[文章编号] 1003－4684(2025)01-0043-06

基于 K-means++和粒子群算法的 SDN 多控制器部署方法

徐 慧，吴美连

（湖北工业大学计算机学院，湖北 武汉 430068）

[摘 要] 针对软件定义网络中的多控制器部署问题，首先通过 K-means++算法对网络节点聚类，得到网络中初始控制域和控制器位置，然后使用粒子群算法以最小化时延和负载均衡为优化目标，多个粒子并行搜索最优解，进一步优化控制域和控制器位置。在小、中、大型网络拓扑上与随机算法、K-means++算法、粒子群算法的多控制器部署方法比较，仿真结果表明，在中小型网络中，比其他 3 种算法在平均传播时延和负载均衡上更加稳定且时延更低，在大型网络中，平均传播时延，最坏传播时延和控制器的负载均衡上均优于其他 3 种算法。

[关键词] 软件定义网络；多控制器部署；K-means++；粒子群算法；时延；负载均衡

[中图分类号] TP393 [文献标识码] A

软件定义网络[1]（software defined networking，SDN）核心理念是将网络中的控制平面与数据平面分离，控制平面通过单一的控制器管理网络中的所有设备，但是随着网络规模的扩大，交换机数量的增加，单一的控制器在处理越来越多的请求时不堪重负，从而无法保证高效的性能。有学者提出了一种分层式设计的多控制器体系结构解决这些问题[2]，例如 Kandoo[3]，此种体系结构由多个控制器构成，且控制器间可以相互通信，随着这一体系的诞生，多个控制器如何合理放置在控制平面上成为了亟待解决的问题。多控制器放置问题[4]（controller placement problem，CPP）是一个 NP 难问题，在 SDN 网络中部署多个控制器，针对设定的优化目标，选出最合适的控制器数量和最佳放置位置，确定控制器与其管理的交换机的映射。

Heller[5]等首次提出了 SDN 网络中的 CPP，Torkamani[6]等提出了一种基于 GSO 算法的多控制器部署方法并应用于不同类型网络中，实现了最小的时间延迟，没有讨论控制器的负载情况。Singh[7]等提出一种基于 VBO 的方法，实现了可靠的 CPP，但是只考虑了总平均时延。马勤[8,9]等提出的方法以负载均衡为主要目标进行多控制器的部署，实现了网络中控制器与交换机的最佳分配问题，但没有考虑控制器间的时延。黄尔杰[10,11]等提出了以最小化时延为优化目标的控制器部署方法，但未考虑控制器的负载。Wang[12]等提出了一种优化的 K-means 算法，减少了交换机与控制器之间的延迟，但未考虑控制器间的延迟和控制器的负载。

综上，部分研究以时延为指标，仅考虑交换机与控制器之间的时延，而忽略了控制器间的时延。另外部分研究以负载均衡为指标，忽略了控制器的负载，未能从多个方面综合考虑。故本文综合考虑交换机与控制器之间以及控制器间的时延和负载均衡对 CPP 的影响，提出基于 K-means++和粒子群算法（particle swarm optimization algorithm，PSO）的 SDN 多控制器部署方法，通过 K-means++算法划分初始控制域并确定控制器位置，再引入 PSO 算法保证网络链路连通性，同时以最小化时延和负载均衡为优化目标，调整控制域，得出最佳的控制器部署位置和控制域。

1 问题描述及模型

1.1 问题描述

控制器作为 SDN 网络中控制平面的核心，数据平面的交换机转发数据包时将流请求发送给相应的控制器，控制器收到请求后下发流表给交换机，在这个过程中，SDN 网络的性能主要由两个方面决定，一是交换机与控制器之间的时延以及控制器间的时延，其影响着数据的传输速度；二是控制器收到请求时承载的负载容量，其决定着数据传输的稳定性，这

[收稿日期] 2022－10－20
[基金项目] 国家自然科学基金(61602162)
[第一作者] 徐 慧(1983—)，女，湖北武汉人，工学博士，湖北工业大学教授，研究方向为网络管理与安全。

两方面共同决定着 SDN 的网络性能。

在 SDN 网络的数据传输中，若通信设备之间产生的传播时延过长，则会影响网络的整体性能，因此降低交换机与控制器以及控制器间通信产生的平均时延和最坏时延能保证网络数据传输的高效性。

负载均衡是衡量网络性能的重要指标。如果控制器超过其自身负载容量，就会导致网络事件的响应时延增加和控制器故障率增加等问题[13]，造成网络的不稳定性。

1.2 多控制器部署模型

将 SDN 网络拓扑表示为一个无向图 $G=(V,E)$，其中，V 表示网络中节点的集合，E 表示物理链路集合。S 为交换机的集合，$S=\{s_1,s_2,\cdots,s_N\}$，交换机个数为 N。C 为控制器的集合，$C=\{c_1,c_2,\cdots,c_k\}$，控制器个数为 k，网络被划分为 k 个控制域，每个控制域的交换机由一个控制器控制。网络中物理设备的传播速率取光纤的传播速率 $v=3\times10^8\,\mathrm{m/s}$。通过球面半正矢公式(1)计算两个节点之间的传播路径长度，其中 φ_2,φ_1 表示网络节点的纬度，λ_2,λ_1 表示网络节点的经度，根据网络拓扑的经纬度计算每条物理链路的距离

$$d=2r\arcsin\cdot\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \tag{1}$$

指标 1：平均传播时延 T_{avg} 是网络中交换机到控制器之间平均传播时延和控制器间平均传播时延之和。

$$T_{\mathrm{avg}}=T_{\mathrm{cs_avg}}+T_{\mathrm{cc_avg}} \tag{2}$$

其中，$T_{\mathrm{cs_avg}}$ 表示交换机与控制器之间的平均传播时延，即 SDN 网络中所有控制器与它们所管理的交换机之间的平均传播时延，如式(3)所示，$d_{i,j}$ 是交换机与控制器之间的最短路径距离。$T_{\mathrm{cc_avg}}$ 表示控制器间的平均传播时延，如式(4)所示，在有 k 个控制器的网络中，控制器间的链路数为 $k(k-1)$，$d_{j,k}$ 代表任意一对控制器间的最短路径距离。

$$T_{\mathrm{cs_avg}}=\frac{1}{N}\times\frac{\sum_{si\in S}\sum_{cj\in C}d_{i,j}}{v} \tag{3}$$

$$T_{\mathrm{cc_avg}}=\frac{1}{k(k-1)}\times\frac{\sum_{cj,ck\in C}d_{j,k}}{v} \tag{4}$$

指标 2：最坏传播时延 T_{worst} 是交换机到控制器之间的最大传播时延和控制器间最大传播时延之和。

$$T_{worst}=T_{\mathrm{cs_worst}}+T_{\mathrm{cc_worst}} \tag{5}$$

其中，$T_{\mathrm{cs_worst}}$ 表示交换机与控制器的最坏传播时延，即 SDN 网络中控制域内控制器与其所管理交换机之间最小时延的最大值，如式(6)所示。$T_{\mathrm{cc_worst}}$ 表示控制器间的最坏传播时延，即 SDN 网络中任意一对控制器间最小时延的最大值，如式(7)所示。

$$T_{\mathrm{cs_worst}}=\frac{\max\sum_{si\in S,cj\in C}d_{i,j}}{v} \tag{6}$$

$$T_{\mathrm{cc_worst}}=\frac{\max\sum_{cj,ck\in C}d_{j,k}}{v} \tag{7}$$

指标 3：负载均衡参数 BL_P 衡量控制器间的负载差异，控制域内所有交换机产生的流请求之和决定控制器负载[14]。若交换机 S_j 在控制域 CS_i 内产生的流请求为 $\lambda_{i,j}$，则控制域的控制器负载为

$$CL_i=\sum_{S_j\in CS_i}\lambda_{i,j} \tag{8}$$

其中，控制域 CS_i 为控制器 C_i 管理的交换机集合，通过控制器的最大负载与最小负载的比值定义负载均衡参数

$$BL_P=\frac{CL_{\max}}{CL_{\min}},\max,\min\in C \tag{9}$$

$$BL_P\geqslant1 \tag{10}$$

$CL_{\max}$ 为网络中最大控制器负载，$CL_{\min}$ 为网络中最小控制器负载。BL_P 值越接近 1，各个控制器的负载就越接近。

基于上述指标的多控制器部署问题是一个带约束的多目标优化求解问题。为简化该多目标规划问题，使指标最小化，本文提出的多控制器部署模型为

$$F_{\min}=\theta\cdot T_{\mathrm{avg}}+\mu\cdot T_{\mathrm{worst}}+\rho\cdot BL_P \tag{11}$$

$$\theta+\mu+\rho=1,(0\leqslant\theta,\mu,\rho\leqslant1) \tag{12}$$

$$CL_i<\mu_i,i\in C \tag{13}$$

其中，式(11)为需要优化的目标函数，且 θ,μ,ρ 表示不同指标的权重，式(12)表示 θ,μ,ρ 的取值范围，式(13)表示控制器的负载应小于其最大负载 μ_i。

2 多控制器部署方法

2.1 初始控制域划分和控制器部署

在 SDN 网络中部署多个控制器，每个控制器管理其周边的若干个交换机，在网络中形成多个控制域，因此控制器部署过程可以被描述为样本点的聚类过程，本文假设一个控制器管理一个控制域，控制域类比为聚类，控制器的位置类比为聚类中心，交换机的位置类比为聚类的其他样本点。由于 K-means 算法随机选取样本点作为初始聚类中心，极大影响了算法的收敛速度，导致最终聚类结果差异较大，而 K-means++算法初始聚类中心的选取进行了改进，通过计算每个样本点被选为聚类中心的概率来确定聚类中心，加快了算法的收敛速度，因此本文采用 K-means++算法。

对于第一个聚类中心的选取，选择样本节点间

距离相对较远的点作为第一个聚类中心,样本节点间的距离采用欧式距离计算,dis 为每个样本节点与当前聚类中心的最短距离。

$$\mathrm{dis} = d(v_j, c_i) \tag{14}$$

其中,c_i 为聚类中心,v_j 为其他样本节点,采用轮盘法选择其他初始聚类中心,v_j 被选为下一个聚类中心的概率为

$$p(v_j) = \frac{\mathrm{dis}^2}{\sum \mathrm{dis}^2} \tag{15}$$

其中,$\sum \mathrm{dis}^2$ 为所有样本点到聚类中心的距离之和,第 n 个样本节点被选为下一个聚类中心的累积概率为

$$p(v_n) = \sum_{j=1}^{n} p(v_j) \tag{16}$$

为了便于解释累积概率对聚类中心的选择过程,可将累积概率转换为对应的概率区间,

在区间 [0,1] 之间生成一个均匀分布的伪随机 r,判断 r 所在的概率区间位置,选出对应的样本节点作为聚类中心。确定 k 个初始聚类中心后,使用 K-means 算法,选出 k 个聚类中心,不断迭代,直到聚类中心不再改变,最终选出 k 个聚类中心的位置和聚类,即控制器部署位置和控制域。

2.2　控制域和控制器位置的优化

经过 K-means++算法对网络的控制域划分和初始控制器部署后,控制域划分存在两点不足之处,一是忽略了网络实际链路连通性,故使用 dijkstra 算法计算拓扑图中两个节点之间的最短路径,得到最短路径矩阵 CP,保证链路的连通性。二是聚类算法计算的距离是欧式距离,但是在实际网络中,网络节点间是物理距离,因此通过球面半正矢公式(1)计算每条链路的实际距离,得到距离矩阵 DP。对 SDN 网络中平均传播时延、最坏传播时延、负载均衡参数这 3 个指标,采用粒子群算法(particle swarm optimization algorithm,PSO)进一步优化控制域和控制器位置。

粒子群算法[15]是一种以群体迭代为基础的算法,在整体空间内,粒子搜寻时仅仅跟随最优粒子。由于 PSO 算法具有良好的全局搜索能力,对于多控制器部署的多目标问题能更好地求出最优解,因此,在控制域和控制器位置的优化过程中,对建立的多控制器部署模型,采用 PSO 算法找到可行的部署方案,每个粒子都表示一次可行的部署方式。PSO 算法以初始控制器的位置为粒子的初始位置,式(11)作为适应度函数,不断迭代,粒子群中的粒子搜寻合适的交换机位置作为新的控制器位置,确定控制器的最佳位置。

首先初始化粒子群,粒子将初始控制器的位置设为当前最优位置 $P_{p_{\mathrm{best}}}$,设置粒子的初始和飞行速度,粒子的搜寻路线为网络的链路,设置最大粒子群总数为网络拓扑节点数。PSO 算法根据输入的控制器集合 C,控制器管理的交换机集合 S_c,对初始划分的控制域,选择负载最大的控制器,找出其管理的控制域边缘位置交换机节点为可移动节点,以距离和 $F_{\min}$ 为指标不断调整可移动交换机的位置,划分到相邻最近的控制域,进一步调节控制域间的最小时延和控制器的负载。以式(11)作为适应度函数,找到粒子的当前个体极值 p_{best_i} 和粒子群的当前最优解 g_{best_i},更新粒子的速度和位置

$$v_i = \omega \times v_i + c_1 \times \mathrm{rand}() \times (p_{\mathrm{best}_i} - x_i) + c_2 \times \mathrm{rand}() \times (g_{\mathrm{best}_i} - x_i) \tag{17}$$

$$x_i = x_i + v_i \tag{18}$$

其中,ω 表示惯性因子,c_1 和 c_2 称为增速常数,一般取值为 $c_1 = c_2 \in [0,4]$,rand() 是区间 [0,1] 上的随机数。

若 $F_{\min}(P_{\mathrm{position}}) < F_{\min}(P_{p_{\mathrm{best}}})$,将粒子的当前位置 P_{position} 设为局部最优位置 $P_{p_{\mathrm{best}}}$,若 $F_{\min}(P_{p_{\mathrm{best}}}) < F_{\min}(P_{g_{\mathrm{best}}})$,将粒子的局部最优位置 $P_{p_{\mathrm{best}}}$ 设为全局最优位置 $P_{g_{\mathrm{best}}}$,直到最大迭代次数,得到最终的控制器部署位置 $P_{g_{\mathrm{best}}}$,控制域的交换机集合 S_c。

2.3　基于 K-means++和粒子群算法的多控制器部署方法流程

基于 K-means++和粒子群算法的多控制器部署方法首先采用 K-means++算法对网络的控制域进行划分,并初步部署控制器位置,然后采用 PSO 算法以目标函数作为适应度函数,多个粒子并行搜索最优解,进一步优化控制域和控制器的位置,以达到网络中时延最小和控制器负载均衡的目的。多控制器部署方法流程如图 1 所示。算法的具体步骤如下所示。

Step 1:根据公式(14)选出样本节点间距离最大的节点作为第一个聚类中心,再根据公式(15)和公式(16)计算其余样本节点被选为聚类中心的概率,确定 k 个初始聚类中心;

Step 2:计算每个样本节点到各个聚类中心的最短距离 dis,将样本节点划分到距离最近的聚类中心所对应的类中;

Step 3:计算每个类中所有样本节点的均值,更新每个类的聚类中心,直到聚类中心位置不再改变,得到 k 个初始控制域和控制器位置;

Step 4:使用 dijkstra 算法计算网络节点间最短路径矩阵 CP,根据公式(1)计算网络链路的距离矩

阵 DP；

Step 5：对负载最大的控制域，将距离控制域边缘最近的交换机分配给相邻的控制域，更新控制域的交换机集合 S_c ；

Step 6：设置粒子的初始位置为初始控制器的位置，设置粒子的初始和飞行速度，设置最大粒子群总数为网络拓扑节点数；

Step 7：根据公式(11)计算适应度函数值 $F_{\min}$，找到粒子的当前个体极值 p_{best_i} 和粒子群的当前最优解 g_{best_i} ；

Step 8：根据公式(17)和公式(18)分别更新粒子 P 的速度和位置，更新粒子的局部最优位置 $P_{p_{\text{best}}}$ 和全局最优位置 $P_{g_{\text{best}}}$ ；

Step 9：若达到最大迭代次数，则输出控制器部署位置 $P_{g_{best}}$ ，控制域的交换机集合 S_c ，否则返回到 Step 5 继续迭代。

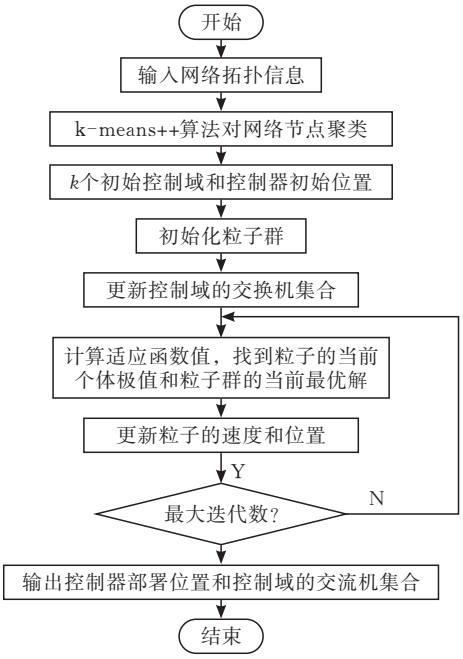


图 1 K-means++和粒子群算法的多控制器部署方法

3 仿真实验与分析

3.1 实验设计

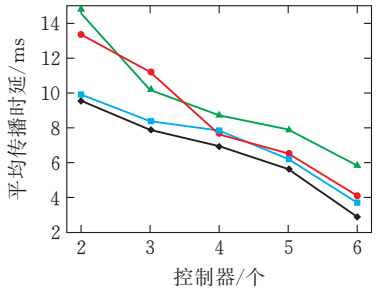
为验证基于 K-means++和粒子群算法的多控制器部署方法的有效性，在实际网络拓扑中对该方法进行仿真验证。仿真使用 Pycharm 软件对多控制器部署模型进行建模，该模型为最小化时延和负载均衡的多目标优化模型，采用 K-means++算法和 PSO 算法对模型求最优解，得出最佳的控制器部署位置和控制域。实验选择网络拓扑动物园的 Savvis 网络、IRIS 网络和 Colt Telecom 网络，分别为小、中、大型网络型拓扑，网络拓扑信息如表 1 所示。假设实验中控制器的性能相同，交换机的性能相同，控制器负载最大值 μ 设置为 1700，交换机请求 λ 设置为 90，最大迭代次数设置为 150，加权因子 θ，μ，ρ 分别设置为 0.3、0.2、0.5。

表 1 网络拓扑信息

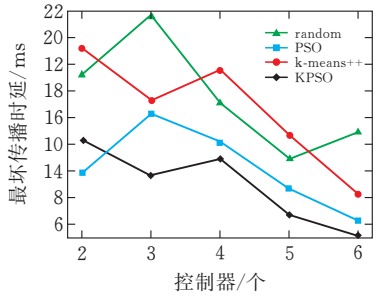
网络拓扑	节点数	链路数
Savvis	20	26
IRIS	51	64
Colt Telecom	153	177

3.2 实验结果分析

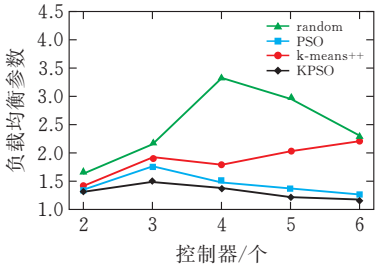
将本文算法与随机算法、K-means++算法、PSO 算法在平均传播时延、最坏传播时延和负载均衡参数这 3 个性能指标上进行比较，图 2、图 3、图 4 分别为 Savvis 网络、IRIS 网络、Colt Telecom 网络的性能指标，为方便对实验结果分析进行表述，将基于 K-means++与粒子群算法的多控制器部署方法简称为 KPSO 算法。



(a)平均传播时延



(b)最坏传播时延

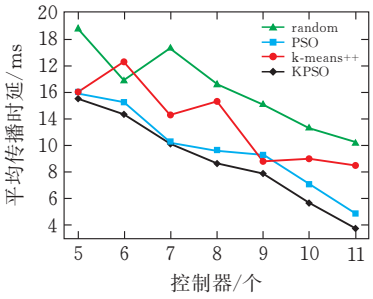


(c)负载均衡参数

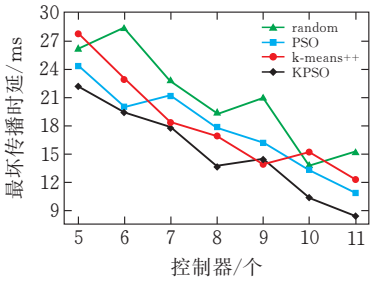
图 2 Savvis 网络性能指标

实验 1：平均传播时延。实验结果如图 2a、图 3a 和图 4a 所示，可以看出随着控制器数量的增加，平均传播时延均在降低，图 2a 中，随机算法的平均传播时延均高于其他 3 种算法，K-means++算法的波动较大，综合图 2a 和图 3a 来看，PSO 与 KPSO 算

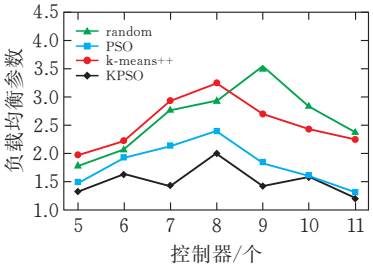
法的折线趋势相对平稳，且 KPSO 算法的平均传播时延始终低于 PSO 算法，说明在中小型规模网络中，KPSO 算法在平均传播时延上更加稳定且时延更低。在图 4a 中，控制器个数为 15，16，19 时，K-means++算法的平均传播时延略低于 KPSO 算法，这是由于随着网络规模增大，网络节点增多，K-means++算法的聚类特性更能表现出来，但 K-means++算法在考虑传播时延时忽略了链路连通性，故在实际网络环境中部署控制器时，KPSO 算法更具备现实合理性。



(a)平均传播时延



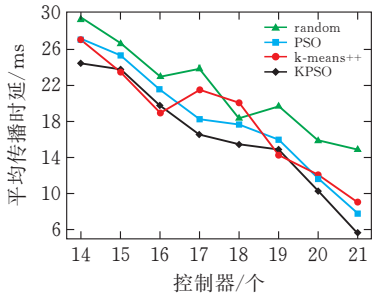
(b)最坏传播时延



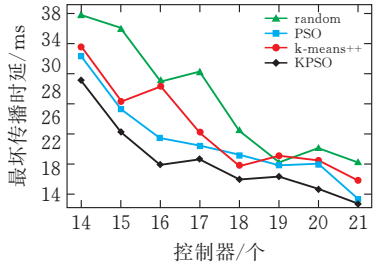
(c)负载均衡参数

图 3　IRIS 网络性能指标

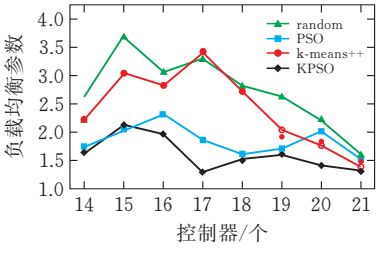
实验 2：最坏传播时延。实验结果如图 2b、图 3b 和图 4b 所示，可以看出最坏传播时延整体趋势随着控制器数量增加而降低。图 2b 中，控制器个数为 2 时，KPSO 算法的最坏传播时延高于 PSO 算法，图 3b 中，控制器数量为 6 时，KPSO 算法的最坏传播时延略高于 PSO 算法，分析其原因，PSO 算法在中小型规模网络中选取初始控制器位置时，控制器间距离较近，部署的控制器个数越少，则控制器间的最长路径距离更短。从图 4b 中可以看出在控制器个数相同的情况下，KPSO 算法在最坏传播时延上均低于其他三种算法，综合来看，KPSO 算法均适用于小、中、大型网络上的多控制器部署。



(a)平均传播时延



(b)最坏传播时延



(c)负载均衡参数

图 4　Colt Telecom 网络性能指标

实验 3：负载均衡。实验结果如图 2c、图 3c 和图 4c 所示，KPSO 算法的负载均衡参数均低于其他 3 种算法。图 1c 中，KPSO 算法的负载均衡参数趋势更平稳，更加接近于 1，最大值与最小值的差值为 0.33，说明在小规模网络中，对比其他 3 种算法，KPSO 算法的负载均衡稳定性最好。图 3c 中，控制器个数为 8 时，K-means++算法、PSO 算法、KPSO 算法的负载均衡参数均上升，随后又下降，分析原因可能是此时三种算法均对控制域划分得不均衡。

图 4c 中，控制器个数少于 17 时，K-means++算法的负载均衡参数大致呈现上升趋势，这是因为大规模网络中节点较多，控制器个数越少，控制域也越少，其管理的交换机个数更容易出现不均衡情况，随着控制器个数增加，该情况会变好，控制器管理的交换机个数更加均衡。

4　结论

以最小化时延和负载均衡为优化目标，建立多控制器部署模型，提出了一种基于 K-means++和粒子群算法的 SDN 多控制器部署方法，并对模型进行求解。该方法将 K-means++算法的分类特点与

PSO 算法寻找目标函数最优解的特点相结合，能够有效地划分控制域，保证网络的连通性，实现时延最小化和负载均衡的多控制器部署。实验结果表明，与随机算法、K-means++算法和 PSO 算法相比，本文提出的方法在不同规模的网络中，实现了最小的平均传播时延和最坏传播时延，且保证了控制器的负载均衡。

[参 考 文 献]

[1] ZHANG J LAN, Y. HU, et al. Survey on scalability of control plane in Software-Defined networking[J]. Journal of Software, 2018, 29(01):160-175.

[2] KILLI B, RAO S. Controller placement in software defined networks: a comprehensive survey[J]. Computer networks, 2019 (163): 106883.

[3] YEGANEH S, YASHAR G. KANDOO. A framework for efficient and scalable offloading of control applications[C]. In Proc. 1st Workshop on Hot Topics in Software Defined Networking, 2012: 19-24.

[4] 贾吾财，吕光宏，王桂芝，等.SDN 多控制器放置问题研究综述[J].计算机科学，2020，47(07)：206-212.

[5] HELLER B, SHERWOOD R, MCKEOWN N. The controller placement problem [J]. ACM SIGCOMM Computer Communication Review, 2012, 42: 473-478.

[6] TORKAMANI A, SAHAND. A new GSO based method for SDN controller placement[J]. Jahanshahi, Computer Communications, 2020, 163: 91-108.

[7] SINGH A, MAURYA S. Heuristic approaches for the reliable SDN controller placement problem[J]. Transactions on Emerging Telecommunications Technologies, 2020, 31: e3761.

[8] 马勤，李志芳.SDN 中一种多控制器负载均衡方案[J].西南师范大学学报(自然科学版)，2021，46(07)：114-119.

[9] 唐铁斌，刘炜.基于改进樽海鞘群算法的 SDN 控制器部署算法[J].计算机应用与软件，2021，38(12)：291-297.

[10] 黄尔杰，尚秋峰.基于时延的软件定义网络控制器部署策略研究[J].科技与创新，2019(03)：68-69.

[11] QI Y, WANG D, YAO W, et al. Towards multi-controller placement for SDN based on density peaks clustering[C]// ICC 2019－2019 IEEE International Conference on Communications (ICC). IEEE, 2019: 1-6.

[12] WANG G, ZHAO Y, HUANG J, et al. A K-means-based network partition algorithm for controller placement in software defined network[C]. 2016 IEEE International Conference on Communications (ICC), 2016: 1-6.

[13] YAO G, BI J, GUO L. On the cascading failures of multi-controllers in software defined networks [C]. 2013 21st IEEE International Conference on Network Protocols (ICNP), 2013: 1-2.

[14] YAO G, BI J, LI Y, et al. On the capacitated controller placement problem in software defined networks[J]. IEEE Communications Letters, 2014, 18: 1339-1342.

[15] KENNEDY J, EBERHART R. Particle swarm optimization[C]. In Proceedings of the 4th IEEE International Conference on Neural Networks. Piscataway: IEEE ServiceCenter, 1995: 1942-1948.

Multi-controller Deployment Method Based on K-means++ and Particle Swarm Optimization in SDN

XU Hui, WU Meilian

(*School of Computer Science, Hubei Univ. of Tech., Wuhan* 430068, *China*)

Abstract: To solve the problem of multi-controller deployment in software defined networking, a multi-controller deployment method based on K-means++ and particle swarm optimization is proposed. Firstly, the K-means++ algorithm is used to cluster the network nodes to obtain the initial control domain and controller position in the network, and then the particle swarm optimization algorithm is used to minimize delay and load balancing as the optimization goal, and multiple particles search for the optimal solution in parallel to further optimize the control domain and controller position. Compared with the multi-controller deployment methods of random algorithm, K-means++ algorithm and particle swarm optimization algorithm in small, medium and large scale network topology, simulation results show that it is more stable and less delay than the other three algorithms in average propagation delay and load balancing in small and medium scale network, and average propagation delay, the worst propagation delay and the load balancing of the controller are better than the other three algorithms in large scale network.

Keywords: Software Defined Networking; multi-controller deployment; K-means++; particle swarm optimization algorithm; delay; load balancing

[责任编校：裴 琴]