软件定义网络中基于效率区间的负载均衡在线优化算法

史久根 徐 皓* 张 径 王 继
(合肥工业大学计算机与信息学院 合肥 230009)

摘 要：在大型复杂软件定义网络中，为提高网络负载均衡，减少控制器与交换机间的传播时延，该文提出一种基于效率区间的负载均衡在线优化算法。在初始静态网络中，通过贪心算法选择初始控制器集合，并以其为根节点构建M棵改进代价的最小生成树(MST)，确定初始M个负载均衡的子网；当网络流量发生变化时，通过广度优先搜索(BFS)调整子网间交换机映射关系使其满足效率区间，保证任意时刻网络的负载均衡。算法均以网络连通性为基础，且均以传播时延为目标重新更新控制器集合。仿真实验表明，该算法在保证任意时刻网络负载均衡的同时，可以保证较低的传播时延，与Pareto模拟退火算法、改进的K-Means算法等相比，可以使网络负载均衡情况平均提高40.65%。

关键词：软件定义网络；控制器部署；负载均衡；传播时延
中图分类号：TP393 **文献标识码**：A **文章编号**：1009-5896(2019)03-0694-08
DOI: 10.11999/JEIT180464

An Efficient Online Algorithm for Load Balancing in Software Defined Networks Based on Efficiency Range

SHI Jiugen XU Hao ZHANG Jing WANG Ji
(*School of Computer and Information*, *Hefei University of Technology*, *Hefei* 230009, *China*)

Abstract: Due to the limitation of individual controller's processing capacity in large-scale complex Software Defined Networks (SDN), an efficient online algorithm for load balancing among controllers based on efficiency range is proposed to improve load balancing among controllers and reduce the propagation delay between a controller and the switch. In the initial static network, the initial set of controllers is selected by a greedy algorithm, then M improved Minimum Spanning Trees (MST) rooted at the initial set of controllers are constructed, so initial M subnets with load balancing are determined. With the dynamic changes of load, for the purpose of making the controller work within efficiency range at any time, several switches in different subnets are reassigned by Breadth First Search (BFS). The initial set of controllers is updated for minimizing propagation delay in the algorithms' last step. The algorithm is based on the connectivity of intra-domain and inter-domain. Simulation results show that the proposed algorithms not only guarantee the load balancing among controllers, but also guarantee the lower propagation delay. As to compare to PSA algorithm, optimized K-Means algorithm, etc., it can make Network Load Balancing Index (NLBI) averagely increase by 40.65%.

Key words: Software Defined Networks (SDN); Controller placement; Load balancing; Propagation delay

1 引言

软件定义网络(SDN)作为一种新型的网络架构，通过对控制平面和转发平面间的解耦，将控制逻辑分配给控制器，每一个控制器管理着多个交换机，可对交换机进行可编程化管理。受限于单个控制器的处理能力，控制器过载会造成网络数据流传播时延过大等问题，因此需要使用多个控制器管理网络分区，如何确定网络所需控制器数量以及如何在网络中部署多个控制器成为Heller等人[1]提出的控制器部署问题的核心。

针对该问题，国内外学者提出了一系列的解决方案，Koponen等人[2]提出Onix方案，通过分布式哈希表来管理不同控制器的状态分布问题；Tootoonchian等人[3]提出的HyperFlow方案，基于WheelFS分布式文件系统处理多控制器的状态分布问题；Yu等人[4]提出DIFANE方案，通过设置权威

收稿日期：2018-05-15；改回日期：2018-09-20；网络出版：2018-10-22
*通信作者： 徐皓 2016170681@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)

交换机和普通交换机，对转发模块进行分层处理；以上方案设计了多控制器的部署架构，未考虑针对不同目标部署控制器，因此具有一定的局限性。

Heller等人[1]最早指出控制器部署问题是一个NP难问题，其通过BF算法对所有潜在节点进行评估以选择最优部署节点，优化了网络平均时延和最大时延；Wang等人[5]提出了优化的K-Means算法，在克服标准K-Means聚类局部性缺陷的基础上，逐次增加控制器数量；张栋等人[6]通过多层K路划分方法划分网络区域，降低了网络通讯代价；文献[7]采用贪心算法划分网络区域，但仅以时延为优化目标，上述几种方案考虑初始静态情况下控制器部署策略，且均以传播时延为优化目标，未考虑负载均衡问题。Yao等人[8]首次引入负载均衡目标，其实验表明算法可以有效减少网络所需控制器数量及控制器最大负载，但未表明网络时延情况；Sallahi等人[9]提出了可以同时确定最优的控制器数量、控制器类型及部署位置的数学模型，但文献中未给出算法过程，且仅适用于小规模网络；Jimenez等人[10]提出了K-Critical算法，通过构建鲁棒树来选择所需最少的控制器数量及部署位置，算法仅考虑控制器负载；Lange等人[11]提出了基于Pareto模拟退火的启发式算法，基于失效率，时延，负载等不同指标设计了相应的帕拉托最优算法；文献[12]考虑将多目标遗传算法用于控制器部署问题，但以上几种方案均未考虑网络的连通性。

综上，本文针对控制器部署问题中的初始静态和动态情况，提出了一种基于效率区间的负载均衡在线优化算法，在初始静态情况下，通过贪心算法选择初始控制器集合，并以其为根节点构建多棵改进代价的最小生成树，生成初始负载均衡子网；当网络流量发生变化时，通过广度优先搜索调整域间交换机映射关系，使各控制域负载满足效率区间，避免下一时隙流量激增造成控制器过载问题，或流量骤减造成控制器休眠问题，保证任意时刻网络的负载均衡；以上两种情况均以降低传播时延为目标重新更新各域内控制器位置。

本文的后续章节安排如下，第2节描述控制器部署模型；第3节描述控制器部署算法及调整算法；第4节描述仿真实验，分析实验结果；最后对本文提出总结与展望。

2　控制器部署模型

2.1　系统模型

SDN网络拓扑可用无向图$G=(V,E)$表示，n表示节点个数，V表示节点集合，E表示节点间边的集合，D表示节点度的集合，划分网络后，C表示控制器集合，子网i可用无向图$\mathrm{SN}_i=(\mathrm{SV}_i,\mathrm{SE}_i)$表示。网络初始所需控制器个数$m$可由单位时间内流经$n$个节点的流量请求之和与单个控制器可以处理的平均流量请求L的比值确定[13]。

$$D=\{|D_1|,|D_2|,\cdots,|D_i|\},\ C=\{C_1,C_2,\cdots,C_j\},\\ i=1,2,\cdots,n,j=1,2,\cdots,m \tag{1}$$

$$\bigcup_{i=1}^{m}\mathrm{SV}_i=V,\ \mathrm{SV}_i\cap\mathrm{SV}_j=\varnothing,\ \bigcup_{i=1}^{m}\mathrm{SE}_i=E,\\ \mathrm{SE}_i\cap\mathrm{SE}_j=\varnothing,\forall i\neq j,i,j=1,2,\cdots,m \tag{2}$$

$$X_{i,j}=\{1|\mathrm{SV}_i\text{由控制器}C_j\text{管理},\mathrm{SV}_i\in V,C_j\in C\},\\ \sum_{j=1}^{m}X_{i,j}=1,i=1,2,\cdots,n \tag{3}$$

式(2)表示子网间没有重叠的节点与边，式(3)表示交换机与控制器间一对一映射。

定义1　请求时延(Request Delays of Transmission, RDT)：表示交换机j向控制器i发出数据流请求并接收到响应的总时延，数据流在节点间传播时延$T_{i,j}$与传输距离$d_{i,j}$成正比[14]，T_q表示数据流经过其他节点的平均排队时延，T_p表示控制器对数据流进行处理的平均处理时延，因此请求时延可表示为

$$\mathrm{RDT}_{i,j}=2\times\sum_{u\in i,v\in j}(T_{u,v}+T_q)+T_p,\\ i=1,2,\cdots,n,j=1,2,\cdots,m \tag{4}$$

定义2　域内总请求时延(Total Request Delays of Transmission, TRDT)：划分网络后，可用域内交换机与控制器之间请求时延$\mathrm{RDT}_{i,j}$之和表示子网i的域内总请求时延TRDT_i, TRDT_i可表示为

$$\mathrm{TRDT}_i=\sum_{j\in\mathrm{SV}_i}\mathrm{RDT}_{i,j},i=1,2,\cdots,m \tag{5}$$

定义3　最小生成树代价(Weight of the minimum spanning tree, 简称为W)：节点的度可以反映节点的重要程度以及经过该节点的流的数量，节点之间的距离大小可以反映其进行信息交换的难易程度，采用节点度与节点间距离的组合重新定义其边的代价$W_{i,j}$，本文采用Prim算法生成改进代价的最小生成树。

$$W_{i,j}=\frac{2\times\max(D)}{D_i+D_j}\times d_{i,j},i,j=1,2,\cdots,n \tag{6}$$

其中，D_i与D_j分别为节点i与节点j的节点度。

定义4　局部密集度(Local Density, LD)：用于表示节点到其邻居节点的紧凑程度，节点i的直接相邻节点集合为NS_i，节点i的局部密集度LD_i与

节点i和其直接相邻节点之间的请求时延之和成反比，数据流无需经过其他交换机转发，则有

$$\mathrm{LD}_i=\frac{1}{2\times\sum\limits_{j\in\mathrm{NS}_i}T_{i,j}+|\mathrm{NS}_i|\times T_p},i=1,2,\cdots,n \tag{7}$$

定义5 网络负载均衡指标(Network Load Balancing Index, NLBI)：划分网络后，子网的控制器负载(Load Capacity of Controller, LCC)由其管理区域内交换机请求数量之和决定，可用域间最大负载和最小负载的比值定义时隙t内的网络负载均衡指标NLBI^t，实际工作中，若网络负载均衡指标NLBI接近于1，则该网络接近于负载均衡分布。

$$\mathrm{NLBI}^t=\frac{\max(\mathrm{LCC}_i^t)}{\min(\mathrm{LCC}_j^t)},i,j=1,2,\cdots,n \tag{8}$$

对于同一类型控制器，其最大负载能力(Maximum Load Capacity of Controller, MLCC)有限且相同，实际工作中，控制器不仅需工作在最大负载能力范围内，且需工作在效率区间$[\alpha\times\mathrm{MLCC},\beta\times\mathrm{MLCC}]$内，原因在于控制器负载低于效率区间下限会造成其利用率过低，增加了部署成本，控制器负载高于效率区间上限会造成其可接受数据流量过小，网络流量激增时极易过载[15]，其中，α与β均表示衰减因子[15]。

基于以上网络负载均衡指标和域内总请求时延，受负载、时延约束的多控制器部署问题可以找到一个合适部署方案，使得以上指标最小化，为此，可以建立带约束的线性规划模型：

$$\begin{aligned}\mathrm{Min}\ F_t(V,E)&=\mu\times(\mathrm{NLBI}^t-1)\\&\quad+\nu\times\sum_{i=1}^{m}\mathrm{TRDT}_i^t\end{aligned} \tag{9}$$

$$\mu+\nu=1,0\le\mu,\nu\le1 \tag{10}$$

$$\text{s.t. }\max(\mathrm{Lh}_{i,j}^t)\ge1,i,j=1,2,\cdots,m \tag{11}$$

$$\max(T_{i,j})\le T_{\max},i,j=1,2,\cdots,n \tag{12}$$

$$\begin{aligned}&\alpha\times\mathrm{MLCC}\le\mathrm{LCC}_i^t\le\beta\times\mathrm{MLCC},\\&0<\alpha,\beta<1,i=1,2,\cdots,m\end{aligned} \tag{13}$$

在任意时隙t内，$\mathrm{Lh}_{i,j}^t$表示控制器i与控制器j之间的链路跳数，则式(11)表示任意子网域间保持连通性；式(12)表示网络任意节点间传播时延小于网络最大允许传播时延($T_{\max}$)；式(13)表示控制器应工作在效率区间内。

3 控制器部署算法

综上所述，本文提出初始M控制域选择(Initial M Control Domains Selection, IMCDS)算法和M控制域调整(M Control Domains Adjustment, M-CDA)算法。

3.1 初始静态情况

根据式(1)、式(4)、式(6)和式(7)，分别构造节点度数组D，节点间请求时延矩阵$\boldsymbol{R}$，代价矩阵$\boldsymbol{W}$及局部密集度数组ID。已知控制器最大负载能力MLCC及单位时间内流经任意节点的流量，初始优化负载(OL)为$0.85\times\mathrm{MLCC}$[15]。初始M控制域选择算法步骤如表1所示。根据文献[13]确定初始所需控制器数目M(步骤1)，首先，采用贪心思想确定首个控制器：(1)选择节点度最大的节点作为首个控制器C_1；(2)若存在多个度相同的节点，则选择局部密集度最高的节点作为首个控制器C_1，将C_1加入初始控制器集合ICon(步骤2)，在子网满足式(11)、式(12)的条件下，采用Prim算法以控制器C_1为根节点，根据代价矩阵$\boldsymbol{W}$，构建负载最大为优化负载(OL)的改进代价的最小生成树(步骤3)，更新初始交换机集合IS_1并对已选节点进行标记(步骤4)；其次，采用式(14)从未被标记的节点中选择收益P_i最大的节点作为新的控制器C_i，式(14)中

表 1 初始M控制域选择算法

算法1：初始M控制域选择算法(IMCDS)

输入：$G=(V,E),f,\mathrm{MLCC},T_{\max},\boldsymbol{W},D,\mathrm{ID},\{\mathrm{Flag},\mathrm{Temp},\mathrm{ICon},\mathrm{IS}\}=\varnothing$

输出：$\mathrm{ICon},\mathrm{IS}$

(1) $M\leftarrow\sum_{i=1}^{|V|}f_i\big/\mathrm{OL},\mathrm{OL}=0.85\times\mathrm{MLCC}$;

(2) $\mathrm{ICon}\leftarrow C_1,C_1\in\{\max(D)\cap\max(\mathrm{ID})\}$;

(3) Adopt $\mathrm{Prim}(C_1,\boldsymbol{W},\mathrm{OL})$ to construct first improved MST based 式(11) and 式(12);

(4) Update $\mathrm{IS}_1,\mathrm{Flag}_1\leftarrow1,\mathrm{Temp}\leftarrow1$;

(5) **do**

(6) $\mathrm{ICon}\leftarrow C_i,C_i\in\{\mathrm{Flag}_i=0\cap\text{furthest from ICon}\cap\max(D)\cap\max(\mathrm{ID}),i=1,\cdots,n\}$;

(7) Adopt $\mathrm{Prim}(C_i,\boldsymbol{W},\mathrm{OL})$ to construct improved MST based on 式(11) and 式(12);

(8) Update $\mathrm{IS}_i,\mathrm{Flag}_i\leftarrow1,\mathrm{Temp}\leftarrow\mathrm{Temp}+1$;

(9) **while** $\mathrm{Temp}>M$, the network has been divided into M domains;

(10) **if** there are nodes has not been selected **do**

(11) Add nodes to one closest improved MST by BFS based on 式(11) and 式(12);

(12) Update $\mathrm{IS},\mathrm{Flag}\leftarrow1$;

(13) **end if**

(14) Update centroids $\mathrm{ICon}=\{C_1,C_2,\cdots,C_M\}$ based on the sum of the shortest distance from all switches in IS_i to new controller C_i is minimized;

(15) Output $\mathrm{ICon},\mathrm{IS}$。

$\sum_{j\in \mathrm{ICon}} d_{i,j}$表示待选节点$i$与初始控制器集合中已选控制器的传输距离之和，$D_i$表示待选节点$i$的度，$\mathrm{LD}_i$表示待选节点$i$的局部密集度；重复上述最小生成树的构建过程及新的控制器选取过程，直到生成M棵改进代价的最小生成树(步骤5—步骤9)；若有节点未被选中，通过广度优先遍历将其加入距离其最近的最小生成树中，直到所有节点均被标记(步骤10—步骤13)；最后在各域内以最小化交换机与控制器间传播时延之和为目标更新控制器位置(步骤14)，生成初始状态下的子网分布。

$$P_i=\frac{\sum\limits_{j\in \mathrm{ICon}} d_{i,j}}{|\mathrm{ICon}|}\times\frac{D_i}{\max(D)}\times\frac{\mathrm{LD}_i}{\max(\mathrm{LD})},\quad i=1,2,\cdots,n \tag{14}$$

IMCDS算法采用基于节点度和局部密集度的改进贪心算法确定首个控制器位置，与随机选择控制器算法和基于最大度选择控制器算法相比，减少了传播时延；采用基于传输距离、节点度、局部密集度3个指标的改进贪心算法确定剩余控制器位置，避免了控制器间过近而导致聚类陷入局部最优。采用Prim算法生成M棵负载最大为优化负载(OL)的改进代价的最小生成树，同时标记已选节点；对于未标记的节点依次加入距离其最近的子网，则网络完全划分为M个子网。采用改进代价的最小生成树及广度优先遍历搜索未标记节点均可降低子网内平均传播时延，更新各子网内控制器位置进一步减少了请求时延，因此M个子网的总请求时延较小。式(11)可以保证控制器间链路跳数至少为1，即保证子网域间连通，采用最小生成树生成子网分布可以保证各子网域内连通，则M个子网始终保持连通性。对于任意子网，其负载均无限趋近于优化负载(OL)，因此IMCDS算法可以生成网络负载均衡指标趋近于1的子网分布。

定理1　IMCDS算法的时间复杂度为$O(n^2)$。

证明　计算局部密集度数组ID、收益P_i、节点间请求时延矩阵$\boldsymbol{R}$和代价矩阵$\boldsymbol{W}$的时间均为$O(n^2)$，构建首个子网的最小生成树的平均时间为$O(|V|^2/M^2)$，标记节点的时间为$O(n)$；算法重复$M-1$次即可划分剩余区域，该阶段的时间为$O((M-1)\times(|V|^2/M^2+n))$；更新各控制域内控制器位置阶段需要的时间为$O(n\times n/M)$。综上，算法的时间复杂度为

$$O(n^2)+O(|V|^2/M^2)+O(n)+O((M-1)\times(|V|^2/M^2+n))+O(n\times n/M)\approx O(n^2) \tag{15}$$

证毕

3.2　流量动态变化情况

当网络流量发生变化时，为使各控制域负载在满足效率区间的同时，减少域内总请求时延，需要重新调整域间交换机映射关系以优化负载和时延指标：控制域负载未达效率区间则增加负载，满足效率区间则保持不变，超过效率区间则减少负载。M控制域调整算法步骤如表2所示。首先，建立中间交换机集合TNS用于标记任意控制域内距离已选控制器ICon_i最远，流量最大的边界交换机$\mathrm{BV}_{i,j}$(步骤2—步骤4)；基于控制器负载过低会造成其利用率过低，增加部署成本，控制器负载过高会造成其可接收数据流量过小，极易过载这一特性，在任意时隙t内，若控制域负载未达效率区间下限$\alpha\times$MLCC，利用广度优先遍历从TNS中选择距离已选控制器ICon_i最近、流量适中的交换机加入该控

表 2　M控制域调整算法

算法2：M控制域调整算法(M-CDA)

输入：$G=(V,E)$, $T_{\max}$, f, MLCC, (α,β), ICon, IS, T, $\{\mathrm{LS},\mathrm{LCon},\mathrm{TNS}\}=\varnothing$

输出：LCon, LS

(1) **for** all t with $0\le t\le T$ **do**

(2)　**for each** $i\in M$ **do**

(3)　　$\mathrm{TNS}\leftarrow \mathrm{BV}_{i,j},\mathrm{BV}_{i,j}\in\{\text{furthest from } \mathrm{ICon}_i\cap\max\left(f_{i1}^t,\cdots,f_{ij}^t\right),j\in\mathrm{IS}_i\}$;

(4)　**end for**

(5)　**for each** $i\in M$ **do**

(6)　　**if** $\sum_{l=t-1}^{t}\mathrm{LC}_i^l<\alpha\times\mathrm{MLCC}$ **do**

(7)　　Add nodes in TNS closest from ICon_i to IS_i by BFS based 式(11)，式(12)，式(13);

(8)　　**end if**

(9)　　**if** $\sum_{l=t-1}^{t}\mathrm{LC}_i^l>\beta\times\mathrm{MLCC}$ **do**

(10)　Add nodes in IS_i furthest from ICon_i to TNS by BFS based 式(11)，式(12)，式(13);

(11)　**end if**

(12)　**end for**

(13)　Compute function

$$F_t=\mu\times(\mathrm{NLBI}^t-1)+\nu\times\sum_{j=1}^{M}\mathrm{TRDT}_j^t$$

(14)　**if** $F_t<F_{t-1}$ **do**

(15)　　$\mathrm{LS}\leftarrow\mathrm{IS}$;

(16)　**end if**

(17) **end for**

(18)　Update centroids $\mathrm{ICon}=\{C_1,C_2,\cdots,C_M\}$ based on the sum of the shortest distance from all switches in LS_i to new controller C_i is minimized, $\mathrm{LCon}\leftarrow\mathrm{ICon}$;

(19) Output LCon, LS。

制域的交换机集合IS_i中(步骤6—步骤8)；若控制域负载超过效率区间上限$\beta\times\mathrm{MLCC}$，利用广度优先遍历选择距离已选控制器ICon_i最远、流量适中的交换机移出该控制域的交换机集合IS_i后加入TNS中(步骤9—步骤11)，在时间T内，以最小化F值为目标(步骤14—步骤16)重复调整多次(步骤1—步骤17)，最后在各域内以最小化交换机与控制器间传播时延之和为目标更新控制器位置(步骤18)，输出调整后的控制器集合LCon及交换机集合LS。

当网络流量发生变化时，M-CDA算法在任意时隙t检测各子网负载是否满足效率区间，是则保持不变，否则进行负载预调整直至各子网负载满足效率区间，对预调整后的子网分布重新计算F值，若F值相较于时隙$t-1$的F值更小，则接受调整，否则在时隙$t+1$时进行下一轮调整，且在时刻T重新调整各子网内控制器位置以进一步减少F值。时隙t的大小可根据网络流量情况进行动态调整以改变时间T内的调整次数，因此M-CDA算法可在调整各子网负载使其满足效率区间的同时，生成F值较优的子网分布。综上可知，M-CDA算法在满足以下两种情况时进行负载调整：(1)存在负载不满足效率区间的子网；(2)预调整后的F值较时隙$t-1$的F值更优。所以当存在负载不满足效率区间的子网时，M-CDA算法对该子网负载进行若干次优化调整使得各子网负载满足效率区间且F值较优后，停止对子网负载进行调整，避免了子网的频繁调整，从而保证了网络的稳定运行。综上，可以得到如下定理：

定理 2 M-CDA算法进行每一轮调整的时间复杂度为$O(n)$。

证明 M-CDA算法在时隙t时包括3个部分：计算不满足效率区间的子网以及标记各子网内流量较大的边界节点的时间均为$O(n)$，对任意需调整的子网选择流量适中的若干边界节点加入或换出的平均时间为$O(n/M)$。综上，进行每一轮调整的时间复杂度为

$$O(n)+O(n)+O((n/M)\times M)=O(n) \tag{16}$$

证毕

4 仿真实验

为了对本文提出的算法进行评估，设置控制器最大负载MLCC为1800 kflows/s[15]以及最大允许传播时延$T_{\max}$为40 ms，设置交换机上的平均排队时延T_q为0.1 ms[16]以及控制器的平均处理时延T_p为0.01 ms[16]，设置目标函数式(9)中网络负载均衡权值μ为0.7[13]，域内总请求时延权值ν为0.3[13]；在初始静态情况下，设置单位时间内流经网络任意节点的请求数量为50 kflows/s，逐次递增请求速率以改变子网数目，通过在真实网络拓扑Internet OS3E上部署IMCDS算法进行实验以评定网络负载均衡情况和时延大小，还将IMCDS分别应用于Noel, Darkstrand等多个不同类型的真实网络拓扑上，分别评定各网络负载均衡情况。在流量动态变化情况下，时间T内，设置单位时间内流经网络任意节点的请求数量满足于泊松分布[13]，通过在真实网络拓扑Internet OS3E上部署M-CDA算法以评定效率区间(α,β)的值，基于(α,β)的值评定网络时延大小，且将M-CDA分别应用于多个不同类型的真实网络拓扑上以评定各网络负载均衡情况。各网络拓扑节点数和链路数如表3所示，本文选择的仿真平台是matlab2016。本文对所提出的M-CDA算法采用分布式决策技术的部署方法[13]，即在各控制器上部署M-CDA算法使其分别进行独立决策：各控制器使用优化引擎确定需迁移的交换机，通过控制器通讯模块进行信息交互完成迁移操作。

表 3 网络拓扑节点数和链路数分布

网络名称	Noel	Darkstrand	OS3E	Arnes	Ntelos	Surfnet
节点数	19	28	34	34	48	50
链路数	35	31	43	49	61	73

4.1 初始静态情况

实验 1 将本文提出的IMCDS算法部署于Internet OS3E网络上，验证网络的负载均衡情况和时延大小。其中PSA算法源自参考文献[11], Optimized K-Means算法源自参考文献[5]，基于节点度的贪心算法(Greedy)划分网络的方法为：选择度最大的节点作为首个控制器后分配交换机，依次选择度最大的节点作为新的控制器并分配交换机，直到所有节点均被选中。设置初始时刻单位时间内流经网络任意节点的请求数量为50 kflows/s，逐次递增请求数量使得子网数M为2, 3, 4, 5, 6, 7, 8时，实验结果如图1所示：IMCDS的负载均衡情况均优于其他算法，且使网络负载均衡指标平均提高40.65%。其中Optimized K-Means算法是对标准K-Means算法的改进，避免了聚类过程陷入局部最优，但其仍以减少欧式距离为首要目标，随着子网数增加其负载均衡情况越差。基于节点度的贪心算法(Greedy)每次选择度最大的节点，当所需子网数增多时，容易陷入局部最优。PSA算法是基于Pareto最优的控制器部署算法，采用模拟退火方法选择节点避免陷入局部最优，在$M=6$时，其负载均衡情况与IMCDS

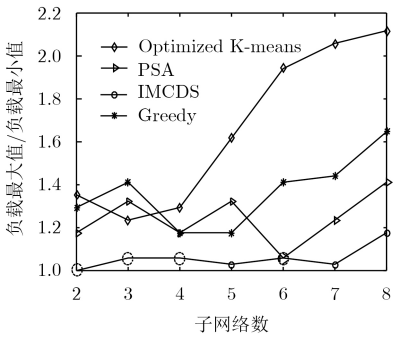

图 1 网络负载均衡情况

相同，其他情况下，负载均衡情况均差于IMCDS。CPP算法源自文献[1]，该文提出控制器部署的两个重要时延参考指标：Average-Case latency(ACCPP)和Worst-Case latency(WCCPP), ACCPP表示平均情况时延，即交换机向控制器发送请求的平均最小传播时延，WCCPP表示最坏情况时延，即交换机向控制器发送请求的最大传播时延。实验结果如图2所示：当子网数M为2, 3, 4, 5, 6, 7, 8时，IMCDS的平均时延接近于ACCPP，远小于WCCPP，由于CPP算法仅考虑传播时延指标，而不考虑负载情况，因此在所需控制器数量较少时，存在着密集区域所属控制器局部最优等问题，所以当子网络数$M=2$时，IMCDS与ACCPP差值最大，为2.33 ms，其他情况下差值均小于1 ms。

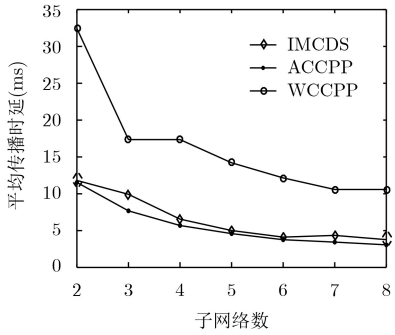

图 2 网络平均时延情况

实验 2　验证本文算法能够广泛应用于不同类型网络拓扑。将IMCDS算法分别应用于多个不同类型网络拓扑上，验证各网络负载均衡情况。按照网络节点个数依次选择Noel和Darkstrand等真实网络拓扑。实验结果如表4所示：在不同类型网络拓扑中采用IMCD划分网络后，各网络负载均衡指标NLBI接近于1，因此各网络负载均衡情况良好。其中Noel网络拓扑节点与边的数量均较少，网络节点较分散，在子网个数增加到7或8时，局部控制器易过载，使其负载均衡指标超过1.4。

表 4　初始静态情况下多网络拓扑的负载均衡情况

子网络数	Noel	Darkstrand	OS3E	Arnes	Ntelos	Surfnet
2	1.05	1.00	1.00	1.00	1.00	1.00
3	1.10	1.07	1.06	1.06	1.00	1.02
4	1.26	1.14	1.06	1.06	1.08	1.04
5	1.05	1.07	1.03	1.18	1.25	1.20
6	1.26	1.06	1.06	1.06	1.13	1.08
7	1.48	1.25	1.03	1.23	1.17	1.12
8	1.52	1.43	1.18	1.41	1.17	1.12

4.2　流量动态变化情况

实验 3　将本文提出的M-CDA算法分别部署于Internet OS3E网络中各控制器节点上，验证效率区间(α,β)不同区间值对目标函数F的影响，设置时间$T=60$ min，时间间隔$t=5$ min，在时间T内，设置单位时间内流经网络任意节点的请求数量满足于泊松分布[13]，将α与β分别在区间$(0,1)$上取不同值后多次实验得目标函数F的平均值均描述于图3上，实验结果如图3所示：α和β分别为0.39和0.82时，F值最小，由图中曲线可知效率区间(α,β)下限α过小或上限β过大均不利于最小化F值，原因在于控制器负载过低或过高易导致网络负载不均衡，因此设置效率区间(α,β)为$(0.39, 0.82)$。基于已设定(α,β)值，时间Ts内平均时延大小如图4所示：当子网数M为2, 3, 4, 5, 6, 7, 8时，与CPP[1]相比较，M-CDA算法的平均传播时延接近于ACCPP，与其平均差值为0.67 ms，远小于WC-CPP，但是当子网数大于6时，由于部分控制域陷入局部最优无法进行迭代调整，造成平均时延增大。

实验 4　基于以上实验已设定(α,β)值，将本文提出的M-CDA算法分别部署于多个不同类型网络的各控制器节点上，验证网络的负载均衡情况。实验结果如表5所示：当子网数M为2, 3, 4, 5, 6, 7, 8时，各网络负载均衡指标NLBI趋向1，因此各网络负载均衡情况均较优，但随着子网数的增加超过5时，各网络负载均衡情况变差，NLBI超过1.3，

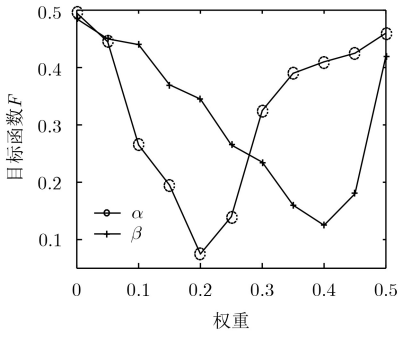

图 3 最优效率区间

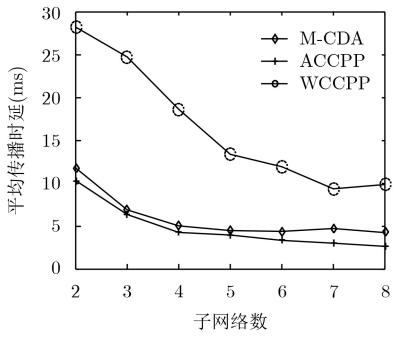

图 4 网络平均时延情况

表 5 动态情况下多网络拓扑的负载均衡情况

子网络数	Noel	Darkstrand	OS3E	Arnes	Ntelos	Surfnet
2	1.08	1.04	1.02	1.04	1.03	1.05
3	1.09	1.10	1.08	1.11	1.06	1.03
4	1.18	1.23	1.12	1.16	1.08	1.09
5	1.20	1.25	1.13	1.23	1.18	1.15
6	1.34	1.28	1.26	1.28	1.12	1.31
7	1.41	1.33	1.38	1.36	1.33	1.24
8	1.51	1.48	1.26	1.34	1.23	1.18

原因在于控制域数量增多易造成部分控制域迭代后陷入局部最优。

5 结束语

本文基于贪心算法和改进代价的最小生成树，提出算法依次解决初始静态下的控制器放置问题和流量动态变化下的区域优化调整问题。通过理论分析和仿真实验可知，本文的算法适用于以上两种情况，调整各子网负载使其满足效率区间的同时能够保证较低的请求时延。未来会考虑流量激增时的新增控制域操作以及控制器利用率过低时的休眠操作。

参 考 文 献

[1] HLLER B, SHEERWOOD 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] KOPONEN T, CASADO M, GUDE N, *et al*. Onix: A distributed control platform for large-scale production networks[C]. Usenix Conference on Operating Systems Design and Implementation, Vancouver, Canada, 2010: 351–364.

[3] TOOTOONCHIAN A and GANJALI Y. HyperFlow: A distributed control plane for OpenFlow[C]. Internet Network Management Conference on Research on Enterprise Networking, San Francisco, USA, 2010: 3–8.

[4] YU M, REXDORD J, FREEDMAN M J, *et al*. Scalable flow-based networking with DIFANE[J]. *ACM Sigcomm Computer Communication Review*, 2010, 40(4): 351–362. doi: 10.1145/1851275.1851224.

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

[6] 张栋, 郭俊杰, 吴春明. 层次型多中心的SDN控制器部署[J]. 电子学报, 2017, 45(3): 680–686. doi: 10.3969/j.issn.0372-2112.2017.03.027.

ZHANG Dong, GUO Junjie, and WU Chunming. Controller placement based on hierarchical multi-center SDN[J]. *Acta Electronica Sinica*, 2017, 45(3): 680–686. doi: 10.3969/j.issn.0372-2112.2017.03.027.

[7] SAHOO K S, SAHOO B, DASH R, *et al*. Optimal controller selection in software defined network using a greedy-SA algorithm[C]. International Conference on Computing for Sustainable Global Development, New Delhi, India, 2016: 2342–2346.

[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] 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.

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

[11] 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 & Service Management*, 2015, 12(1): 4–17. doi: 10.1109/TNSM.2015.2402432.

[12] JALILI A, AHMADI V, KESHTGARI M, *et al*. Controller placement in software-defined WAN using multi objective genetic algorithm[C]. International Conference on Knowledge-Based Engineering and Innovation, Tehran, Iran, 2016: 656–662. doi: 10.1109/KBEI.2015.7436121.

[13] RATH H K, REVVOORI V, NADAF S M, *et al*. Optimal controller placement in software defined networks (SDN) using a non-zero-sum game[C]. International Symposium on A World of Wireless, Mobile and Multimedia Networks, Sydney, Australia, 2014: 1–6. doi: 10.1109/WoWMoM.2014.6918987.

[14] 史久根, 郑伟. 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 and ZHU Wei. 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.

[15] 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.

[16] 覃匡宇, 黄传河, 王才华, 等. SDN网络中受时延和容量限制的多控制器均衡部署[J]. 通信学报, 2016, 37(11): 90–103. doi: 10.11959/j.issn.1000-436x.2016219.
QIN Kuangyu, HUANG Chuanhe, WANG Caihua, *et al*. Balanced multiple controllers placement with latency and capacity bound in software-defined network[J]. *Journal on Communications*, 2016, 37(11): 90–103. doi: 10.11959/j.issn.1000-436x.2016219.

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.

徐　皓：男，1994年生，硕士生，研究方向为软件定义网络、网络负载均衡和嵌入式系统.

张　径：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.

王　继：男，1993年生，硕士生，研究方向为软件定义网络、网络虚拟化和规则缓存.