SDN 中基于双向匹配的多控制器动态部署算法

胡涛，张建辉，孔维功，杨森，曹路佳
（国家数字交换系统工程技术研究中心，河南 郑州 450002）

摘 要：针对分布式软件定义网络（SDN, software defined networking）中控制器负载不均衡问题，提出一种基于双向匹配的多控制器动态部署算法。首先，周期性收集网络中跳数、时延和流量信息，分别构建交换机和控制器的匹配列表。然后，按照优化排序原则从 2 个列表中选取交换机和控制器实施双向匹配，并通过模拟退火算法优化匹配关系，实现分布式网络中多控制器的动态部署。仿真结果表明，与现有的方法相比，该算法能够合理配属交换机和控制器之间的连接关系，有效降低流请求排队时延，同时控制器负载均衡率至少提高了 17.9%。

关键词：软件定义网络；控制器；负载均衡；双向匹配
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000-436x.2018015

Dynamic deployment algorithm for multi-controllers based on bidirectional matching in software defined networking

HU Tao, ZHANG Jianhui, KONG Weigong, YANG Sen, CAO Lujia
National Digital Switching System Engineering R&D Center, Zhengzhou 450002, China

Abstract: Aiming at the controller load imbalance problem in distributed SDN, a multi-controller dynamic deployment algorithm based on bidirectional matching was proposed. Through collecting hop counts, delay and flow information in the network periodically, match lists of switch and controller was built respectively. According to the principle of optimal queuing, switches and controllers were selected from two match lists for implementing bidirectional matching, and the relationship of matching with the help of simulated annealing algorithm was optimized, which achieved dynamic deployment for multi-controller in distributed network. Results show that, compared with the existing approaches, this algorithm can match the connections between switches and controllers reasonably, and reduce the queue delay of flow request effectively. Moreover, and the controller load balancing rate has increased by 17.9% at least.

Key words: software defined network, controller, load balancing, bidirectional matching

1 引言

软件定义网络作为一种新型网络体系架构[1]，实现了控制平面和数据平面的完全解耦，将束缚在转发设备内的控制功能抽象到上层，能够灵活地管理网络且具有直接可编程[2]的特点，受到人们的广泛关注，在域内数据中心和域间数据中心[3]有着广阔的应用前景。

随着应用需求多元化和网络流量的不断增长，同时为了提高控制器的可扩展性，避免单点失效问题[4]，控制平面通常应用于分布式系统，采用多控制器部署，如 HyperFlow[5]、Kandoo[6]等。多控制器架构将交换机静态部署给控制器，由一个控制器管理多个交换机形成 SDN 子域。控制器间交互网

收稿日期：2017-02-13；**修回日期**：2017-06-12
基金项目：国家自然科学基金资助项目（No.61521003, No.61372121）；国家科技支撑计划基金资助项目（No.2014BAH30B01）；国家高技术研究发展计划（“863”计划）基金资助项目（No.2015AA016102）；河南省科技攻关计划基金资助项目（No.162102210034）
Foundation Items: The National Natural Science Foundation of China (No.61521003, No.61372121), The National Key Technology R&D Program (No.2014BAH30B01), The National High Technology Research and Development Program of China (863 Program) (No.2015AA016102), The Key Scientific and Technological Project of Henan Province (No.162102210034)

络状态信息，共同完成流请求处理。

多控制器架构的引入增强了网络的灵活性和可靠性，然而由于控制器与交换机配属失衡和 SDN 子域规划不合理，一旦子域中出现控制器连接交换机数量过多，或流量突发造成控制器处理容量不足时，很容易导致各子域网络中控制器负载差异较大，降低了网络的通信性能。

近年来，关于控制器负载均衡的研究可以分为 2 种。1) 从控制器角度分析，优化控制器的部署数量和位置[7~11]。此种方案主要通过合理部署控制器实现控制器负载均衡并提高控制平面可扩展性，不足之处在于需要同时考虑多种度量因素，部署策略僵化、算法复杂度较高。2) 从交换机角度分析，改进现有的交换机设计模式或调整交换机与控制器之间的连接关系[12~16]，保证控制器资源的合理利用。但交换机设计受限于已有的硬件条件，同时将交换机动态调整作为一种弹性控制方式，很容易造成网络架构不稳定，并且会产生大量的额外通信开销，控制器逻辑一致性和状态同步性较差。

针对上述研究中存在的问题，同时考虑交换机和控制器 2 个方面因素，以控制器负载均衡为目标，应用匹配思想[17]，提出了一种基于双向匹配的多控制器动态部署（MCDD, multi-controllers dynamic deployment）算法。本文的主要贡献和创新工作总结如下。

1) 对 SDN 多控制器部署过程进行建模研究，将交换机和控制器之间的流请求传输与处理刻化为排队模型，分析发现网络中时延和流量是影响控制器负载的 2 个因素。

2) 设计了基于双向匹配的 MCDD 算法。根据网络参数分别构建交换机和控制器的匹配列表，交换机匹配具有轻流量负载的控制器，保证数据分组处理速率；控制器综合考虑跳数、流请求速率和交换机历史流量信息，匹配列表中最优交换机，提升控制器资源利用率。同时，在双向匹配过程中引入模拟退火算法提高匹配效率，并优化匹配关系，实现多控制器负载均衡。

3) 在理论层面，通过数学推导论证，分析 MCDD 算法可行性。在实验层面，综合多种负载均衡性能评价指标，与现有的代表性算法进行对比，开展相关实验研究，验证 MCDD 算法的性能优势。

2 相关工作

近年来，控制器负载均衡作为 SDN 领域的研究热点，研究人员分别从控制器和交换机 2 个角度开展相关研究。

从控制器角度开展的研究主要考虑控制器部署状态问题。Heller 等[7]首次提出控制器部署和负载问题，注重网络中平均时延和最大时延 2 类因素，将交换机和控制器之间的连接转化为设备位置问题，构建相应数学模型，确定控制器最佳部署状态。但在该文献中并没有考虑网络流量的动态传输。Hock 等[8]将控制器部署转化为 Pareto 弹性优化问题，设定时延限制，失效容忍和负载均衡作为求解目标，通过 Pareto 优化去配置不同的性能度量，在时效性和算法准确性之间进行权衡。文献[9]提出一种基于 *K*-means 聚类划分的控制器部署方案，初始化 *K* 个聚类，并且基于节点距离为每个聚类分配一个中心，将该聚类中心作为控制器部署点。按照距离最短原则为控制器分配交换机，以迭代优化的方式进行聚类调整，优化控制器部署位置和子域规模，有效地降低了控制器通信时延和负载开销。文献[10]提出使用博弈论解决 SDN 控制器最优部署，在交换机到控制器时延、控制器间时延、控制器负载均衡 3 个目标之间进行博弈考虑，寻求一种均衡度量方法，确定控制器部署数量和位置。但由于算法复杂度较高，仅适应于小规模网络拓扑。文献[11]提出使用贪婪算法均衡控制器负载。通过对流请求的传输与处理进行分析，计算数据收集代价，流表安装代价和控制器同步代价，并建立相应的开销函数，求解负载均衡过程中控制器最小部署代价。

在 SDN 中，由于交换机的 Packet-in 请求是控制器负载的主要来源，因此，从交换机角度出发，相关学者提出优化交换机自身性能和调整交换机与控制器连接关系，从而均衡控制器的负载分配。文献[12]提出了 DIFANE 方案，其核心思想是设定网络中部分交换机为权威交换机。根据预先定义的分区规则和权威规则，交换机不需要频繁请求控制器，从而减轻了控制器负载。DevoFlow[13]采取规则复制和局部操作 2 种方式，减小了 OpenFlow 交换机和控制器的信息交互，从而有效地减轻和均衡了控制平面负载。文献[14]对控制器进行分层处理，构建控制器集合，通过一个分层的超级控制器进行协调。在不同的分组规则下基于单连接创建子网，每个控制器仅管理网络的一部分。但不合理的子网规划，容易导致交换机和控制器在局部网络域中出

现连接失衡。Chen 等[15]提出了一种 SDN 弹性控制机制，将交换机从高负载控制器迁移至低负载控制器实现了负载在控制平面的重新分配。同时，文献[15]还设计了相应的分布式逐跳算法，使不同控制器可以独立地运行算法逻辑对交换机实施迁移。不足之处在于，交换机迁移作为一种动态均衡方式会造成 SDN 不稳定，并且控制器间需要进行频繁通信去保持 SDN 控制逻辑一致性。

3　模型构建

3.1　问题重述

本节主要对多控制器部署过程中控制器负载不均衡问题进行重述。在现有的分布式 SDN 中，多控制器部署架构如图 1 所示。整个网络被划分为 2 个子域即 Subdomain_1 和 Subdomain_2。在 Subdomain_1 中，控制器 C_1 的剩余处理容量较小为 5 MB，控制 3 个交换机。Subdomain_2 中控制器 C_2 的剩余处理容量较大为 10 MB，仅控制 2 个交换机，则 C_1 和 C_2 处于负载不均衡状态。因此，在多控制器部署架构下所要解决的问题是，对于给定网络拓扑，在数据平面交换机连接已知，并且控制平面控制器状态确定的情况下，如何配属交换机和控制器的连接关系来构建合理的分布式 SDN 子域，从而实现多控制器负载均衡。

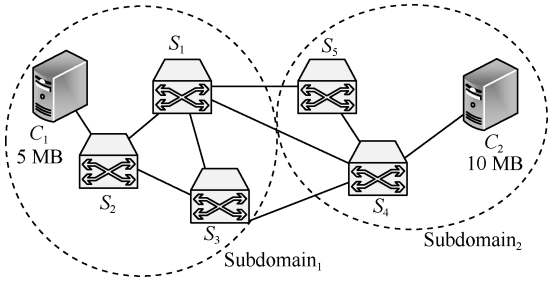


图 1　交换机和控制器部署失衡状态

3.2　网络建模和参数设定

根据图论的相关知识，对多控制器 SDN 实施建模。整个网络拓扑用无向图 $G=(V,E)$ 表示，其中，V 表示节点集合，E 表示节点间链路集合。在网络中总共包含 M 个控制器和 N 个交换机，有 $|V|=M+N$。控制器集合定义为 $C=\{c_1,c_2,\cdots,c_M\}$，控制器处理容量 $\alpha=\{\alpha_1,\alpha_2,\cdots,\alpha_M\}$，文献[16]对每个控制器设置相应的冗余因子 $\beta_j\in(0,1)$，保证控制器预留出足够的处理容量预防流量突发和进行状态同步。交换机集合定义为 $S=\{s_1,s_2,\cdots,s_N\}$。d_{ij} 网络中 2 个设备之间的跳数。由于真实的网络流量具有突发性和自相似性[18]，因此，本文设定交换机流请求速率是时间的函数，同时，将交换机产生的 Packet-in 消息作为流请求的主要内容，第 i 个交换机在时间 t 内的流请求速率可以表示为 $\lambda(t)_i$。考虑交换机在部署环境和承载网络服务方面差异较大（例如，核心交换机相比边缘交换机要承受更多的交换量），根据交换机历史流量信息，对每个交换机设定相应的转发因子 $\theta_i(t)\in(0,1]$，$\theta_i(t)$ 越大，则交换机 s_i 在相应的历史周期内转发的数据分组越多，该交换机对于控制器的处理容量需求也就越大。$x(t)_{ij}$ 是一个二进制数，表示在时间 t 内，第 i 个交换机是否成功连接到第 j 个控制器，如式(1)所示。因此，交换机和控制器之间的配署定义为 $N\times M$ 二进制矩阵 $\boldsymbol{\Lambda}$。为了满足动态约束，交换机仅选择一个控制器作为主控制器。

$$x(t)_{ij}=\begin{cases}1, \text{成功连接}\\ 0, \text{其他}\end{cases} \tag{1}$$

SDN 多控制器架构的核心思想是逻辑上集中但物理上分布，整个网络被划分为多个分布式子域，同时每个子域由特定的控制器进行集中式控制。交换机通过向控制器发送 Packet-in 消息实现路由信息的查询与获取。本文将 OpenFlow 架构下流量传输抽象为请求与服务的排队过程。考虑到现有的 OpenFlow 交换机均为多核交换机，支持多端口多 VLAN 背板上数据的并行处理[19]。同时，ONOS，OpenDaylight 等主流控制器均采用多线程处理方式。因此，交换机和控制器之间的通信过程可以抽象为 G/M/m 转发队列模型，其中，G 表示交换机的 Packet-in 消息到达是一般过程，M 为控制器对于流请求的处理是马尔可夫型，m 控制器当前线程数，表明控制器有能力同时连接多个交换机。当交换机流请求到来时，控制器会在可用的线程空间内对 Packet-in 消息进行并行处理。基于该通信模型，对网络中相关参数进行计算，包括交换机到控制器的平均时延、控制器流量负载和控制器负载均衡。

1) 交换机到控制器的平均时延

第 j 个控制器在时间 t 内需要处理的流请求是和它相连的所有交换机的流请求速率之和，设为 $L(t)_j$，如式(2)所示，其中，各交换机之间的流请求相互独立。

$$L(t)_j = \sum_{i=1}^{N} \lambda(t)_i x(t)_{ij} \tag{2}$$

应用 Little 原理，得到流请求传输的平均停留时间，即流请求传输排队时延为 $\omega(t)_j$，如式(3)所示，其中，$\alpha(t)_j$ 为 t 时刻控制器 c_j 的剩余处理容量，且 $0 < \alpha(t)_j \leqslant \alpha_j$。

$$\omega(t)_j = \sum_i \frac{1}{\alpha(t)_j - \lambda(t)_{ij}} \tag{3}$$

则第 j 个控制器的平均处理时间为

$$\Delta(t)_j = \frac{\omega(t)_j \alpha(t)_j}{\alpha(t)_j - tL(t)_j} \tag{4}$$

因此，交换机到控制器的平均时延为

$$\tau(t) = \frac{1}{\sum_{j=1}^{M} L(t)_j} \left(\sum_{j=1}^{M} (\Delta(t)_j + \omega(t)_j) L(t)_j \right) \tag{5}$$

2) 控制器流量负载

现有的 SDN 以带内通信为主，数据流传输和交换机请求消息处理共享链路带宽。假设每一个 Packet-in 请求需要被发送给至少一个控制器，同时控制器间需要进行周期性通信以更新控制逻辑，确保网络视图一致性。因此，SDN 中控制流量负载主要包含 2 个部分内容：转发开销和状态同步开销。转发开销 P_f 主要指交换机请求的 Packet-in 消息被控制器处理并下发流表给交换机所带来的网络开销，如式(6)所示，其中，v_r 为轮询交换机的平均速率，与连接到交换机的平均链路数目有关。状态同步开销 P_s 主要指控制器间交互状态信息所产生的开销，如式(7)所示，其中，v_s 为控制器状态信息的平均传输速率，但在网络中 v_s 小于 v_r，因为控制器间不会同步子域所有信息。因此，SDN 控制器流量负载是控制器转发开销和状态同步开销的线性相加，如式(8)所示。

$$P_f = v_r \sum_{i \in N} \sum_{j \in M} \lambda_i d_{ij} x(t)_{ij} \tag{6}$$

$$P_s = v_s \sum_{j,k \in M} d_{jk} \tag{7}$$

$$\eta(t) = v_r \sum_{i \in N} \sum_{j \in M} \lambda_i d_{ij} x(t)_{ij} + v_s \sum_{j,k \in M} d_{jk} \tag{8}$$

3) 控制器负载均衡

为了保证控制器负载的均衡分布，需要合理地配属交换机和控制器之间的连接关系，确保控制器流量负载处于正常范围内，同时，减少控制器和交换机之间的时延。因此，目标函数如式(9)所示，权值 $\delta \in (0,1)$。

目标函数： $\min \ \ Object = [\delta\tau(t) + (1-\delta)\eta(t)]$ (9)

约束： $\forall i, j, \ x(t)_{ij} \in \{0,1\}$ (10)

$$\forall j, \ L(t)_j \leqslant \beta_j \alpha(t)_j \tag{11}$$

$$\forall i, \ \sum_{j=1}^{M} x(t)_{ij} = 1 \tag{12}$$

式(10)对网络中所有设备的连接关系进行限定。式(11)确保网络中没有控制器出现过载状况。式(12)表示在给定时间内所有交换机都能精确连接到主控制器。

4 算法设计

4.1 匹配机制

匹配机制的总体思路是，对于给定的网络拓扑，通过构建相应设备的匹配列表，实现交换机和控制器的双向匹配。

定义 1 匹配，定义为 $a \succ_c b$，表示节点 c 在节点 a 和节点 b 之间选择连接节点 a，则 a 和 c 之间形成匹配关系。例如，在图 1 中，交换机 s_3 在控制器 c_1 和 c_2 之间选择 c_1，则表示 s_3 匹配 c_1，即 $c_1 \succ_{s_3} c_2$。

由于设备性质不同，因此交换机和控制器在选择匹配对象时考虑不同的约束条件。

交换机目标。交换机更愿意选择具有大处理容量的控制器，以避免在数据分组传输过程中产生流请求拥塞。交换机匹配列表定义如下。

定义 2 交换机匹配列表。第 i 个交换机 s_i 的匹配列表为 $\Gamma(s_i) = \{c_j, \cdots\}$。该列表中所包含的控制器按照处理容量 β_j 大小进行降序排列且都能满足 s_i 的流请求处理需求。

控制器目标。考虑交换机连接控制器带来的通信开销（Packet-in 数据分组请求、流表下发等）和控制器处理容量消耗，控制器同时考虑跳数 d_{ij}、流请求速率 $\lambda(t)_i$ 和交换机转发因子 $\theta(t)_i$ 这 3 种指标，并通过计算 $d_{ij}\lambda(t)_i\theta(t)_i$ 对交换机选择进行量化评价。控制器匹配列表定义如下。

定义 3 控制器匹配列表。第 j 个控制器 c_j 的匹配列表为 $\Gamma(c_j) = \{s_i, \cdots\}$。该列表中所包含的交换

机按照 $\min(d_{ij}\lambda(t)_i\theta(t)_i)$ 原则进行降序排列，同时保证控制器负载未超出处理容量限制。

定义 4　双向匹配，定义为 $s_i \Theta c_j$，表示在一组匹配关系中，若交换机 s_i 优先选择控制器 c_j 作为所属控制器；同时控制器 c_j 优先连接交换机 s_i，则 s_i 和 c_j 形成双向匹配，如图 2 所示。交换机 s_1 可供选择的控制器有 c_1 和 c_2，控制器 c_1 可供连接的交换机有 s_1、s_2 和 s_3。若交换机 s_1 优选 c_1，且控制器 c_1 优先连接 s_1，则 c_1 和 s_1 之间实现双向匹配。

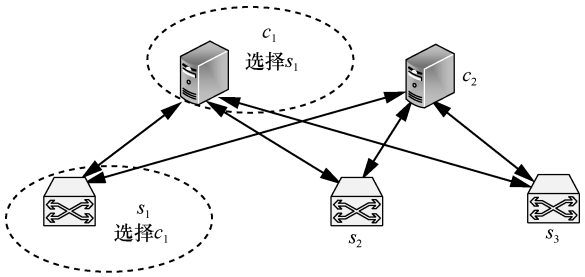


图 2　交换机和控制器实施双向匹配

因此，交换机和控制器完成双向匹配需要满足以下条件。

$$c_j \succ_{s_i} (\Gamma(s_i)) \tag{13}$$

$$s_i \succ_{c_j} (\Gamma(c_j)) \tag{14}$$

$$\begin{cases} L(t)_j + \lambda(t)_i \leqslant \alpha(t)_j \beta_j \\ L(t)_j - \sum_i \lambda(t)_i + \lambda(t)_i \leqslant \alpha(t)_j \beta_j \end{cases} \tag{15}$$

式(13)表示交换机 s_i 在列表 $\Gamma(s_i)$ 中选择控制器 c_j 作为目标控制器。式(14)表示控制器 c_j 在列表 $\Gamma(c_j)$ 中选择连接交换机 s_i。式(15)表示双向匹配必须满足控制器处理容量约束。

4.2　MCDD 算法

在双向匹配原则的基础上，本文提出多控制器动态部署（MCDD）算法。区别于以往算法仅从交换机角度或控制器角度进行部署选择，MCDD 算法同时考虑交换机和控制器 2 个匹配原则。根据式(9)定义的控制器负载均衡目标函数可知，在优化控制器负载时，需要对平均时延 $\tau(t)$ 和控制器流量负载 $\eta(t)$ 进行加权求和。因此，MCDD 算法在实施交换机和控制器双向匹配过程中，将控制器负载均衡转化为多目标混合优化问题，基于模拟退火算法[20]实施求解。模拟退火作为一种启发式算法，源于固体退火原理，综合了统计物理学知识，通过“降温”的方式寻求系统平衡状态，从而得到近似最优解。它的渐进收敛性（理论上已被证明以概率 1 收敛于全局最优解）和简单通用性适合于本文所要求解的优化目标。

MCDD 算法的基本流程如算法 1 所示。基于定义 2 和定义 3 构建的匹配列表（第 2)行），交换机同时向多个控制器发出匹配请求。控制器接收请求后，在容量允许的情况下，和控制器匹配列表中交换机进行双向匹配（第 3)~5)行）。通过不断调用目标函数，借助模拟退火算法，对匹配列表 $\Gamma(s_p)$ 和 $\Gamma(c_q)$ 中选取的匹配元素实施优化，寻求使目标函数达到最小的匹配方式（第 6)~17)行）。该进程一直重复，直至网络中不产生任何匹配请求，所有设备形成有效连接，得到网络部署矩阵 $\boldsymbol{\Lambda}$（第 24)~25)行）。

算法 1　多控制器动态部署（MCDD）

输入　$G=(V,E)$

交换机流请求速率 $\lambda(t)_i$

控制器处理容量 $\alpha(t)_j$

控制器冗余因子 β_j

输出　交换机和控制器连接关系 $x(t)_{ij}$

交换机和控制器之间的部署矩阵 $\boldsymbol{\Lambda}$

1) 遍历网络中所有节点，得到 $\lambda(t)_i$，$\theta(t)_i$，$\alpha(t)_j$，β_j

2) 构建交换机和控制器的匹配列表：$\Gamma(s_i)$，$\Gamma(c_j)$

3) 得到 $c_j \succ_{s_i} (\Gamma(s_i))$ 和 $s_i \succ_{c_j} (\Gamma(c_j))$

4) if　$L(t)_j \leqslant \beta_j \alpha(t)_j$ & $L(t)_j - \sum_i \lambda(t)_i + \lambda(t)_i \leqslant \alpha(t)_j \beta_j$

5)　while ($\Gamma(s_i) \neq \Phi \bigcup \Gamma(c_j) \neq \Phi$)

6)　　for each　$s_p \in \Gamma(c_q), c_q \in F(s_p)$　do

7)　　$Object_{\text{now}} = 0$，初始退火温度 $T = T_0$

8)　　选取匹配组合 (s_p, c_q)

9)　　while ($T > T_{\min} \bigcup \min Object$)

10)　　$Object_{\text{new}} = [\delta\tau(t) + (1-\delta)\eta(t)]$

11)　　$\xi = Object_{\text{new}} - Object_{\text{now}}$

12)　　if ($\xi \geqslant 0$)

13)　　$Object_{\text{now}} \leftarrow Object_{\text{new}}$

14)　　else $Object_{\text{new}} \leftarrow$ 以 $e^{\frac{\zeta}{T}}$ 概率将 $Object_{\text{new}}$ 赋值

15) end if
16) $T = T\sigma$ (σ 为降温因子)
17) end while
18) 输出一组匹配结果 $s_p \Theta c_q$
19) $\Gamma(s_i) = \Gamma(s_i) \setminus \{c_q\}, \Gamma(c_j) = \Gamma(c_j) \setminus \{s_p\}$
20) end while
21) else $L(t)_j > \beta_j \alpha(t)_j$
22) 在 $\Gamma(s_i)$ 中按次序选择所属控制器
23) end if
24) 匹配交换机和控制器得到 $x(t)_{ij}$
25) 由连接关系 $x(t)_{ij}$ 构建部署矩阵 $\boldsymbol{\Lambda}$

MCDD 算法的复杂度与匹配列表中元素个数以及模拟退火的降温次数有关。在构建匹配列表时，需要对网络中所有设备元素进行轮询，复杂度为 $O(MN)$。在执行双向匹配时借助模拟退火算法实施负载均衡优化，其复杂度主要与初始温度 T_0，冷却温度 $T_{\min}$ 以及降温因子 σ 有关，复杂度为 $O(I)$ 且 $O(I) < O(M \log N)$，其中，I 为降温次数。因此 MCDD 算法复杂度最大为 $O(N^2)$。在数据处理时，只需要进行简单的线性运算，包括求和、相乘等，无复杂计算，且模拟退火算法以概率 1 收敛。最终经过有限次迭代后，MCDD 算法快速收敛，具有实时性和有效性。

4.3 算法可行性证明

4.3.1 稳定的连接关系

首先，证明通过算法 1 可以产生一个稳定的连接关系。

证明 因为交换机选择控制器依赖于控制器的处理容量 $\alpha(t)_j$ 和冗余因子 β_j，如式(4)和式(11)所示，且交换机匹配列表中元素相同，只是排列顺序不同。因此，在每一轮中，所有交换机将会提议给相同的控制器，然后，控制器和交换机进行双向匹配。

情况 1 对于 $c_j \succ_{s_i} (\Gamma(s_i))$，$s_i \succ_{c_j} (\Gamma(c_j))$，$L(t)_j + \lambda(t)_i \leqslant \alpha(t)_j \beta_j$ 的情况，在 c_j 中可匹配列表为空，此时将收缩交换机的选取条件，直至满足控制器处理容量需求。

情况 2 对于 $c_j \succ_{s_i} (\Gamma(s_i))$，$s_i \succ_{c_j} (\Gamma(c_j))$，$L(t)_j - \sum_i \lambda(t)_i + \lambda(t)_i \leqslant \alpha(t)_j \beta_j$ 的情况，表明此时允许部分交换机进行迭代。然而，在每次迭代中，控制器将会收到所有匹配请求，并从中选取交换机进行匹配。因此，对于情况 1，控制器匹配列表为空将不会发生。

综上，整个匹配进程将不会遗漏任何交换机和控制器，二者之间连接是稳定且可靠的，因此，原命题得到证明。

4.3.2 算法优化性能分析

为了验证 MCDD 算法的可行性，下面进行数学推导论证，同理想情况下的最优解进行对比分析。

在此简化约束条件，假设控制器的处理容量都为 α，且处理流请求的平均速率为 $\overline{L(t)}$，在时间 t 内连接到第 j 个控制器的交换机具有最小请求速率 $\lambda(t)_j^{\min}$，且所有交换机都具有相同的转发因子 $\theta(t)_i = 1$。

理想状态下，当输入请求在所有交换机中都是等值分布时，此时控制器负载最小，如式(16)所示。

$$\tau(t)^{\min} = \frac{|V|^2 M \dfrac{\overline{L(t)}}{\alpha - \overline{L(t)}}}{\sum_{j=1}^{M} L(t)_j} \tag{16}$$

当算法结束时，$\forall i, j, L(t)_j - L(t)_i < \tau(t)_j^{\min}$。

$L(t)_i < \overline{L(t)} < L(t)_j$ 且 $L(t)_j < L(t)_i + L(t)_j^{\min} < \overline{L(t)} + \tau(t)_j^{\min}$，所以 $L(t)_j - \overline{L(t)} < \tau(t)_j^{\min}$。

当 MCDD 算法形成稳定连接时，控制器可以分为 2 个子集 $C = C_{\text{over}} \bigcup C_{\text{below}}$（$C_{\text{over}}$ 中负载大于 $\overline{L(t)}$，C_{below} 中负载小于 $\overline{L(t)}$）。

下面，计算 MCDD 算法同理想状态（如式(16)所示）的差距。

$$\begin{aligned}\varepsilon(\tau(t)) &= \frac{|V|^2}{\sum_{i=1}^{M} L(t)_i} \{\sum_{i=1}^{M} [\frac{L(t)_i}{\alpha - L(t)_i} - \frac{\overline{L(t)_i}}{\alpha - \overline{L(t)_i}}]\} \\ &< \frac{|V|^2}{\sum_{i=1}^{M} L(t)_i} \sum_{i \in C_{\text{over}}} \frac{\alpha(L(t)_i - \overline{L(t)})}{(\alpha - L(t)_i)(\alpha - \overline{L(t)})}\end{aligned} \tag{17}$$

可以看出式(17)的右边随着 $(L(t)_i - \overline{L(t)})$ 单调增加，所以有

$$\varepsilon(\tau(t)) < \frac{|V|^2}{\sum_{i=1}^{M} L(t)_i} \frac{M \alpha \tau(t)^{\max}}{(\alpha - \overline{L(t)} - \tau(t)^{\max})(\alpha - \overline{L(t)})} \tag{18}$$

$$\tau(t)^{\max} = \max(\tau(t)_i^{\min}), i \in C_{\text{over}} \tag{19}$$

因此，MCDD 算法与理想状况的性能比值为

$$\rho = 1 + \frac{\alpha \tau(t)^{\max}}{\overline{L(t)}(\alpha - \overline{L(t)} - \tau(t)^{\max})} \tag{20}$$

由于 α 和 $\overline{L(t)}$ 都是已知，且交换机的流请求速率远小于控制器处理容量，$\tau(t)^{\max}$ 可以被认为是一个小数字，因此，比值 ρ 是一个略大于 1 的正整数，接近理想状态下最优结果，验证了 MCDD 算法的可行性。

5　仿真分析

5.1　仿真环境搭建

关于仿真环境和实验参数，本文做出如下说明。

1) 实验平台

在控制器选取方面，本文采用 OpenDaylight 作为实验控制器，同时在斯坦福大学研发的 Mininet 平台上进行测试。OpenDaylight 控制器基于 Java 语言编写，运行于 JVM 上，支持多种版本的 OpenFlow 协议。为了保证实验效率，将 Mininet 和 OpenDaylight 以虚拟机的形式部署在 2 个不同的物理设备。实验机器的配置为 Intel Corei7 3.4 GHz 8 GB RAM，并配属一个 2 Gbit/s 的网卡。

2) 拓扑选择和算法实现

实验拓扑采用具有较高认可度的 Internet2 OS3E 网络拓扑。OS3E 拓扑总共包含 34 个节点和 42 条链路，网络中所有节点都具备部署交换机或控制器的能力，且各节点之间的统计相互独立。本文算法采用 Java 语言在 OpenDaylight 应用层进行实现，并借助 Matlab 工具对实验结果进行分析。

3) 仿真参数设定

对交换机流请求速率 $\lambda(t)_i$ 进行差异化处理用于模拟真实的流量特征，请求速率范围为 100 ~ 500 kB/s。控制器处理能力基本相同，处理容量 α_j 为 8 MB。根据文献[12]，设控制器冗余因子 β_j 在区间[0.9,1]任意取值，轮询一个交换机的平均速率 $v_r = 10$ kB/s，控制器状态同步信息的传输速率 $v_s = 1$ kB/s。对比文献[16]中部分设定，目标函数权值 δ 只是为了平衡时延和控制流量对于控制器负载的影响，改变之后，各因素的相对关系变化不大。因此，本文设定 δ 的取值为 0.5。

5.2　仿真验证与分析

为了验证 MCDD 算法的控制器负载均衡性能，在这里同文献[9]中 *K*-means 控制器均衡部署算法和文献[15]中控制器弹性控制方法（ECA, elastic control approach）进行对比。*K*-means 方法依据距离原则对节点进行聚类操作，选取聚类中心作为控制器部署点，通过聚类分割均衡控制器负载。ECA 将交换机动态分配作为算法核心，按照负载自适应方式选取交换机实施迁移，从而调整控制器负载分布。

5.2.1　实验 1

本实验主要验证不同算法在 SDN 子域规划方面性能。子域规划作为判定控制器负载均衡的重要指标，在 SDN 中，控制器管理的交换机数量越均衡，则子域规划越合理，控制器负载均衡性能越好。基于 OS3E 网络拓扑，类比文献[7]中拓扑设定，在整个网络中共部署 5 个具有相同条件的控制器，但交换机与控制器之间的匹配关系并未确定，对比 3 种算法在交换机分配和子域规划方面的性能，实验结果如图 3 所示。从宏观拓扑角度进行分析，*K*-means 对于子域的规划效果不够理想，各子域中交换机数量差异较大。ECA 和 MCDD 虽然都能够实现交换机在各子域的均衡部署，但 ECA 中存在跨域节点，会导致严重的跨域通信问题。

为了更加清楚地对比不同算法的子域规划效果，对图 3 中实验数据进行统计汇总与分析，结果如图 4 所示。在相同的控制器部署条件下，*K*-means 中控制器管理的交换机个数相差最大，其中，控制器 c_1 管理的交换机数量是控制器 c_2 和 c_4 结果的 2 倍。这是因为 *K*-means 在规划过程中，只根据设备间距离进行节点聚合，当节点间距离差异较大时，聚类操作容易陷入局部最优，造成交换机区域性聚集，子域规划效果差。ECA 和 MCDD 都较好地实现了交换机和控制器的均衡连接。但由于 ECA 在动态分配时，将交换机分配给剩余处理容量最大的控制器，虽然可以在一定程度上均衡控制器负载，但忽略了子域间节点连通性，在交换机和控制器之间产生跨域通信问题。相比较而言，MCDD 从交换机和控制器 2 个角度进行分析，基于模拟退火算法对双向匹配关系进行优化，保证了子域规划的合理性，同时控制器匹配列表考虑设备间距离、流请求速率和交换机历史流量信息，避免了仅考虑单一网络度量产生的节点孤立和跨域交互问题。根据交换机部署数量标准差对分布式子域的节点均衡率进行量化，经过归一化处理后，MCDD 的节点均衡率达到了 0.872，相较于 *K*-means（节点均衡率 0.517）和 ECA（节点均衡率 0.743），具有明显的部署优势。

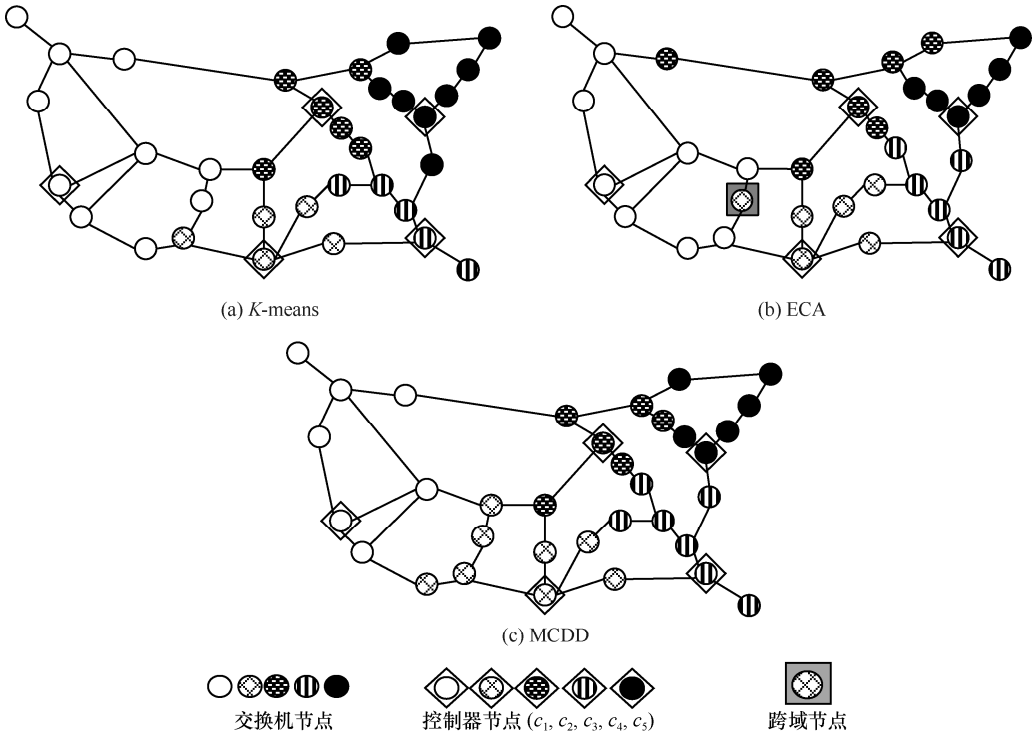


图 3　OS3E 网络子域规划结果

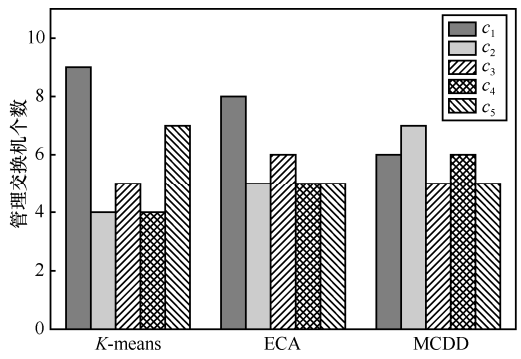


图 4　3 种算法得到的交换机部署结果对比

5.2.2　实验 2

在实验 1 拓扑规划的基础上，本实验对于 3 种算法的流请求排队时延进行评估。假设数据分组在光纤电缆中传输时，传输速率可靠稳定，无分组丢失现象。为了消除实验随机误差，3 种算法均在相同的实验条件下运行 30 次，记录仿真数据，以测量平均值作为实验结果，如图 5 所示。

图 5 中 Subdomain_1~Subdomain_5 分别表示控制器 c_1~c_5 所在的 SDN 子域。从整体来看，*K*-means 中各子域的流请求排队时延较高；ECA 在 Subdomain_2 的流请求排队时延达到了近 45 ms，而在其他子域的排队时延基本上都小于 20 ms；MCDD 各子域的流请求排队时延相差不大，且均处于较低值（最大时延 23 ms）。这是因为 *K*-means 方案虽然优化了控制器的位置选择，但对子域内交换机数量缺乏规划，在控制器处理性能相同的条件下，交换机的过量部署会导致子域中流请求处于高排队时延状态。ECA 作为一种弹性控制方式，虽然能够快速转地移控制器负载，保证控制器高处理性能，但很容易在交换机迁移过程中产生跨域通信问题，网络状态的不稳定反而导致流请求排队时延急剧增加。MCDD 对交换机和控制器进行双向匹配，考虑流请求速率和跳数因素，使交换机自适应地连接剩余处理容量最大的控制器，流请求被快速处理，各子域排队时延较低。

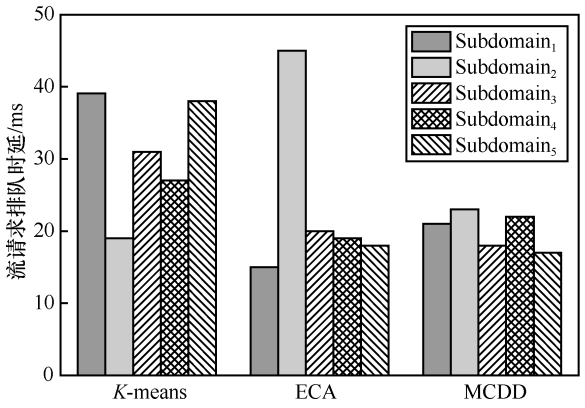


图 5　3 种算法得到的流请求排队时延对比

5.2.3　实验 3

本实验对比3种算法在不同流请求状况下控制器负载均衡率及相应的网络开销变化情况。同样基于OS3E 网络拓扑，利用流量生成器在交换机中产生持续不断的流请求，如图 6 所示。整个过程可以划分为 2 个阶段。阶段 1（0~6 h）：所有交换机的流请求速率均小于 200 kB/s，流量分布具有自相似性。阶段 2（7~12 h）：在交换机中产生大量流请求，模拟流量突发状况，直至网络中所有交换机的流请求速率均达到峰值。经过多次重复实验，记录实验数据。参考仿真参数设定，以各控制器经过归一化处理后的负载标准差作为依据，计算控制器负载均衡率，并且得到 3 种算法在均衡控制器负载过程中产生的转发开销 P_f 和状态同步开销 P_s。实验结果如图 7~图 9 所示。

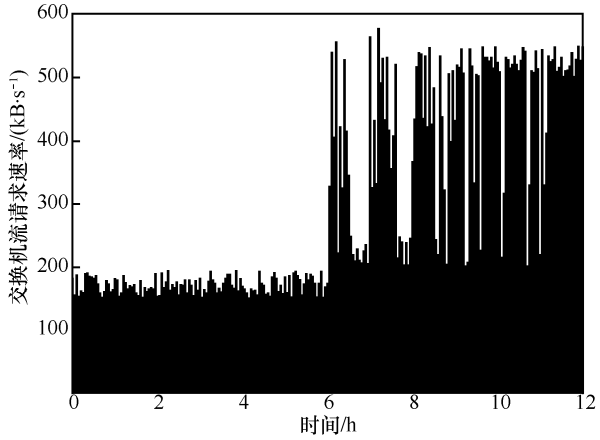


图 6　OS3E 网络交换机流请求速率

在图 7 中，当所有交换机的流请求速率都较低且具有自相似特性时（0~6 h），3 种算法的控制器负载均衡率基本保持平稳状态，其中，MCDD 的控制器负载均衡率最高，*K*-means 次之，ECA 最低。这说明 MCDD 通过双向匹配可以有效实现稳定网络状态下交换机和控制器之间的合理连接，保证了控制器负载的均衡分布。虽然 *K*-means 根据节点间距离最短原则进行聚类划分，已经具有一定的均衡效果，但它仅考虑控制器如何部署而忽略了交换机的合理配置，对于控制器负载均衡率的提升仍然不够显著。ECA 的负载自适应调节方式在低流请求速率状况下难以体现出动态调整的优势，控制器负载均衡率较低。当网络中交换机逐渐出现流量突发状况时（7~10 h），部分控制器出现负载溢出。*K*-means 基于聚类分割得到的单个网络子域规模较小，仅能预防少数流请求突发状况，一旦突发流数量增长过快，控制器负载状况更加恶化，负载均衡率快速下降。虽然 ECA 通过迁移交换机能有效消除控制器过载问题，但无法实现控制器负载的细粒度调整，整体效果优于 *K*-means，和 MCDD 相比仍然较差。MCDD 在实施双向匹配连接时就已经预先设置冗余因子和转发因子，保证控制器预留出部分冗余容量处理突发流请求，同时，根据交换机转发数据分组的历史信息，将有流请求突发倾向的交换机预先连接至具有较大剩余处理容量的控制器。因此，MCDD 能够有效应对较多数量的突发流请求，将控制器负载均衡率维持在较高水平，相比于其他 2 种算法，控制器负载均衡率至少提高了 17.9%。当所有交换机都产生流量突发问题时（11~12 h），3 种算法全部失效，尽管此时负载均衡率接近 1，但控制器均处于高负载状态，网络发生瘫痪，必须增加新控制器或提高控制器性能来改善网络通信质量。

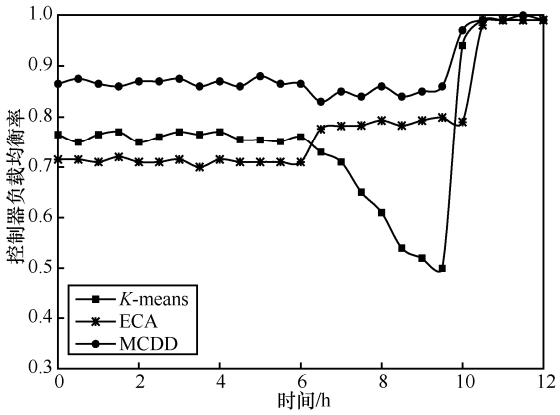


图 7　控制器负载均衡率

在图 8 和图 9 中分别显示了 3 种算法的转发开销和状态同步开销对比情况。在流量自相似状况下，网络状态相对稳定，无过载控制器产生，因此，3 种算法的状态同步开销相差不大。由于 MCDD 中控制器需要收集交换机的历史转发信息，所以转发开销略高于 *K*-means 和 ECA。当网络中出现流量突发状况时，转发开销和状态同步开销都有所上升。ECA 的 2 类开销上升幅度最大，这是因为交换机迁移需要在网络中交互和同步大量设备状态信息。MCDD 和 *K*-means 都采用先验式网络构建方式，子域中交换机和控制器连接关系基本保持稳定，当流请求速率增加时，转发开销和状态同步开销也会有所提高，但与 ECA 相比，仍处于较低水平。

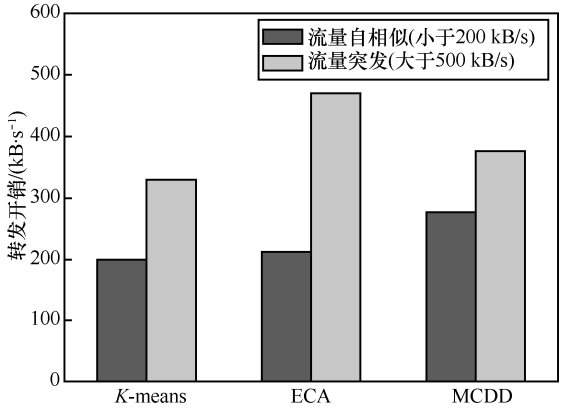


图 8　转发开销

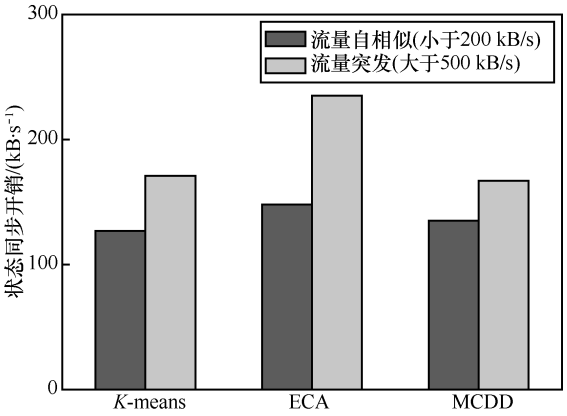


图 9　状态同步开销

综合比较图 7~图 9 的实验结果可以得出，相比于 *K*-means 和 ECA，MCDD 具有较好的控制器负载均衡性能，并且在应对网络中流量突发状况时，产生的转发开销和状态同步开销均在可接受范围之内。

6　结束语

本文针对分布式软件定义网络中存在的多控制器负载不均衡问题，提出了一种基于双向匹配的多控制器动态部署算法。通过对 SDN 多控制器网络实施建模，分析时延和控制流量对于控制器负载均衡性能的影响，并分别构建交换机和控制器的匹配列表。根据设定的匹配规则选择列表中元素进行双向匹配，借助模拟退火算法提升匹配效率，优化了网络中设备间连接关系，实现多控制器的动态部署。通过数学推导与分析，论证了算法可行性。同时，基于多种评估度量，设置一系列的仿真对比实验，结果表明，MCDD 算法在子域规划，流请求排队时延和控制器负载均衡率方面相比其他算法具有明显优势。未来的研究工作将围绕以下 2 个方面内容展开：1) 在进行多控制器部署时，加入链路失效和交换机失效模型；2) 将时延问题进一步拓展到交换机间时延和控制器间时延，实施多目标优化。

参考文献：

[1] 左青云, 陈鸣, 赵广松, 等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报, 2013, 24(5): 1078-1097.
ZUO Q Y, CHEN M, ZHAO G S, et al. Research on OpenFlow-based SDN technologies[J]. Journal of Software, 2013, 24(5): 1078-1097.

[2] MCKEOWN N, ANDERSON T, BALAKRISHAN H, et al. OpenFlow: enabling innovation in campus network[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[3] FARES M A, RADHAKRISHNAN S, et al. Hedera: dynamic flow scheduling for data center networks[C]//USENIX NSDI. 2010: 71-78.

[4] TOOTOONCHIAN A, GORBUNOV S, GANJALI Y, et al. On controller performance in software-defined networks[C]//USENIX HotICE. 2012: 1-6.

[5] TOOTOONCHIAN A, GANJALI Y. Hyperflow: a distributed control plane for OpenFlow[C]//2010 Internet Network Management Conference on Research on Enterprise Networking. 2010.

[6] HASSAS Y S. Kandoo: a framework for efficient and scalable offloading of control applications[C]//First Workshop on Hot Topics in Software Defined Networks. ACM, 2012: 19-24.

[7] HELLER B, SHERWOOD R, MCKEOWN N. The controller placement problem[C]//First Workshop on Hot Topics in Software Defined Networks. 2012: 7-12.

[8] HOCK D, GEBERT S, HARTMANN M, et al. POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks[C]//Network Operations and Management Symposium (NOMS). 2014: 1-2.

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

[10] KSENTINI A, BAGAA M, TALEB T. On using bargaining game for optimal placement of SDN controllers[C]//2016 IEEE International Conference on Communications (ICC). 2016: 1-6.

[11] OBADIA M, BOUET M, ROUGIER J L. A greedy approach for minimizing SDN control overhead[C]//The 2015 1st IEEE Conference on Network Softwarization (NetSoft). 2015: 1-5.

[12] YU M, REXFORD J, FREEDMAN M J, et al. Scalable flow-based networking with DIFANE[C]//In Proc ACM SIGCOMM, 2010: 1-6.

[13] CURTIS A R, MOGUL J C, TOURRILHES J, et al. DevoFlow: scaling flow management for highperformance networks[C]//SIGCOMM Toronto. 2011: 254-265.

[14] ZHANG H L, GUO X. SDN-based load balancing strategy for server cluster[C]//2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems. 2014: 662-667.

[15] CHEN H C, CHENG G Z, WANG Z M. A game-theoretic approach to elastic control in software-defined networking[J]. China Communication, 2016 (5):103-109.

[16] MÜLLER L F, OLIVEIRA R R. Survivor: an enhanced controller placement strategy for improving SDN survivability[C]//2014 IEEE Global Communications Conference. 2014:1909-1915.

[17] ROTH A E, SOTOMAYOR M A O. Two-sided matching: a study in game-theoretic modeling and analysis[M]. Cambrideg: Cambridge University Press,1992.

[18] VAUGHAN J, STOEV S. Network-wide statistical modeling, prediction, and monitoring of computer traffic[J]. Technometrics, 2013, 55(1): 79-93.

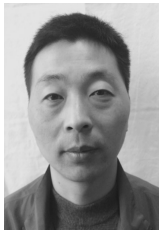
[19] OGASAWARA S, TAKAHASHI Y. Performance analysis of traffic classification in an OpenFlow switch[C]//2016 Cloudification of the Internet of Things (CIoT). 2016: 1-6.

[20] GARCÍA-MARTÍNEZ C, LOZANO M, RODRÍGUEZ-DÍAZ F J. A simulated annealing method based on a specialised evolutionary algorithm[J]. Applied Soft Computing Journal, 2012, 12(12): 573-588.

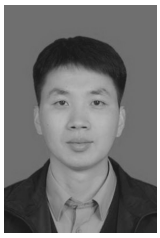
[作者简介]

胡涛（1993-），男，陕西武功人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为宽带信息网、软件定义网络。

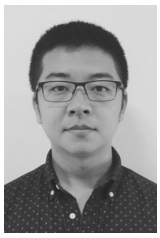
张建辉（1977-），男，河南平顶山人，国家数字交换系统工程技术研究中心副研究员，主要研究方向为宽带信息网、网络安全。



孔维功（1980-），男，河南封丘人，国家数字交换系统工程技术研究中心博士生，主要研究方向为宽带信息网。



杨森（1985-），男，辽宁盖州人，国家数字交换系统工程技术研究中心助理研究员，主要研究方向为通信与信息网络。



曹路佳（1983-），男，河北抚宁人，国家数字交换系统工程技术研究中心助教，主要研究方向为网络安全。