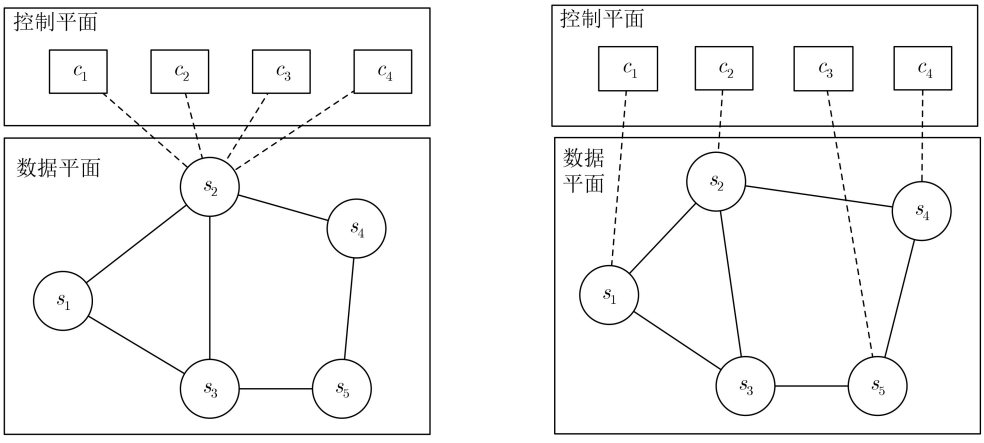
(2) 设计了一个基于局部搜索的控制器布局算法。算法可以通过更新策略，不断产生邻居控制器布局，最终得到满足终止条件的控制器布局。

(3) 在仿真中，在多种网络中对算法性能进行了分析，验证了算法在不同规模网络中的普适性；并通过与现有其他算法进行对比，验证了算法可以以更低的控制器部署数量实现满足容错需求的控制平面布局。

2 相关研究

近年来，多控制器控制平面的布局问题一直是SDN网络中的核心问题[7,8]，其研究主要分为主备控制平面布局问题和容错控制平面布局问题。主备控制平面布局问题主要针对控制平面中出现控制器崩溃、连边中断等故障进行设计[9]。容错控制平面则在主备控制平面的基础上更进一步，其不仅解决控制平面的控制器崩溃、连边中断等问题，还解决控制平面中出现“叛徒”控制器的问题，该类控制平面中交换机需要同时被多个控制器控制，该机制通过拜占庭通信协议，可以保证在$3f+1$个控制器组成的控制平面中即使f个控制器发生拜占庭故障，网络中的交换机仍然可以正常工作。图1(圆圈代表交换机，方块代表控制器)展示了容错控制平面布局的两种方式，两种方式中控制器c_1,c_2,c_3,c_4同时控制着交换机s_1,s_2,s_3,s_4,s_5。图1(a)是集中式部署方式[10]，该方法将全部的控制器部署在1个交换机上，但是这种方式存在一个风险，即一旦部署控制器的节点(比如交换机s_2)出现故障，网络中的控制器可能全部出现故障；图1(b)是分布式部署方式[3]，该方法将控制器广泛的部署在不同的交换机节点上，避免1个控制节点故障时，整个网络都失去控制。

上述研究大多关注如何在网络中部署控制器的问题，缺乏对多控制器的控制平面中部署多少个控制器的分析。事实上，如何利用数量最少的控制器实现对网络中交换机的控制，也是SDN网络的研究重点。文献[11]在1个交换机仅有1个Master控制器的网络中，利用权重公式进行量化网络中每一个交换机的重要性，构造出需要控制器数量最少的可靠布局；文献[6]通过为网络设置可靠性下界，结合启发式评级函数，建立一个最少控制器的控制平面布局模型。文献[12]最接近本文思路，其通过最小化单控制器和双控制器的场景下增量控制器的部署数量，进而求解控制器的最小覆盖问题，但是该思路存在两方面问题：(1)主要针对主备切换的SDN架构进行设计，缺少了对控制器之间时延关系这个对容错控制平面至关重要的要素的分析；(2)是一种



(a) 全部控制器部署在1个交换机上　　(b) 全部控制器分散部署在多个交换机上

图 1 容错控制平面的布局方式

基于贪婪算法的求解思路，该算法很容易陷入局部最优解而不是全局最优解。

3　控制平面的最小容错覆盖问题

3.1　相关参数

本文主要针对分布式容错控制平面进行研究，数据平面由N个交换机组成，可以表示为$S=\{s_1, s_2, \cdots, s_N\}$，每一个交换机生成的流请求数量为$q_i$；架构的控制平面由控制器组成，可以表示为$C=\{c_1, c_2, \cdots, c_m\}$，每一个控制器的负载容量为$u_j$。网络中两个交换机之间的最短距离可以被定义为$d_{ij}$，若交换机$i$被控制器$j$控制，则参数$w_{ij}=1$，反之$w_{ij}=0$。对于一个交换机$i$而言，交换机$i$被分配的控制器数量越多，表示交换机$i$控制平面的容错能力越强，本文中，为简化模型，将交换机被分配的控制器数量等同于交换机的容错能力r_i。此外，在分配过程中，为了避免单点故障导致的错误，每一个交换机不仅需要被多个控制器控制，还需要这些控制器部署在不同的节点上，本文设定1个节点上只能部署1个控制器，若交换机j部署于控制器，则变量$p_j=1$，反之$p_j=0$。

3.2　问题描述

MFTCP问题包含多种约束条件。首先考虑到在MFTCP的分布式容错控制平面中，只有部署控制器的节点才可以作为交换机的控制节点，故交换机的控制节点约束可以表示为

$$w_{ij} \le p_j,\ \ i \in S, j \in S \tag{1}$$

然后考虑到软件定义网络中，时延需求是网络性能的一个核心需求，对MFTCP问题中的容错控制平面而言，时延需求主要体现在交换机和控制器之间的通信时延和控制器之间的同步时延上，故需要分别对交换机与控制器之间的通信时延和控制器之间的同步时延进行讨论。

交换机与控制器之间的通信时延约束：在交换机与控制器进行通信时，首先需要向控制器发送请求，这就要求交换机到控制器之间的时延必须要在一定的范围内，若交换机到控制器之间的时延的最大值为$d_{\max}^{\mathrm{cs}}$，则网络中任意交换机i和控制器j之间的时延关系可以表示为

$$d_{ij} w_{ij} \le d_{\max}^{\mathrm{cs}},\ \ i \in S, j \in S \tag{2}$$

控制器之间的同步时延约束：在交换机向控制器发送过请求消息后，交换机的全部控制器之间需要进行相互通信与同步，并按照大数判决原则对控制器的回复消息进行判决[6]，这一过程中需要控制器之间的时延必须要在一定的范围内，若控制器之间的最大时延为$d_{\max}^{\mathrm{cc}}$，则网络中控制交换机i的两个控制器j和j'之间的时延关系可以表示为

$$d_{jj'} w_{ij} w_{ij'} \le d_{\max}^{\mathrm{cc}},\ \ i \in S, j \in S, j' \in S \tag{3}$$

在MFTCP问题中，网络的可靠性是必须满足的需求。为了保证网络的可靠性，每一个交换机都需要被足够的控制器控制，进而满足交换机的容错需求。若交换机i的容错需求为r_i，则交换机i至少需要被r_i个控制器控制，则交换机的控制器数量约束可以表示为

$$\sum_{j \in S} w_{ij} \ge r_i,\ \ i \in S \tag{4}$$

考虑到控制器与交换机之间的时延关系及控制器之间的时延关系大大限制了交换机i的控制器选择范围，即交换机的容错需求是被时延约束限制的，故需要分析交换机i的容错需求r_i与时延约束之间的关系。若参数N_i^d表示以交换机i为中心，时延范围为d之内的全部节点集合，则交换机i的候选控制器集合$\Gamma(s_i)$(candidate controller set)可以表示为

$$\Gamma(s_i) = N_i^{d_{\max}^{\mathrm{cs}}} \cap N_j^{d_{\max}^{\mathrm{cc}}},\ \ i \in S, j \in N_i^{d_{\max}^{\mathrm{cs}}} \tag{5}$$

其中$N_i^{d_{\max}^{\mathrm{cs}}}$表示交换机$i$在满足$d_{\max}^{\mathrm{cs}}$约束下的节点集合，$N_j^{d_{\max}^{\mathrm{cc}}}$(其中$j \in N_i^{d_{\max}^{\mathrm{cs}}}$)表示节点集合中的节点$j$在满足$d_{\max}^{\mathrm{cc}}$约束下的节点集合。$N_i^{d_{\max}^{\mathrm{cs}}} \cap N_j^{d_{\max}^{\mathrm{cc}}}$构成了同时满足$d_{\max}^{\mathrm{cs}}$和$d_{\max}^{\mathrm{cc}}$约束下的交换机$i$候选控制器集合$\Gamma(s_i)$。

式(5)本质上就是时延约束下的候选控制器集合，故在式(5)的基础上，根据交换机的容错需求r_i，可以得到时延约束限制下的交换机的容错需求约束

$$|\Gamma(s_i)| \ge r_i,\ \ i \in S \tag{6}$$

最后考虑到MFTCP问题中的控制器的负载存在上限，即每一个控制器可以处理的交换机的请求数量存在上限。若控制器的最大容量为u_j，交换机的请求数为q_i，则控制器容量与交换机请求数之间的容量约束可以表示为

$$u_j p_j \ge \sum_{i \in S} q_i w_{ij},\ \ i \in S, j \in S \tag{7}$$

考虑到容错控制平面中控制器部署数量可以表示为：$\sum_{j \in S} p_j$，结合上述约束，故MFTCP问题可以建模为

$$\min \sum_{j \in S} p_j \tag{8}$$

s.t.

式(1)—式(4)，式(6)和式(7)

$$p_j \in \{0,1\},\ \ j \in S \tag{9}$$

$$w_{ij} \in \{0,1\},\ \ i \in S, j \in S \tag{10}$$

3.3 复杂性分析

定理1 软件定义网络中MFTCP问题是NP完全问题。

证明：软件定义网络中的控制器覆盖问题是典型的NP完全问题[13]，控制器覆盖问题的核心在于寻找部署最少数量的控制器，使得每一个交换机被1个控制器控制。控制器覆盖问题是MFTCP问题的一个特殊情况，这是因为一旦将MFTCP问题中的交换机与控制器的一对多映射简化为一对一映射，控制器覆盖问题就和简化后的MFTCP问题完全等价，故控制器覆盖问题只是MFTCP问题的一个子实例，故MFTCP问题也是NP完全问题。证毕

4 模型求解

考虑到MFTCP问题是NP完全问题，直接对MFTCP问题进行求解最优解，需要的时间随着问题规模呈指数增长。因此本文设计了一种基于局部搜索算法的控制器布局算法(Local Search based Controller Placement Algorithm, LSCPA)对MFTCP问题进行求解，算法从一个控制器布局开始，通过邻居更新策略，不断产生邻居候选控制器布局，根据评估函数评估邻居候选解的质量，重复上述过程，直到满足终止条件。

4.1 控制器布局搜索算法

为了搜索出控制器布局，首先需要满足网络中低度交换机的控制器部署需求，低度交换机是指周围邻居交换机数量较少的交换机。优先满足低度交换机的原因在于：由于该类交换机通常部署在网络边缘的缘故，其候选控制器数量通常远小于网络中的其他交换机。如果先满足其他交换机的控制器部署需求，最后考虑低度交换机的控制器部署需求，这些低度交换机的候选控制器集合中很可能没有足够仍有容量的控制器，用以满足低度交换机的控制器部署需求。

为了解决上述问题，本文通过为交换机设定分配顺序，用以保证每一个交换机都可以满足其控制器部署需求。交换机的分配顺序和交换机的候选控制器群(Candidate Controller Group, CCG)相关，候选控制器群为交换机i的候选控制器集合$\Gamma(s_i)$中满足容错需求r_i的控制器组合，每一个候选控制器群CCG_x都要包含r_i个控制器，且控制器之间的时延小于$d_{\max}^{\mathrm{cc}}$。以图1(b)为例，在图1(b)中，控制器c_1, c_2, c_3, c_4是交换机s_1, s_2, s_3, s_4和s_5的候选控制器，若每一个交换机的容错需求都为3，则交换机s_1, s_2, s_3, s_4和s_5的候选控制器群为$\{c_1,c_2,c_3\}$, $\{c_1,c_2,c_4\}$, $\{c_1,c_3,c_4\}$和$\{c_2,c_3,c_4\}$。

由于控制器容量存在上限，随着控制器的分配，候选控制器群中的控制器容量会动态的发生变化。一旦1个控制器容量达到上限，所有包含该容量达到上限控制器的候选控制群将无法再作为交换机的候选控制器群。故用有效候选控制器群(Valid Candidate Controller Group, VCCG)表示仍可以分配给交换机的候选控制器群，对于交换机s_i而言，其候选控制器群集合中的有效候选控制器群个数可以计算为

$$\Theta(s_i)=\sum_{x=1}^{\mathrm{num(CCG)}}\prod_{j\in \mathrm{CCG}_x}\gamma_j,\quad i\in S \tag{11}$$

其中$\mathrm{num(CCG)}$表示全部候选控制器群的个数，γ_j表示控制器c_j的容量是否已经达到上限，如式(12)所示，若控制器c_j的容量达到上限，则$\gamma_j=0$，反之$\gamma_j=1$。$\Theta(s_i)$表示随着网络中控制器容量变化时，交换机s_i的有效候选控制器群个数

$$\gamma_j=\begin{cases}0, & u_jp_j\le\sum\limits_{i\in S}q_iw_{ij}\\ 1, & u_jp_j>\sum\limits_{i\in S}q_iw_{ij}\end{cases},\quad j\in S \tag{12}$$

为了保证有效候选控制器群数量少的交换机仍可以分配到足够的控制器，故设定交换机s_i的有效候选控制器群个数越少，其对控制器的分配需求就越高。故交换机的分配顺序可以由式(13)计算

$$\mathrm{order}_i=\frac{1}{\Theta(s_i)},\quad i\in S \tag{13}$$

按照顺序对交换机分配控制器，故控制器布局搜索方法(Controller Assignment Algorithm, CAA)如算法1所示。

其中步骤1和步骤2初始化交换机的分配顺序；步骤4的$\mathrm{order}(S)$表示从交换机集合S中取出分配顺序最高的交换机，并将该交换机移出交换机集合

算法1 控制器布局搜索CAA

输入：网络拓扑G，交换机集合S

输出：控制器布局P

步骤1 计算每一个交换机的CCG

步骤2 利用式(13)计算交换机的分配顺序

步骤3 While (S)

步骤4 $s\leftarrow\mathrm{order}(S)$, $S\leftarrow S-\{s\}$

步骤5 随机选择1个CCG

步骤6 $P\leftarrow$ CCG

步骤7 Load $\leftarrow$ Load (P)

步骤8 利用式(11)更新交换机的分配顺序

步骤9 End While

步骤10 输出控制器布局P

S；步骤5和步骤6表示将所选择的控制器群放入控制器布局P中，并在步骤7中更新控制器负载；步骤9的终止条件为：交换机集合S为空，表明全部的交换机都已经分配了控制器；步骤10表示输出一种控制器布局P。

4.2 控制器布局更新机制

为了保证算法可以尽可能地遍历整个解空间，找到控制器布局的最优解，并避免出现陷入局部最优的情况，单纯依靠控制器布局搜索算法CAA显然是不充足的。故需要一种合适的控制器布局更新算法，在1个控制器布局的基础上按照更新原则生成新的控制器布局。在MFTCP问题中，更新算法的设计依赖于交换机的计算顺序，这是因为在MFTCP问题中交换机的计算顺序决定了控制器布局的结果。由于控制器的容量有限，1个交换机的控制器分配很可能会影响到其他的交换机分配。如图2所示，若交换机s_i先于交换机$s_{i'}$分配控制器，则控制器c_1, c_2和c_3的负载容量会被交换机s_i占用一部分，若控制器c_1, c_2和c_3中存在控制器c_2的负载已经达到最大值，其他原本可以选择控制器c_1, c_2, c_3作为控制器的交换机$s_{i'}$将无法选择容量已经达到上限的控制器c_2，而必须选择c_4作为其控制器。类似地若交换机$s_{i'}$先于交换机s_i分配控制器，交换机s_i则无法选择控制器c_1, c_2和c_3。显然的，交换机的分配顺序严重影响的控制器的分配结果。

故本文借鉴遗传算法中的变异算子，通过对交换机的计算顺序进行操作，进而完成控制器布局的更新。在对交换机的计算顺序进行更新时，采用随机迭代改进(randomized iterative improvement)对交换机的计算顺序进行调整。首先随机生成一个变异算子wp $\in [0, N]$，从当前布局的交换机计算顺序中选择第wp个交换机，并从该交换机的候选控制器群集合中随机选择一个新的CCG，由于CCG的变化，交换机的计算顺序也会发生变化。然后按照新的交换机分配顺序，重新为计算顺序在该交换机之后的交换机分配控制器。当网络中全部的交换机都分配完控制器时，控制器布局完成更新。更新过程示例如图3所示，若网络包含5个交换机，第l次迭代中交换机计算顺序为$s_1 \to s_2 \to s_3 \to s_4 \to s_5$，若在第$l+1$次迭代时，变异算子wp为3，则需要对交换机分配顺序第3位上的交换机至交换机分配顺序最后一位的交换机重新分配控制器群，此时，交换机计算顺序可以为$s_1 \to s_2 \to s_5 \to s_3 \to s_4$。

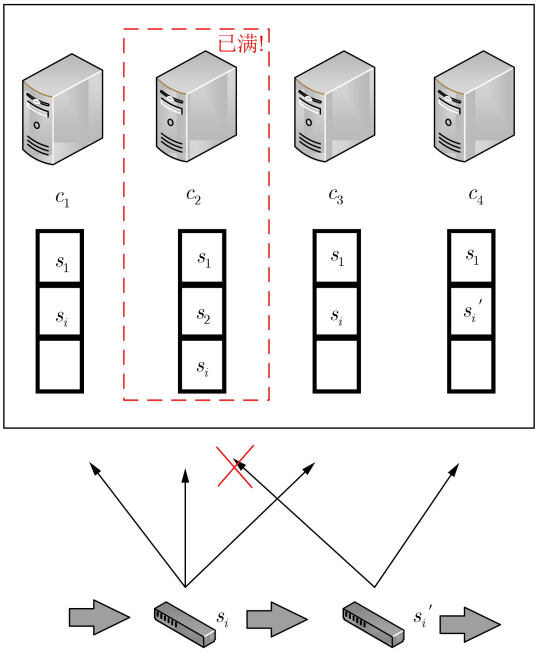


图 2 交换机计算分配顺序的影响

4.3 基于局部搜索算法的控制器布局算法

为了对新的控制器布局进行评估，本文采用部署的控制器数作为评估目标。结合控制器布局搜索算法和控制器布局更新机制，可以得到基于局部搜索算法的启发式算法LSCPA，可以表示为算法2。

在LSCPA中，步骤1和步骤2是初始化阶段，步骤3—步骤14是局部搜索算法的搜索过程，步骤4和步骤5表示更新过程，其中的$P_{l+1} \leftarrow \text{Update}(P_l)$表示控制器布局$P_l$按照突变算子wp更新成$P_{l+1}$, 步骤6表示LSCPA的评价指标，Num (.)可以评价控制器布局所需要的控制器数量。步骤7—步骤12表示一旦发现更优的控制器布局，则重置迭代次数，直到算法迭代次数超过最大迭代次数$l_{\max}$时，算法终止。

算法的复杂度分析：对于MFTCP问题而言，遍历其全部解空间的计算复杂度为$O(N!)$，但是由于在实际计算过程中，每一个交换机可以选择的控制器范围受限，实际上很多解是无效的，故LSCPA算法的计算复杂度远远小于遍历全部解空间的计算复杂度。由于CAA算法的计算复杂度为$O(N)$，每一次更新过程的最大复杂度为$O(N)$，故LSCPA算法的计算复杂度为$O\left(N^2 l_{\max}\right)$。

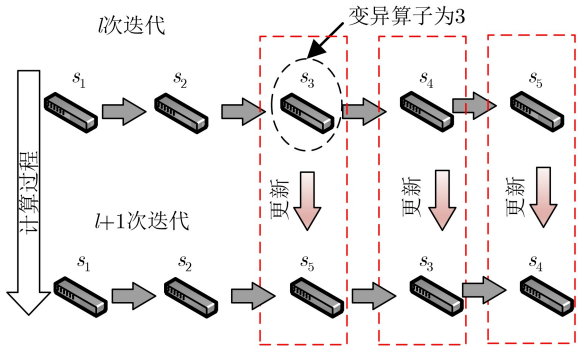


图 3 控制器布局更新实例

算法2 基于局部搜索算法的控制器布局算法LSCPA

输入：网络拓扑G，交换机集合S，最大迭代次数$l_{\max}$

输出：优化后的控制器布局$P*$

步骤1 运行CAA得到控制器初始布局P_0

步骤2 $l=0$

步骤3 While $(l \le l_{\max})$

步骤4 计算变异算子wp

步骤5 $P_{l+1} \leftarrow \text{Update}(P_l)$

步骤6 If $(\text{Num}(P_{l+1}) < \text{Num}(P_l))$

步骤7 $P* = P_{l+1}$

步骤8 $l=0$

步骤9 $P_l = P*$

步骤10 else

步骤11 $P* = P_l$

步骤12 $l = l+1$

步骤13 End If

步骤14 End While

步骤15 输出控制器布局$P*$

5 性能评估

本节对LSCPA算法进行实验评估，全部实验将在随机生成的3种网络中进行，3种网络分别为稀疏网络、中等网络和稠密网络，稀疏网络中，每一个节点的周围邻居节点数量为3～6个，节点间的时延范围为15～60 ms之间；中等网络中每一个节点的邻居节点数量为5～8个，节点间的时延范围为12～40 ms之间；稠密网络中每一个节点的邻居节点数量为7～10个，节点间的时延范围为10～20 ms之间。在进行实验仿真时，本文假定每一个交换机的容错需求都相同、产生的流量负载也相同，控制器的负载容量也相同。

为了在3种网络中验证算法的性能，LSCPA算法将分别在每一种网络中与随机(random)算法、ABC算法[12]及理论下界(Theoretical Lower Bound, TLB)进行比较，随机算法是指在网络中随机选择控制器的位置，直到网络中全部的交换机都分配到足够的控制器。ABC算法是一种控制器布局近似算法。理论下界法则是理想情况下，全部交换机生成的负载与控制器平均容量的比值。为了避免算法的随机性，实验中将运行LSCPA算法、随机算法、ABC算法与理论下界方法1000次，然后在不同网络中通过比较它们所使用控制器的评价数量，分别验证网络规模、容错需求和控制器容量对控制器布局的影响。

5.1 网络规模的影响

首先，为了验证算法在不同规模网络下的性能，本文通过改变3种网络中交换机数量变化网络的规模，然后分析不同规模下4种算法的性能。实验中，交换机的容错需求为3，控制器的容量为10，时延约束$d_{\max}^{cs}$和$d_{\max}^{cc}$为20 ms，网络中交换机的数量范围为[100,600]。

通过实验图4，可以发现算法在3种网络中的性能是存在差别的，3种网络中随机方法都是最差的布局方法，LSCPA算法所需要的控制器数量都要小于ABC算法。此外还能注意到，虽然3种网络需要控制器的理论下界相同，但是在实际部署中稀疏网络需要的控制器数量明显大于稠密网络，导致这种结果的原因在于，稀疏网络中由于交换机间的时延较大，使得每一个控制器在时延约束内可以控制的交换机数量较少，控制器的容量利用率较低，导致网络需要的控制器数量增加。

5.2 容错需求影响

为了验证网络容错需求对控制器布局的影响，固定网络中的交换机数量为100个，控制器的容量为10，时延约束$d_{\max}^{cs}$和$d_{\max}^{cc}$为20 ms。

通过图5，可以发现4种算法在网络中的控制器部署数量都随着容错需求的提高而快速增加，在网

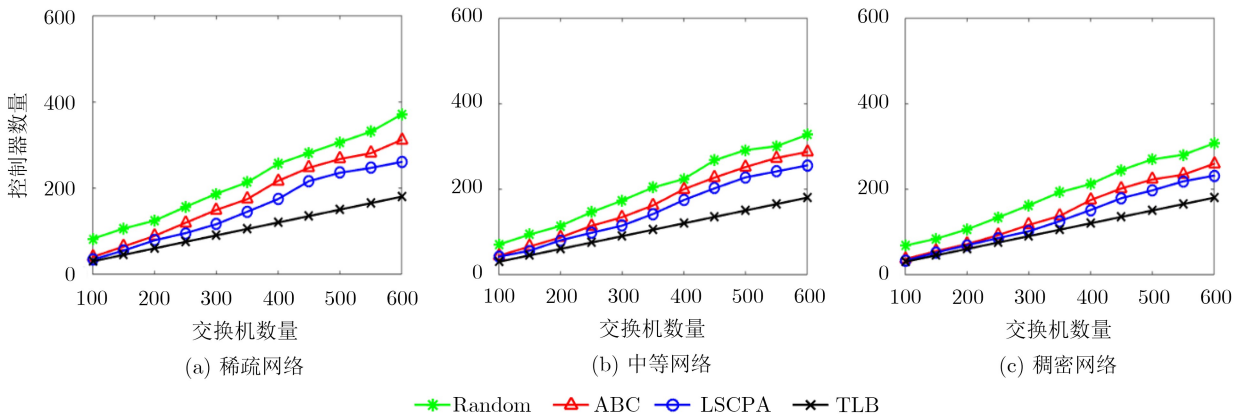


图 4 算法在不同规模网络中的需要部署的控制器数量

络的容错需求较低时(r= 1时)，ABC算法布局网络需求的控制器数量和LSCPA算法非常接近，这是由于ABC算法和LSCPA算法都可以找出容错需求为1的网络的最少控制器布局。但随着网络的容错需求不断提高(r> 1时)，LSCPA算法布局网络需求的控制器数量将会逐渐少于ABC算法，这是由于LSCPA算法可以通过随机迭代改进不断更新控制器布局方案，相对于ABC算法，可以避免布局结果陷入局部最优解的情况。

5.3 控制器容量影响

最后，验证控制器容量对网络中需要部署的控制器数量的影响。通过图6，可以发现随着控制器容量的变大，虽然3种算法部署的控制器数量都在变少，但LSCPA算法所需要的控制器数量仍然少于ABC算法和随机算法。但值得注意的是，当控制器的容量到达1个阈值时，3种算法所需要的控制器数量变化将会较为缓慢。导致这些现象的原因在于，每一个交换机和控制器的时延约束及控制器和控制器之间的时延约束极大的限制了每一个控制器的控制范围，即使控制器拥有大量的容量，由于时延的限制，控制器仍然无法控制更多的控制器。

6 结束语

软件定义网络的研究中，用如何低成本的在网络中部署控制器是一个非常重要的问题。本文针对容错控制平面进行分析，构造了以控制器部署数量为目标的最小覆盖容错模型，为了对模型进行求解，设计了一种启发式控制器布局算法，算法利用局部搜索的思路，通过邻居更新策略，产生邻居控制器布局，不断循环，最终找到满足终止条件的控制器布局。实验仿真表明算法不仅在不同规模的网络中都具有不错的性能，而且可以以更少的控制器数量完成满足网络容错需求的控制器部署。

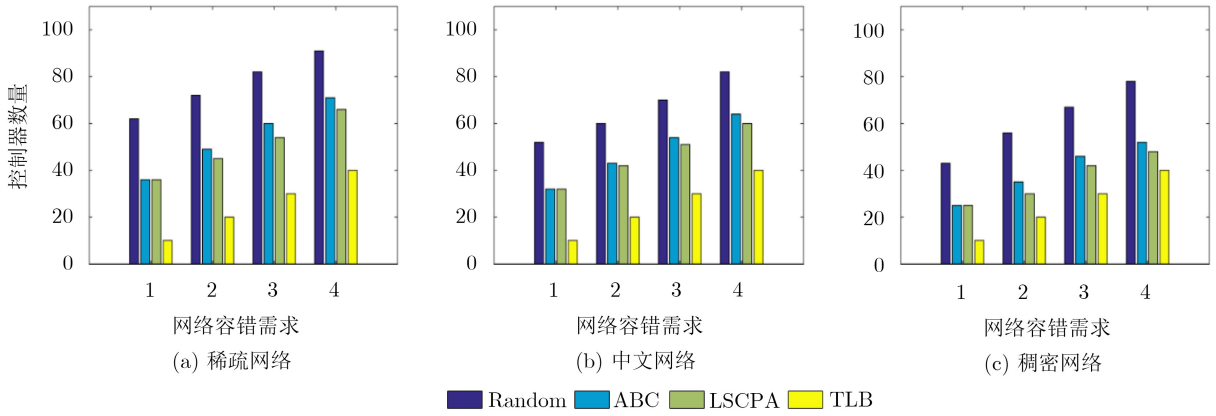


图 5 3种网络中容错需求对算法性能的影响

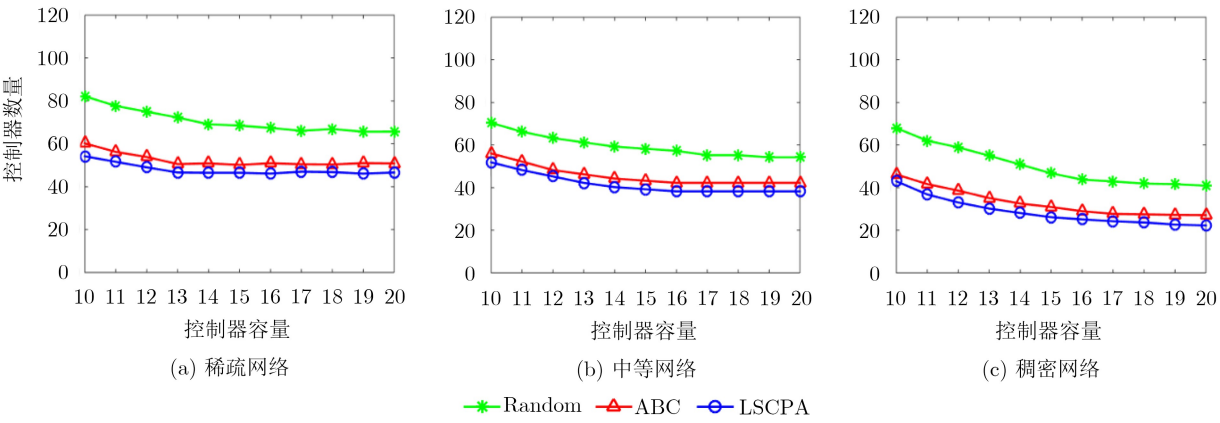


图 6 3种网络中控制器容量对算法性能的影响

参 考 文 献

[1] SCOTT-HAYWARD S, NATARAJAN S, and SEZER S. A survey of security in software defined networks[J]. *IEEE Communications Surveys & Tutorials*, 2016, 18(1): 623–654. doi: 10.1109/COMST.2015.2453114.

[2] KILLI B P R and RAO S V. Optimal model for failure foresight capacitated controller placement in software defined networks[J]. *IEEE Communications Letters*, 2016, 20(6): 1108–1111. doi: 10.1109/LCOMM.2016.2550026.

[3] LI He, LI Peng, GUO Song, *et al*. Byzantine–resilient secure

software–defined networks with multiple controllers in cloud[J]. *IEEE Transactions on Cloud Computing*, 2014, 2(4): 436–447. doi: 10.1109/TCC.2014.2355227.

[4] 李军飞, 胡宇翔, 邬江兴. 基于拜占庭容错提高SDN控制层可靠性的研究[J]. 计算机研究与发展, 2017, 54(5): 952–960. doi: 10.7544/issn1000-1239.2017.20160055.
LI Junfei, HU Yuxiang, and WU Jiangxing. Research on improving the control plane's reliability in SDN based on byzantine fault-tolerance[J]. *Journal of Computer Research and Development*, 2017, 54(5): 952–960. doi: 10.7544/issn1000-1239.2017.20160055.

[5] WOOD T, SINGH R, VENKATARAMANI A, *et al*. ZZ and the art of practical BFT execution[C]. The 6th Conference on Computer Systems, Salzburg, Austria, 2011: 123–138. doi: 10.1145/1966445.1966457.

[6] ROS F J and RUIZ P M. On reliable controller placements in software–defined networks[J]. *Computer Communications*, 2016, 77: 41–51. doi: 10.1016/j.comcom.2015.09.008.

[7] 史久根, 谢熠君, 孙立. 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. doi: 10.11999/JEIT181053.
SHI Jiugen, XIE Yijun, and SUN Li. Multi–controller placement strategy based on latency and load in software defined network[J]. *Journal of Electronics & Information Technology*, 2019, 41(8): 1869–1876. doi: 10.11999/JEIT181053.

[8] 史久根, 郑伟, 贾坤荥, 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. doi: 10.11999/JEIT170464.
SHI Jiugen, ZHU Wei, JIA Kunying, *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. doi: 10.11999/JEIT170464.

[9] KILLI B P R and RAO S V. Controller placement with planning for failures in software defined networks[C]. 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, Bangalore, India, 2016: 1–6. doi: 10.1109/ANTS.2016.7947795.

[10] ELDEFRAWY K and KACZMAREK T. Byzantine fault tolerant Software–Defined Networking (SDN) controllers[C]. 2016 IEEE 40th Annual Computer Software and Applications Conference, Atlanta, USA, 2016: 208–213. doi: 10.1109/COMPSAC.2016.76.

[11] JIMÉNEZ Y, CERVELLÓ-PASTOR C, and GARCIA A J. On the controller placement for designing a distributed SDN control layer[C]. 2014 IFIP Networking Conference, Trondheim, Norway, 2014: 1–9. doi: 10.1109/IFIPNetworking.2014.6857117.

[12] XIE Junjie, GUO Deke, ZHU Xiaomin, *et al*. Minimal fault–tolerant coverage of controllers in IaaS datacenters[J]. *IEEE Transactions on Services Computing*, 2017, 36(3): 1–14. doi: 10.1109/TSC.2017.2753260.

[13] GAREY M R and JOHNSON D S. Computers and INTRACTABILITY: A GUIDE to the Theory of NP–Completeness[M]. New York: W. H. Freeman and Company, 1979: 199–201.

吴 奇：男，1991年生，博士生，研究方向为网络空间安全.
陈鸿昶：男，1964年生，教授，研究方向为网络空间安全、大数据分析.

责任编辑：余 蓉