软件定义网络中面向时延和负载的多控制器放置策略

史久根 谢熠君* 孙 立 郭 胜 刘雅丽
(合肥工业大学计算机与信息学院 合肥 230009)

摘 要：在多控制器管理的软件定义网络(SDN)中，时延和负载是控制器放置问题(CPP)要考虑的重要因素。该文以降低控制器之间的传播时延、流请求的传播时延和排队时延、均衡控制器间负载为目标，提出一种控制器放置及动态调整的策略，其中包括用于初始控制器放置的负载均衡算法(BCRA)和遗传算法(GA)，用于动态调整控制器负载的在线调整算法(ADOA)。以上算法均考虑网络连通性。仿真结果表明：在初始控制器放置时，在保证流请求的传播时延、排队时延和控制器传播时延较低的情况下，BCRA部署在中小型网络中时，其负载均衡性能与GA相近且优于k-center和k-means算法；GA部署在大型网络中时，与BCRA, k-center和k-means算法相比，使得负载均衡率平均提高了49.7%。在动态情况下，与现有动态调整算法相比，ADOA可以保证较低排队时延和运行时间的同时，仍能使负载均衡参数小于1.54。

关键词：软件定义网络；控制器放置；负载均衡；网络时延；动态调整
中图分类号：TP393.3 **文献标识码**：A **文章编号**：1009-5896(2019)08-1869-08
DOI: 10.11999/JEIT181053

Multi-controller Placement Strategy Based on Latency and Load in Software Defined Network

SHI Jiugen XIE Yijun SUN Li GUO Sheng LIU Yali
(*School of Computer and Information*, *Hefei University of Technology*, *Hefei* 230009, *China*)

Abstract: In Software Defined Networks (SDN), latency and load are important factors for Controller Placement Problem (CPP). To reduce the transmission latency between controllers, the propagation latency and queuing latency of flow requests, and balance the controller load, a strategy on how to place and adjust the controller is proposed. It mainly includes Genetic Algorithm (GA) and Balanced Control Region Algorithm (BCRA) which are used to place the initial controller and one Algorithm of Dynamic Online Adjustment (ADOA), that is an online adjusting algorithm in term dynamic controlling. The above algorithms are all based on the network connectivity. The simulation results show that in initial controller placement situation, under the premise of guaranteeing the lower propagation latency, queue latency and controller transmission latency of flow request, when BCRA is deployed in small and medium-sized networks, its load balancing performance is similar to that of GA and superior to k-center and k-means algorithm; When GA is deployed in large networks, compared with BCRA, k-center and k-means, the load balancing rate increases averagely 49.7%. In the dynamic situation, ADOA can guarantee lower queuing delay and running time, and can still make the load balance parameter less than 1.54.

Key words: Software Defined Network (SDN); Controller placement; Load balancing; Network latency; Dynamic adjustment

1 引言

软件定义网络(Software Defined Network，SDN)作为一种新型的网络架构，其思想是将网络的控制层与数据层解耦，从而利用集中式的控制器对底层设备可编程化管理，进而实现对网络资源的灵活调配。随着SDN的应用，带来了一些新的挑战，如控制器放置问题(Controller Placement Problem, CPP)：即给定一个网络，确定控制器的数量及位置。

Heller等人[1]最先指出CPP是一个NP难题，以

收稿日期：2018-11-20；改回日期：2019-04-09；网络出版：2019-04-23
*通信作者： 谢熠君 2017110977@mail.hfut.edu.cn
基金项目：国家重大科学仪器设备开发专项(2013YQ030595)
Foundation Item: The National Major Scientific Instruments Development Project (2013YQ030595)

平均时延和最大时延为指标，选择控制器位置，并指出一个随机选择的放置方案比最优方案会造成更大的时延；Lange等人[2]考虑控制器间时延，并提出一种适用于广域网的启发式放置算法；Zhang等人[3]以提高网络可靠性、控制器负载均衡和控制器的响应时间为目标，使用自适应细菌觅食优化算法解决控制器初始放置问题。Jalili等人[4]考虑了传播时延和负载均衡，并提出遗传算法来解决控制器最佳放置问题。Hu等人[5]建立二进制整数模型，以最小化网络能源消耗为目标提出遗传启发式算法。但上述方案没有考虑控制器排队时延和动态调整负载问题。对于网络的负载问题，Sallahi等人[6]提出了控制器放置模型，并且考虑了控制器间的差异性及控制器间的连接问题；Jimenez等人[7]仅考虑了控制器管理的交换机数量和控制器的流量负载；Yao等人[8]假定交换机的请求流量为一个固定值，提出单控制器和多控制放置方案，并给出了交换机迁移的动态算法；文献[9]利用控制器模块，一个控制模块内放置多个控制器，根据每个交换机的负载请求，动态调整控制模块内的控制器数量。但上述方案，仅讨论了网络的动态调整，没有考虑控制器间的传播时延和排队时延。对于控制器的排队时延，Sood等人[10]考虑传播时延和排队时延对网络性能的影响，但并没有采用实际的拓扑图进行实验。Wang等人[11]考虑端到端的传播时延以及控制器排队时延，提出聚簇网络划分算法来减少端到端的传播时延，并放置多控制器来减少排队时延，但是没有考虑负载均衡和动态调整问题。

综上所述，大部分研究主要以传播时延和负载均衡为目标来研究多控制器静态放置问题，然而多控制器放置问题是一个多目标组合优化问题，故本文在考虑传播时延和负载均衡的基础上，进一步考虑控制器间的传播时延和排队时延对CPP的影响。并针对这些目标，提出负载均衡算法(Balanced Control Region Algorithm, BCRA)和遗传算法(Genetic Algorithm, GA)用于解决多控制器静态放置问题，在线动态调整算法(Algorithm of Dynamic Online Adjustment, ADOA)用于解决网络动态负载问题。

本文的后续章节安排如下，第2节控制器放置模型；第3节介绍控制器放置及动态调整算法；第4节给出仿真实验；最后给出结束语。

2 控制器放置模型

2.1 问题描述

在SDN中，传播时延、排队时延和控制器负载影响着网络性能。因此本文主要考虑以下3个方面：

(1) 总时延。网络时延主要由传播时延，排队时延，处理时延和发送时延组成。对于广域网来说，传播时延、排队时延会从根本上影响网络的性能，因此本文定义控制器和交换机间的传播时延与排队时延之和为总时延。

(2) 控制器间传播时延。在部署多控制器的网络中，控制器间需要通信来获得全局网络的视图，因此减少控制器间的传播时延，有利于提高整个网络的一致性。

(3) 负载均衡。在SDN中，控制器的负载越大，则处理时间越久，从而造成控制器处理效率低下。因而，均衡各个控制器的负载，会提高整个网络的服务质量。

因此，本文考虑静态部署控制器时，在保证总时延和控制器传播时延较低的同时，均衡各个控制器的负载。动态调整负载时，保证较低的排队时延和运行时间。

2.2 系统模型

用无向图$G(V,E)$表示SDN, V表示网络中节点集合，E表示网络中边的集合，n表示网络中节点数量，D表示节点的度。假设在SDN中放置k个控制器，则控制器集合表示为$C=\{i\,|\,i=1,2,\cdots,k\}$。每个控制器$i$管理的交换机集合表示为$\mathrm{CA}_i$。那么将网络划分为$k$个子网需要满足以下两个条件：

$$V=\bigcup_{i=1}^{k}\mathrm{CA}_i,\ i=1,2,\cdots,k \tag{1}$$

$$\mathrm{CA}_i\cap\mathrm{CA}_h=\phi,\ i\neq h,\ i,h=1,2,\cdots,k \tag{2}$$

其中，式(1)表示所有子网的并集要覆盖整个网络，式(2)表示子网间没有重叠的节点。

为进一步描述多控制器放置问题，本文做出以下定义：

定义1 传播时延。传播时延主要分为两个部分，(1)交换机到控制器间的传播时延(TCS)；(2)控制器之间传播时延(TCC)。openflow交换机的数据流在节点间传播时延$\mathrm{TD}_{i,j}$之和与传输距离$d_{i,j}$成正比，vc表示数据在链路中的传输速度。即

$$\mathrm{TCS}_i=\frac{1}{|\mathrm{CA}_i|}\cdot\sum_{j\in\mathrm{CA}_i}\frac{d_{i,j}}{\mathrm{vc}},\ i\in C \tag{3}$$

$$\mathrm{TCC}_i=\frac{1}{|C|}\cdot\sum_{h\in C}\frac{d_{i,h}}{\mathrm{vc}},\ i\in C \tag{4}$$

式(3)表示控制器i所在子网的平均传播时延，式(4)表示控制器i到其他控制器的平均传播时延。

定义2 排队时延Tq。控制器排队时延[11–13]是指流量进入控制器后在输入队列中排队等待处理的

时间。由于每个子网仅由1个控制器管理，因此用排队论的M/M/1模型对子网进行建模。假设每个交换机的请求相互独立且服从泊松分布[13]，控制器i的处理速率为μ_i，第j个交换机的流请求服从参数为$\lambda_{i,j}$的泊松分布，则排队时延如式(5)所示

$$\mathrm{Tq}_i=\frac{\sum\limits_{j\in \mathrm{CA}_i}\lambda_{i,j}}{\mu_i\cdot\left(\mu_i-\sum\limits_{j\in \mathrm{CA}_i}\lambda_{i,j}\right)},i\in C \tag{5}$$

定义3　总时延Total。网络划分完毕后，在不拥塞的情况下，网络中有新的数据流进入openflow交换机时，交换机向控制器发出Packet – in请求，控制器处理该请求并做出相应的决策，并向交换机下发流表。这个过程的总时延可由式(6)和式(7)示。

$$T_i=Tq_i+TCS_i,i\in C \tag{6}$$

$$\mathrm{Total}=\frac{1}{|C|}\cdot\sum_{i\in C}T_i \tag{7}$$

式(6)中T_i表示控制器i所在子网的总时延，式(7)表示整个网络的平均总时延。

定义4　负载均衡参数BL。控制器负载LCP由其管辖的交换机的流请求之和决定[8]。假设子网CA_i中交换机j的流请求为$\lambda_{i,j}$，则控制器i的负载为$\mathrm{LCP}_i=\sum_{j\in \mathrm{CA}_i}\lambda_{i,j}$。为了衡量控制器间的负载差异。定义负载均衡参数为网络中最大负载与最小负载的比值，由式(8)所示

$$\mathrm{BL}=\frac{\max \mathrm{LCP}_i}{\min \mathrm{LCP}_h},i,h\in C \tag{8}$$

基于以上以总时延、控制器间传播时延和负载均衡参数为目标的多控制器放置问题，可以找到一个合适的部署方案，使得以上指标最小化，为此本文提出控制器初始放置模型为

$$\min F=\alpha\cdot\mathrm{Total}+\beta\cdot\mathrm{TCC}+\gamma\times(\mathrm{BL}-1) \tag{9}$$

s.t.

$$\alpha+\beta+\gamma=1,0\le\alpha,\beta,\gamma\le 1 \tag{10}$$

$$\sum_{j\in \mathrm{CA}_i}\lambda_{i,j}<\mu_i,i\in C \tag{11}$$

$$\mathrm{TCC}=\frac{1}{|C|}\cdot\sum_{i\in C}\mathrm{TCC}_i,\text{式(3)}-\text{式(8)} \tag{12}$$

其中，式(9)是优化目标，且α,β,γ是用于调节目标函数的影响权重。式(10)表示α,β,γ的取值范围。式(11)表示控制器不应过载。式(12)表示所有控制器间的平均传播时延。

网络流量是不断变化的，由式(5)可知，当控制器负载越接近其处理速率时，排队时延越大，为了保证网络性能，需要调整控制器的负载。在动态调整过程中，需要满足式(11)和式(13)

$$\mathrm{Tq}_i<\mathrm{Tqmax},i\in C \tag{13}$$

式(13)表示控制器的排队时延不应超过一定的阈值Tqmax。

3　控制器放置及在线动态调整算法

为了解决上述问题，本文根据建立的模型分别提出初始控制器静态放置算法其中包括BCRA和GA，以及动态调整控制器负载的ADOA。静态放置算法的目的是在保证总时延和控制器间传播时间较低的情况下，均衡各个控制器的负载。在线动态调整算法目的是通过调整控制器的负载，保证较低的排队时延。

3.1　负载均衡算法BCRA

BCRA分为两个阶段，如表1所示：(1)网络划分阶段步骤((1)—步骤(10))；(2)子网调整阶段(步骤(11)—步骤(20))。在网络划分阶段，首先选择度最大的节点作为首个控制器节点(步骤(1)—步骤(4))，并以该节点为中心，利用广度优先搜索方法，在满足式(11)的前提下，以距离最近原则构建子网(步骤(6)—步骤(8))。若网络中还有未被分配的节点，则从剩余节点中找到度最大的节点并组成集合，再从该集合中选择距离当前控制器节点最近的点作为新的控制器节点(步骤(4))，重复步骤(5)—步骤(10)，直到所有节点都被分配。至此，网络已经被划分为k个负载不均的子网。接着进入子网调整阶段，根据F不断调节交换机的归属区域直到负载均衡参数接近于1(步骤(11)—步骤(19))，最终产生k个负载均衡的子网。

在初始控制器放置情况下，度越大的节点网络连通性越好，适合部署控制器，接着选择与该节点距离最小的交换机加入子网，这可以保证每个子网的传播时延始终保持最小；当一个子网划分完毕后，从剩余度最大的节点中选择与该子网距离最近的节点部署新控制器，这可以保证控制器间的传播时延尽可能小；由于之前的步骤使得时延尽可能小而忽略了负载，造成F值偏大，因此在最后重新调节交换机时，以F值是否降低为标准来调节交换机，使得网络负载达到均衡，在此过程中时延可能会稍许增大，但是网络负载会更加均衡，F值也更小，多种目标达到互相折中，使得整体最优。

3.2　遗传算法GA

由于BCRA容易陷入局部最优且在大规模网络中解的精度较低，为了解决这个问题，本文提出一种改进的遗传算法(GA)。其关键部分如下：

表 1 BCRA算法

算法1 BCRA算法

输入：图$G(V,E)$, $\lambda_{i,j}$, Cons, $\mathrm{CA}_{\mathrm{cons}}$, C //$\mathrm{CA}_{\mathrm{cons}}$, C 都是空集

输出：F的值，控制器集合C和子网集合$\mathrm{CA}_{\mathrm{cons}}$

(1) 计算各个节点的度D; //步骤(1)～步骤(10)是网络划分阶段；

(2) 计算$k=\sum_{j=1}^{n}\lambda_j/\mathrm{LCPmax}$; //$\lambda_j$表示第$j$个交换机流量，LCPmax表示最大负载；

(3) **while** $|C|<k$

(4) 如果算法第1次执行，则选择度最大的节点Cons作为控制器节点；否则，从剩余节点中找到度最大的节点并组成集合，再从该集合中选择距离当前控制器节点最近的点Cons作为新的控制器节点；

(5) 将节点Cons加入控制器集合C；

(6) **while** $\mathrm{LCP}_{\mathrm{cons}}<\mu_{\mathrm{cons}}$//基于Con使用广度优先搜索方法构建树；

(7) 选择离Con最近且未被分配的节点j，将其加入$\mathrm{CA}_{\mathrm{cons}}$；

(8) **end while**

(9) **end while**

(10) 找到每个控制器对应的交换机集合$\{\mathrm{CA}_{\mathrm{cons}}\}$, $\mathrm{cons}\in C$,计算F 的值并记为Fold；

(11) **while** $\mathrm{BL}-1>\xi$//ξ是一个极小值，步骤(11)—步骤(19)是子网调整阶段；

(12) 找到负载最大的控制器Cons的子网，并计算子网内的边界交换机与相邻控制器之间的距离；

(13) 预先计算将距离最小的交换机分配给相邻控制器后的F的值，并标记为Fnew；

(14) **if** Fnew $\le$ Fold

(15) 将距离最小的交换机分配给相邻控制器，并令Fold = Fnew；

(16) 更新交换机集合$\{\mathrm{CA}_{\mathrm{cons}}\}$并计算$\sum_{j\in\mathrm{CA}_{\mathrm{cons}}}\lambda_{\mathrm{cons},j}$；

(17) **end if**

(18) $F=$ Fold；

(19) **end while**

(20) 输出F, C, $\{\mathrm{CA}_{\mathrm{cons}}\}$

染色体编码与种群初始化：采用二进制编码方式，对控制器待选集合进行编码。染色体的长度由待选集合的元素个数决定，1条染色体代表1种控制器部署方案。在初始化种群时，先将拓扑中的节点按照度降序排列，并选择大于网络平均度的节点加入控制器待选集合。

适应度函数：为了找寻最佳控制器部署位置，以式(9)作为适应度函数，对于每代染色体，采用BLA(Balancing Load Algorithm)算法找到可行的部署方案。表2为BLA算法描述。BLA首先根据输入的控制器集合，在保证控制器不过载的情况下，将交换机按最近距离分配给控制器(步骤(1)—步骤(7))，形成负载不均的多个子网。接着调用BCRA的子网调整阶段算法根据F调节子网间负载(步骤(8))，最后输出适应度函数值。

选择、交叉和变异：在选择过程中，采用锦标赛选择策略。在交叉过程中，采用单点交叉。在变异过程中，采用基本位变异。

3.3 在线动态调整算法ADOA

网络在实际的运行过程中，流量是不断变化的，为了应对控制器过载问题，本文提出在线动态调整算法(ADOA)，如表3所示。ADOA的原理是，基于初始划定的控制域的基础上，将初始控制器设为主控制器，通过在各控制域放置从控制器，来降低主控制器的负载以及排队时延。在划定初始控制区域后，每当控制器负载发生变化，则根据式(11)和式(13)判断自身是否过载，排队时延是否过大(步骤(1))，如果是，则选取当前子网中度最大

表 2 BLA算法

算法2 BLA算法

输入：图$G(V,E)$, $\lambda_{i,j}$, μ_i, CA, C //CA是空集

输出：适应度函数值F

(1) **for** $i\in C$

(2) 选择节点j，其满足离控制器i最近且未被标记；

(3) **if** $\sum_{j\in\mathrm{CA}_i}\lambda_{i,j}<\mu_i$

(4) 将节点j加入CA_i；

(5) **end if**

(6) **end for**

(7) 找到每个控制器对应的交换机集合$\{\mathrm{CA}_i\}$, $i\in C$，计算F的值并记为Fold；

(8) 接着调用BCRA的子网调整阶段算法；//即算法1的步骤(11)—步骤(19)；

(9) 输出F

表 3　ADOA算法

算法3 ADOA算法

输入：CA_i, $\lambda_{i,j}$, μ_i, C, LCP_i, D

输出：$\mathrm{CA}_{i,o}$

(1) **while** LCP_i改变并且$\mathrm{Tq}_i > \mathrm{Tqmax}$

(2) 选择当前子网中度最大的节点o作为从控制器；

(3) **for** $j \in \mathrm{CA}_i$

(4) 如果$\mathrm{LCP}_o > \mathrm{LCP}_i$，则跳出当前循环并转入步骤2；

(5) **if** 相对于控制器i，节点j离控制器o近；

(6) 预先计算将j分配给o后的Tq_o；

(7) 如果$\mathrm{Tq}_o < \mathrm{Tqmax}$，则令$j \in \mathrm{CA}_{i,o}$且$j \notin \mathrm{CA}_i$；

(8) **end if**

(9) **end for**

(10) **end while**

(11) 输出$\mathrm{CA}_{i,o}$

的节点激活从控制器(步骤(2))，判断从控制器负载是否大于主控制器，并依据距离远近和式(11)来分配交换机(步骤(3)—步骤(10))，最后主控制器记录当前子网节点的分配情况(步骤(11))。

ADOA是一个在线并行算法。在动态调整过程中，每个子网的主控制器可实时检测该子网内的控制器排队时延，并自发做出对应的调整策略。主控制器通过激活从控制器可以有效地减轻自身的负载。在整个调整过程中，始终在域内调节交换机的归属，所以控制器与交换机的传播时延只会不断减少。同时由于从控制器仅与本区域内主控制器通信[14]，这区别于主控制器之间的相互通信，因此部署从控制器对控制器之间的平均传播时延影响较小。

4　仿真实验

为了对所提算法进行评估，本文的仿真实验建立在真实的网络拓扑中，分别是网络拓扑动物园的OS3E网络和IRIS网络。OS3E网络有34个节点和42条边，IRIS网络有51个节点和64条边。考虑到网络拓扑的连通性，使用Haversine[15]公式计算节点之间的距离，并使用Dijkstras算法计算最短路径距离。同时，设置控制器的最大负载为1800 flows/s[13]，各个交换机的流请求相互独立且为100 flows/s。Tqmax设置为1 ms[11]。设置vc为3×10^8 m/s[16]，设置交叉概率为0.7，变异概率为0.05，迭代次数为300。在静态情况下，分别比较GA, BCRA, k-center[11]和k-means[17]在总时延、控制器传播时延以及控制器间负载方面的性能；在动态情况下，分别比较ADOA和动态调整算法(DCP)[18]在排队时延、负载均衡、控制器数量以及运行时间方面的性能。本文选用matlab2016进行实验仿真。

4.1　静态部署情况

实验1验证在不同的网络拓扑中，GA, BCRA, k-center与k-means在总时延上的差异。实验结果如图1(a)和图2(a)示。在图1(a)中，当控制器数量为3时，GA, BCRA与k-center在总时延的差值达到最大，分别为1.1 ms和0.9 ms，与k-means的差值分别为2 ms和1.8 ms。说明提出的算法部署在中小型网络中时，其总时延性能接近于k-center和k-means。在图2(a)中，当控制器数量为3, 4, 5时，GA, BCRA, k-means与k-center在总时延的差值较大，且在控制器数量为5时达到最大，分别为3.03 ms, 3.06 ms和3.05 ms。分析其原因可知，k-center注重减小传播时延而忽略控制器的负载情况，当在大规模网络部署少量控制器时，容易造成局部控制器负载过大，导致平均排队时延增大。

实验2验证在不同的网络拓扑中，GA, BCRA, k-center与k-means在控制器传播时延上的差异。实验结果如图1(b)和图2(b)示。在图1(b)中，GA, BCRA的传播时延均远小于k-center和k-means。分析其原因，k-center每次选择距离簇最远的节点为控制器，造成控制器间的传播时延大幅增加。k-

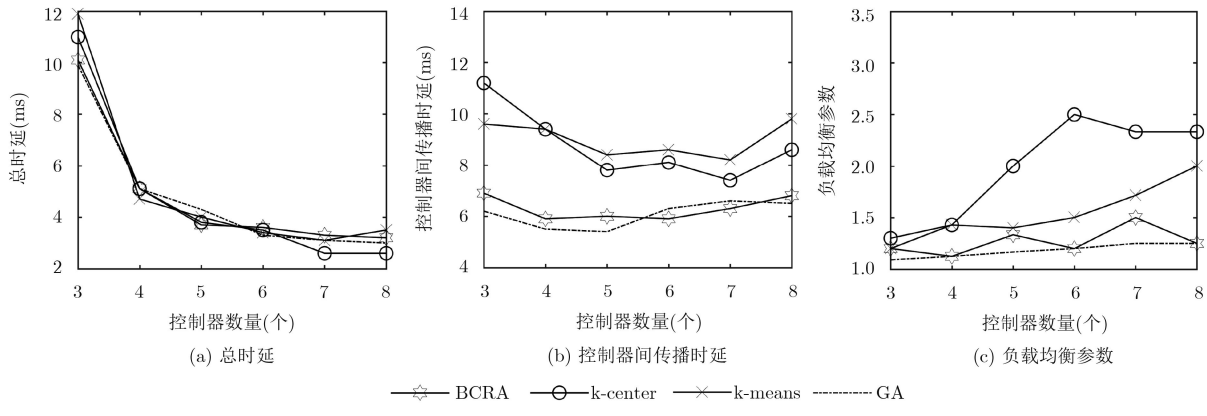

图 1 OS3E网络中的性能指标

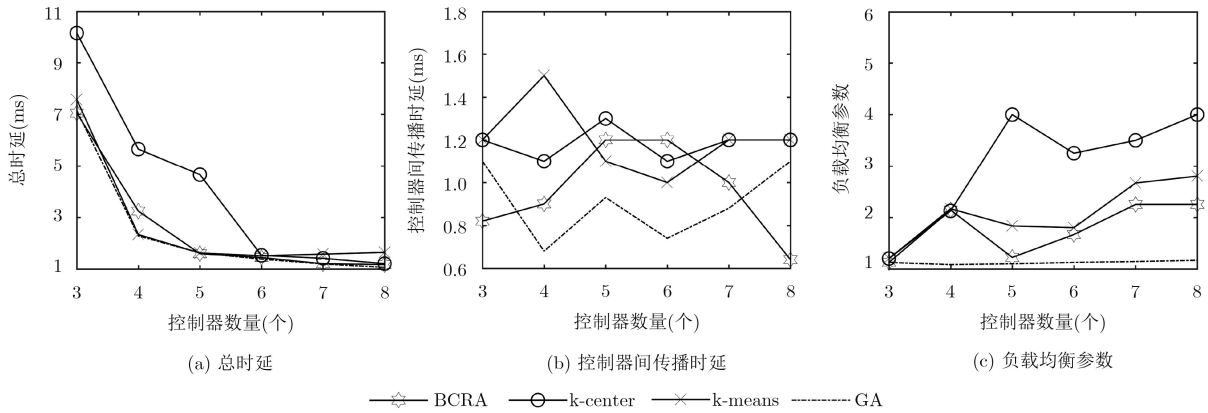

图 2 IRIS网络中的性能指标

means每次选择节点加入簇时，会以最小距离和重新改变控制器的位置。而BCRA每次选择离簇最近的度最大的节点为控制器，能够有效降低控制器间传播时延。GA, BCRA的传播时延相近，且差值始终小于0.7 ms。在图2(b)中，GA始终能保持较低的传播时延，当控制器数量为3和8时，BCRA传播时延较低。分析其原因：BCRA存在着密集区域控制器局部最优问题。

实验3验证在不同的网络拓扑中，GA, BCRA, k-center和k-means在控制器负载上的差异。实验结果如图1(c)和图2(c)示。在图1(c)中，GA的负载均衡参数始终接近于1，且最小值1.09，最大值是1.25。BCRA负载均衡参数始终接近GA，且最大值为1.5。均优于k-center和k-means。分析其原因是：k-center每次选择最大度节点作为控制器节点，随着控制器数量的增加，容易陷入局部最优。k-means以减少传播时延为目标，虽然避免陷入局部最优的情况，但是未考虑控制器的负载，导致局部控制器负载大幅增加。在图2(c)中，GA的负载均衡参数始终接近于1，且最小值1.1，最大值是1.16。与BCRA, k-center和k-means相比，负载均衡率平均提高了49.7%。而BCRA负载均衡参数最大值达到2.25且大部分情况远高于GA。分析其原因：BCRA虽然注重控制器的负载，但是由于其划分网络时，每次选择离当前簇最近的节点作为新簇中心，在大型网络中容易造成部分控制器因连通性的限制而无法调节自身的负载的情况。

实验结果证明了所提算法的优越性，且BCRA适用于小规模网络，GA适用于大规模网络。

4.2 在线动态调整情况

实验4验证在网络局部过载的情况下，在线算法ADOA在排队时延及控制器负载上的性能。本文用matlab泊松分布流量发生器随机生成24 h的请求流量各100次，分别通过ADOA和DCP调整后，计算各个时间段的平均值。实验结果如图3和图4示。在图3中，ADOA通过动态调整控制器负载，使得整个网络平均排队时延始终维持在0.871～0.873 ms之间。而经过DCP调节后，整个网络的平均排队时延在0.912～0.927 ms之间，略高于ADOA。在图4中，ADOA的负载均衡参数在1.50～1.54之间，DCP的负载均衡参数在1.39～1.45之间，且两者差值始终小于0.13。说明在动态调整过程中，ADOA和DCP在排队时延和网络负载均衡方面的性能相近。

实验5验证在动态调整过程中，ADOA的控制器资源消耗情况。本文随机选择部分交换机，逐次递增其请求流量，使得网络局部过载。实验结果如

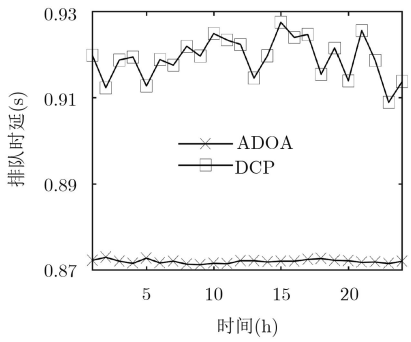

图 3 动态调整后的网络排队时延

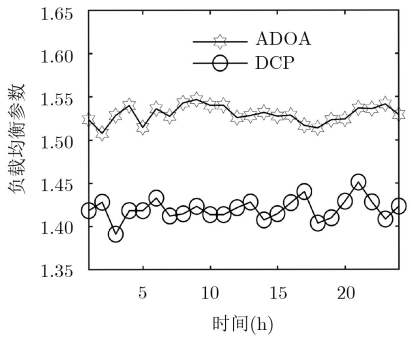

图 4 动态调整后的网络负载均衡情况

图5和图6示。在图5中，DCP每次调整后，均可以保证网络中控制器的数量较少，ADOA在部分交换机流量为150, 250, 350时，控制器数量比DCP多1个，分析其原因：网络局部过载时，存在无需增加控制器的情况，即该局部过载可通过调整域间交换机归属关系解决。DCP是通过调节域间交换机的归属关系来解决控制器的过载情况，这会改变初始的子网规模，造成控制器与部分交换机的传播时延增大。ADOA通过新增控制器调节负载，虽然造成了一定的资源浪费，但是其仅在初始划分的子网内进行交换机调整，这样可以保证控制器与交换机的传播时延不增大。在图6中，ADOA的运行时间始终在1.3 s左右，而DCP的运行时间在3.9～4.4 s之间。分析其原因：每次流量变化时，DCP需要遍历全网，去调节交换机和控制器的映射关系，因此运行时间较长。在动态调整过程中，ADOA虽然会少量增加控制器的数量，但是在时间性能上要两倍优于DCP。

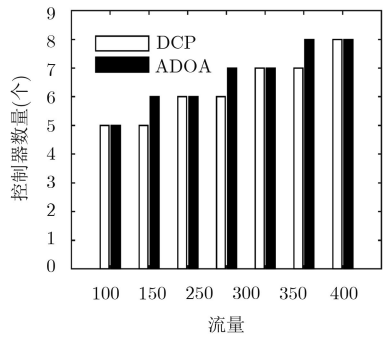

图 5 动态调整后控制器数量

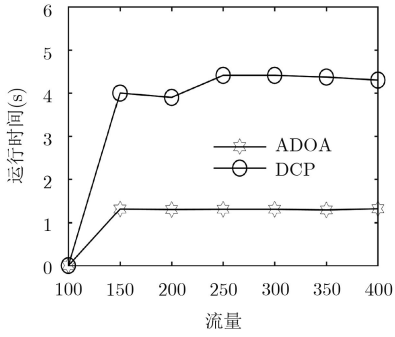

图 6 算法运行时间

5　结束语

合理的放置控制器有利于提升网络性能。本文在考虑控制器之间的传播时延，总时延，负载均衡以及连通性的基础上，提出一种多控制器部署策略。通过仿真实验可知，在静态情况下，BCRA适合部署在中小规模网络中，GA适合部署在大型网络中；在动态情况下，ADOA具有较低的排队时延和运行时间。未来会考虑控制器利用率过低时的休眠操作以及网络的可靠性。

参 考 文 献

[1] HELLER B, SHERWOOD R, and MCKEOWN N. The controller placement problem[J]. *ACM SIGCOMM Computer Communication Review*, 2012, 42(4): 473–478. doi: 10.1145/2377677.2377767.

[2] LANGE S, GEBERT S, ZINNER T, *et al*. Heuristic approaches to the controller placement problem in large scale SDN networks[J]. *IEEE Transactions on Network and Service Management*, 2015, 12(1): 4–17. doi: 10.1109/TNSM.2015.2402432.

[3] ZHANGA Bang, WANG Xingwei, and HUANG Min. Multi-objective optimization controller placement problem in internet-oriented software defined network[J]. *Computer Communications*, 2018, 123: 24–35. doi: 10.1016/j.comcom.2018.04.008.

[4] JALILI A, KESHTGARI M, and AKBARI R. Optimal controller placement in large scale software defined networks based on modified NSGA-II[J]. *Applied Intelligence*, 2018, 48(9): 2809–2823. doi: 10.1007/s10489-017-1119-5.

[5] HU Ying, LUO Tao, BEAULIEU N C, *et al*. The energy-aware controller placement problem in software defined networks[J]. *IEEE Communications Letters*, 2017, 21(4): 741–744. doi: 10.1109/LCOMM.2016.2645558.

[6] SALLAHI A and ST-HILAIRE M. Optimal model for the controller placement problem in software defined networks[J]. *IEEE Communications Letters*, 2015, 19(1): 30–33. doi: 10.1109/LCOMM.2014.2371014.

[7] JIMÉNEZ Y, CERVELLÓ-PASTOR C, and GARCÍA A. 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.

[8] YAO Guang, BI Jun, LI Yuliang, *et al*. On the capacitated controller placement problem in software defined networks[J]. *IEEE Communications Letters*, 2014, 18(8): 1339–1342. doi: 10.1109/LCOMM.2014.2332341.

[9] HUQUE M T I U, SI Weisheng, JOURJON G, *et al*. Large-scale dynamic controller placement[J]. *IEEE Transactions on Network and Service Management*, 2017, 14(1): 63–76. doi: 10.1109/TNSM.2017.2651107.

[10] SOOD K and XIANG Yong. The controller placement problem or the controller selection problem?[J]. *Journal of Communications and Information Networks*, 2017, 2(3): 1–9. doi: 10.1007/s41650-017-0030-x.

[11] WANG Guodong, ZHAO Yanxiao, HUANG Jun, *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. doi: 10.1109/TNSM.2017.2785660.

[12] 高先明, 王宝生, 邓文平, 等. SDN网络中控制器放置问题综述[J]. 通信学报, 2017, 38(7): 155–164. doi: 10.11959/j.issn.1000-436x.2017136.
GAO Xianming, WANG Baosheng, DENG Wenping, *et al*. Survey of controller placement problem in software defined network[J]. *Journal on Communications*, 2017, 38(7): 155–164. doi: 10.11959/j.issn.1000-436x.2017136.

[13] WANG Tao, LIU Fangming, and XU Hong. An efficient online algorithm for dynamic SDN controller assignment in data center networks[J]. *IEEE/ACM Transactions on Networking*, 2017, 25(5): 2788–2801. doi: 10.1109/TNET.2017.2711641.

[14] HU Tao, GUO Zehua, YI Peng, *et al*. Multi-controller based software-defined networking: A survey[J]. *IEEE Access*, 2018, 6: 15980–15996. doi: 10.1109/ACCESS.2018.2814738.

[15] WANG Guodong, ZHAO Yanxiao, HUANG Jun, *et al*. On the data aggregation point placement in smart meter networks[C]. The 26th International Conference on Computer Communication and Networks, Vancouver, Canada, 2017: 1–6. doi: 10.1109/ICCCN.2017.8038499.

[16] 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.

[17] WANG Guodong, ZHAO Yanxiao, HUANG Jun, *et al*. A K-means-based network partition algorithm for controller placement in software defined network[C]. 2016 IEEE International Conference on Communications, Kuala Lumpur, Malaysia, 2016: 1–6. doi: 10.1109/ICC.2016.7511441.

[18] BARI M F, ROY A R, CHOWDHURY S R, *et al*. Dynamic controller provisioning in software defined networks[C]. The 9th International Conference on Network and Service Management, Zurich, Switzerland, 2013: 18-25. doi: 10.1109/CNSM.2013.6727805.

史久根：男，1963年生，副教授，研究方向为嵌入式系统、计算机网络和无线传感器网络.

谢熠君：男，1995年生，硕士生，研究方向为软件定义网络、控制器放置和嵌入式系统.

孙　立：男，1993年生，硕士生，研究方向为软件定义网络、路由多播和嵌入式系统.

郭　胜：男，1993年生，硕士生，研究方向为软件定义网络、网络虚拟化和规则放置.

刘雅丽：女，1996年生，硕士生，研究方向为软件定义网络、网络虚拟化和规则缓存.