基于联盟博弈的自适应 SDN 交换机迁移机制

姚蓝，兰巨龙
（国家数字交换系统工程技术研究中心，河南 郑州 450001）

摘 要：针对软件定义网络（SDN）中不合理的控制器和交换机映射关系导致的控制平面性能低的问题，提出基于联盟博弈的自适应交换机迁移机制。首先，综合考虑控制器资源利用率、控制开销和流建立时间，将交换机迁移问题建模为组合优化问题；然后引入博弈论设计分布式算法，每个控制器独立运行控制逻辑，通过控制器之间协调合作、联盟博弈，实现了依据流量特征的自适应交换机迁移。仿真结果表明，所提机制能更好地适应流量特征，减少约 19%的控制流量开销和 30%的平均流建立时间，提高了控制器资源利用率。

关键词：软件定义网络；交换机迁移；联盟博弈；资源利用率
中图分类号：TP393.2
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2020161

Adaptive SDN switch migration mechanism based on coalitional game

YAO Lan, LAN Julong
National Digital Switching System Engineering & Research Center, Zhengzhou 450001, China

Abstract: The problem of poor control plane performance causes in software-defined Networking due to the unreasonable mapping relationship between controllers and switches. To address this issue, an adaptive switch migration mechanism based on coalitional game was proposed. First, comprehensively considering the controller resource utilization, control overhead, and flow establishment time, the switch migration problem was modeled as a combination optimization problem. Then, a game theory was introduced to design a distributed algorithm, where each controller ran control logic independently and implemented coalitional game between controllers to achieve an adaptive switch migration mechanism that adapted to traffic characteristics. The simulation results show that the proposed mechanism can better adapt to the flow characteristics, reduce the control traffic overhead by about 19% and the average flow settling time by 30%, and improve the controller resource utilization.

Key words: software-defined networking, switch migration, coalitional game, resource utilization

1 引言

软件定义网络（SDN, software-defined networking）通过控制与转发分离，能够为灵活的网络管理和敏捷的服务创新提供有力的支持，目前已经在企业网、数据中心网络等得到推广。SDN 使用逻辑集中的控制平面来收集网络视图信息、处理流请求、计算网络策略[1]。但集中式的控制平面面临严重的可扩展性问题，这一可扩展性问题主要指网络中流量规模的增长带来的控制器超负载问题，会严重影响控制器的性能。

为了提高可扩展性并避免单点故障，SDN 常采用分布式控制平面[2]，部署多控制器对网络进行分

收稿日期：2020−04−14；**修回日期**：2020−06−30
基金项目：国家自然科学基金资助项目（No.61521003, No.61872382）；国家重点研发计划基金资助项目（No.2017YFB0803204）；广东省重点邻域研发计划基金资助项目（No.2018B010113001）
Foundation Items: The National Natural Science Foundation of China (No.61521003, No.61872382), The National Key Research and Development Program of China (No.2017YFB0803204), The Research and Development Program in Key Areas of Guangdong Province (No.2018B010113001)

域管理，将交换机静态映射到一个或多个控制器上。但是，交换机和控制器之间的静态映射易导致控制器响应时间长且变化很大，这是因为网络中的流量频繁波动。在空间上，总流量通常在白天达到高峰，而在晚上流量较少，此外，即使总流量保持不变，流量在短时间内也具有突发性[3]。流量的不确定性会导致控制器之间负载不均衡，从而导致一些过载控制器的交换机响应时间过长，而且控制器长时间超负载工作易发生故障，导致网络瘫痪。因此，设计动态的控制器与交换机映射机制，以缩短控制器响应时间并更好地利用控制器资源，对提高 SDN 控制平面的可扩展性和可靠性具有重要的意义。Dixit 等[4]提出通过动态添加或减少控制器的数量来拓展控制平面，但如何设计适应流量特征的控制器和交换机映射关系仍然是亟需解决的问题。从交换机的角度来看，它更容易映射到响应时间短的控制器来提高性能；从控制器的角度来看，它管理较近的交换机所产生的控制流量开销较小。综上，设计动态的控制器和交换机的映射目标应满足以下 3 个指标：控制器上的负载是均匀的，实现最大利用率，减少流处理时间。

为了解决这些问题，本文提出一种基于联盟博弈的交换机迁移机制（CGSM, coalitional game-based switch migration），综合考虑控制器资源利用率、控制开销和流建立时间，设计可以动态调整控制器的状态和控制器–交换机映射关系的自适应交换机迁移机制。本文的主要贡献如下。

1) 针对 SDN 中控制器–交换机不合理的映射关系导致控制平面性能低下的问题，提出适应流量特征的自适应交换机迁移机制，综合考虑控制器资源利用率、控制开销和流建立时间，将交换机迁移问题建模为组合优化问题。

2) 设计了基于联盟博弈的交换机迁移机制，把博弈论引入控制器运行逻辑当中，设计分布式算法，每个控制器独立运行算法，通过控制器之间联合博弈、协调合作，实现自适应的控制器状态调整和弹性的控制器–交换机映射关系。

3) 开展相关实验对所提交换机迁移机制的性能进行验证，仿真结果表明，本文所提机制能减少约 19%的控制开销和 30%的平均流建立时间，提高了控制平面的性能。

2 相关研究

为了提高 SDN 控制平面的可扩展性[5]，许多研究人员对 SDN 控制器和交换机的动态映射机制展开了深入的研究。SDN 分布式控制平面是提高控制器可扩展性的有效方法[6]。在这样的架构中，控制平面可以由多个控制器组成。多个控制器负责管理网络的不同管理域，并与相邻域交换本地信息以增强全局策略的实施，如 Hyperflow[7]、Onix[8]、ONOS[9]。Zhang 等[10]提出了基于交换机迁移的多域 SDN 中的分发策略，通过优化数据收集、交换机迁移和控制器状态同步 3 个因素来确定目标控制器和迁移的交换机，实现了更好的控制器负载平衡。

胡涛等[11]将控制器关联问题建模为双向匹配问题，从控制器和交换机的角度出发优化控制器–交换机映射问题，仿真表明，该方案能有效降低流请求排队时延，并且实现了控制器之间更好的负载平衡。Xiao 等[12]提出了用于 SDN 控制平面的交换机迁移决策方案，设计迁移收益模型衡量迁移代价来解决迁移决策问题，以减少控制命令传输时延和平衡控制器负载，仿真表明，该方案可以将控制信令的平均传输时延减少 17%，将处理时延减少 22.7%。

Xu 等[13]提出了用于平衡控制器负载的 SDN 交换机迁移方案，在保障迁移成本较低的前提下，实现 SDN 控制器之间的负载平衡。该迁移方案不受交换机迁移中断的影响，并且不会引起任何服务中断。仿真表明，该机制通过仅迁移少量具有低开销的交换机，就可以大大减少 SDN 控制器之间的负载不平衡。Mykola 等[14]考虑流量的优先级和用户服务质量体验来实施交换机迁移策略，实验结果表明，考虑流量的优先级特征和用户的需求，能更充分地利用控制平面资源。Bari 等[15]提出的交换机迁移机制动态地改变了控制器的数量及其在不同条件下的位置，以实现负载平衡，但该机制没有明确控制器之间的通信与信息同步机制。

Zhou 等[16]将交换机迁移解释为签名匹配问题，并将控制器和交换机映射问题建模为 EMD（earth mover’s distance）模型，通过减少流量差异以保护网络中重要节点的控制器，并提出了一种高效的启发式方法来求解大规模网络交换机迁移问题。Wang 等[17]为了提高交换机迁移效率，设计了迁移效率感知的贪婪算法实施交换机迁移，以权衡迁移成本和负载均衡率。Grkemli 等[18]引入一种新颖的动态控制平面体系结构，根据特定的控制器负载和控制器资源利用率或数据流服务类型，在多个控制器实例

之间分配不同的控制流。Li 等[19]分析了不同特征流请求的资源消耗，设计了一种基于数据流特征的动态控制器关联机制，通过基于联盟博弈的最小集合覆盖算法减少了控制平面处理数据流请求的资源消耗。Cheng 等[20]将交换机迁移问题建模为一个集中的资源利用最大化问题，结合控制器 CPU、带宽和内存的资源利用率，通过非合作博弈论方法设计分布式算法，提高了 SDN 的可扩展性。Filali 等[21]提出了一种交换机迁移调度算法，使用多步自回归移动平均模型（ARIMA, autoregressive integrated moving average model）来预测控制器的长期负载水平，提前开启交换机迁移操作以适应流量需求，在确保控制器之间的负载平衡的同时，提高了迁移效率。

现有针对弹性控制问题的方法主要是通过交换机和控制器之间的重新布局来改变控制器的数量和位置，该策略会造成大量的交换机迁移，使网络不稳定。现有解决控制器与交换机弹性映射的方法多采用集中式算法，以全局优化的思想来达到最优，随着网络规模的不断增大，这些算法的计算量很大，严重影响了控制平面的扩展性。

3　研究动机

控制器和交换机之间的动态映射可以更好地适应网络流量、提高控制器资源利用率。网络中流量具有突发性，且高峰流量和低谷流量差距很大，实际网络中有较多交换机迁移的场景。OpenFlow 1.3 中设置了不同的控制器角色（主控制器和从控制器），通过改变控制器和交换机的主从关系，控制器和交换机可以多对多映射，实现控制器和交换机之间灵活的映射关系。

首先，当网络中有大部分控制器均处于轻载状态时，直观的想法是将轻载控制器的负载进行合并，并将空闲的控制器关闭或者休眠以节省电力和通信成本。图 1(a)中，控制器 c_1 和 c_2 均处于轻载状态，把控制器 c_2 的所有负载交换机均迁移到控制器 c_1 上，此时便可以将控制器 c_2 休眠，以节省能量。

当网络中大部分处于工作状态的控制器均已过载时，为了保障控制平面的服务性能，应扩展控制器资源，触发休眠控制器，并将过载控制器的交换机迁移到新的控制器上。图 1(b)中，当控制器 c_1 过载时，为了防止控制器 c_1 因过载而影响性能或者损坏，甚至导致网络崩溃，应触发休眠控制器 c_2 以分担控制器 c_1 的负载。

在 SDN 中，多控制器中往往会出现一些控制器过载，而另一些控制器却处于轻载的状态，但不需要扩展或缩减控制器资源，此时需要实施负载均衡策略以平衡控制器之间的负载，提高控制平面的处理性能。图 1(c)中，当控制器之间负载不平衡时，将部分交换机从过载控制器迁移到轻载控制器，便可以提升控制平面数据流请求的处理效率。当然，在扩展控制器资源和缩减控制器资源之后，也应进行交换机迁移以实施负载均衡策略。

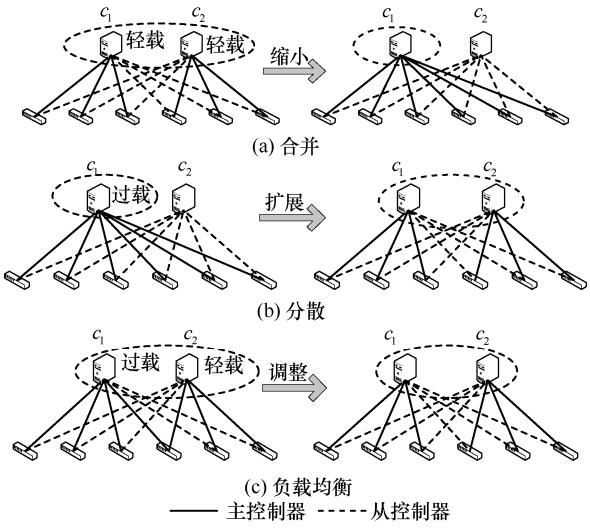


图 1　交换机迁移场景

因此，为了更好地利用控制器资源，控制器和交换机之间的映射应适应网络流量特征，使控制平面可以处理更多的数据流，提升网络性能。本文旨在设计自适应的交换机迁移机制以实现控制器–交换机动态可扩展的弹性映射关系，它可以触发新的控制器或将过载控制器的交换机迁移到轻载控制器上以应对网络中突发的流量，当网络中流量处于低谷时，它可以休眠部分控制器以节省能量。

4　模型构建

本节首先基于图论的知识，对网络模型和控制器负载、控制器处理时延进行描述，然后建立优化目标函数，设计分布式算法，以达到负载均衡的目的。

4.1　网络建模

给定一个多控制器域 SDN 模型，结合图论的相关知识，对 SDN 建模。整个网络拓扑可用无向图 $G=(V,E)$ 表示，其中 V 表示网络节点（即交换

机和控制器所在位置），E 表示节点间链路集合。网络中共有 m 个控制器和 n 个交换机，定义 SDN 控制平面中控制器集合为 $C=\{c_1,c_2,\cdots,c_m\}$，各个控制器处理容量为 $\alpha=\{\alpha_1,\alpha_2,\cdots,\alpha_m\}$，交换机集合为 $S=\{s_1,s_2,\cdots,s_N\}$。设 d_{ij} 表示交换机 i 和控制器 j 之间的跳数，控制器与交换机采用带内通信，所有节点之间以最短距离通信。交换机 i 的流请求速率为 λ_i，控制器与交换机之间的映射关系为 x_{ij}，如式(1)所示。

$$x_{ij}(t)=\begin{cases}1, & \text{交换机 } i \text{ 和控制器 } j \text{ 相连}\\ 0, & \text{其他}\end{cases} \tag{1}$$

定义 1 流建立时间。由于当今的网络能提供较大的带宽，因此数据分组的传播时延（以 μs 为单位）比控制器 CPU 的处理时间（以 ms 为单位）[7]要小。因此，本文仅在控制器上对流请求处理时间进行建模。假设在时隙 t 中第 i 个交换机的流请求数量用 $\lambda_i(t)$ 表示，流请求到达时间遵循泊松过程，其中 $\lambda_i(t)\leqslant a_j$。因此，第 j 个控制器的负载可以表示为

$$\theta_j(t)=\sum_{i=1}^{N}\lambda_i(t)x_{ij}(t) \tag{2}$$

本文做出以下假设：$\lambda_i(t)$ 是相互独立的，控制器队列调度遵循 M/M/1 队列。则第 j 个控制器平均数据流建立时间为

$$\tau_j(t)=\frac{1}{\alpha_j-\theta_j(t)} \tag{3}$$

整个控制平面的所有控制器平均流建立时间为

$$\tau(t)=\frac{\sum_{j=1}^{m}\theta_j(t)\tau_j(t)}{\sum_{j=1}^{m}\theta_j(t)} \tag{4}$$

定义 2 控制开销 η。出于 SDN 的可扩展性考虑，本文控制器与交换机之间的信息交流均采用带内通信，因此，控制流量会占用数据信道的带宽，减少控制流量开销，能提高数据平面的转发性能。定义控制器开销与控制器负载和控制机与交换机之间的跳数有关，如式(5)和式(6)所示。

$$\eta(t)=\sum_{j=1}^{m}\sum_{i=1}^{n}d_{ij}x_{ij}(t)\lambda_i(t) \tag{5}$$

$$\eta_j(t)=\sum_{i=1}^{n}d_{ij}x_{ij}(t)\lambda_i(t) \tag{6}$$

其中，η 表示所有控制器的总开销，η_j 表示控制器 j 产生的总的控制开销。

定义 3 控制器资源利用率 γ_j。控制器资源利用率为控制器负载与控制器最大容量的比值，这是判断控制器是否过载的衡量指标。当控制器资源利用率大于 0.9 时，控制器处于过载状态；当控制器资源利用率小于 0.1 时，控制器处于轻载状态。

$$\gamma_j=\frac{\theta_j(t)}{\alpha_j} \tag{7}$$

4.2 优化目标函数

综合上述流建立时间、控制开销、控制器资源利用率，为了设计自适应交换机迁移机制，出于节能考虑，设置控制器资源利用率阈值上限为 0.9，阈值下限为0.1，即当控制器资源利用率超过0.9时，控制器处于过载状态，需要迁出交换机以卸载；当控制器资源利用率低于 0.1 时，出于节能，需要迁出交换机以休眠控制器；当控制器资源利用率在0.1～0.9 时，实施负载均衡策略，以最小化基于流建立时间和控制开销的加权效益。控制器会时刻监测自己的资源利用率情况，并相互通告以决定是否实施交换机迁移策略 Γ，如式(8)所示。

$$\Gamma=\begin{cases}\text{卸载}, & \gamma>0.9\\ \text{负载均衡}, & 0.1\leqslant\gamma\leqslant0.9\\ \text{休眠控制器}, & \gamma<0.1\end{cases} \tag{8}$$

控制器资源利用率为 0.1～0.9 的优化目标函数设为

$$\min\quad \delta\tau(t)+(1-\delta)\eta(t) \tag{9}$$

约束条件为

$$\forall i,j\quad x_{ij}\in\{0,1\} \tag{10}$$

$$\sum_{j=1}^{m}x_{ij}=1,\ \forall i \tag{11}$$

$$\theta_j(t)\leqslant\beta\alpha_j,\ \forall j \tag{12}$$

目标函数式(9)为最小化基于流建立时间和控制开销的加权和。式(10)和式(11)表示任何时刻每一个交换机能且只能映射到一个控制器上。式(12)表示每一个控制器的负载不得超过负载上限。

5 算法设计

现有解决控制器与交换机弹性映射的方法都是通过将控制器和交换机映射问题转化为 NP 难的

整数线性规划问题，设计启发式算法求解，但这些算法多采用集中式算法，且本质上是一种重映射算法，随着网络规模的不断增大，这些算法具有较高的复杂度，严重影响了控制平面的扩展性。

在大规模网络当中，SDN 常采用分布式控制平面，每个控制器仅管理域内的交换机，通过与邻居控制器交互来处理全局事件。传统的采用集中式的算法往往不适应 SDN 控制平面分布式的网络架构，会带来大量的同步和通信开销。因此，本文设计一种分布式算法以适应 SDN 分布式的控制平面，每个控制器独立运行算法，通过与邻居控制器交互来实现交换机的动态迁移，分布式算法将计算任务分担到每个控制器上，便于增加或休眠控制器，更适用于大规模网络，能增加网络的可靠性。

本文遵循分布式准则设计联盟博弈模型，通过控制器之间协调合作、相互博弈，来实施交换机迁移策略。博弈论是指研究多个个体或团队之间在特定条件制约下的对局中，利用相关方的策略来实施对应策略的学科。博弈论考虑博弈中个体的预测行为和实际行为，并研究它们的优化策略。在控制器–交换机映射问题中，控制器会偏好连接时延较小且产生的控制开销较小的交换机，而不同的交换机连接到不同的控制器会导致不同的时延和控制开销。控制器之间为了追求最大化自身性能效益会形成竞争关系，此时交换机便是控制器之间竞争的商品，通过控制器交互博弈来寻求自身性能的提升。这种竞争并非完全对立的竞争，而是通过彼此合作博弈，使最终整个控制平面的性能达到最优。合作博弈采用的是一种合作的关系，其本质的思想是联盟和分配，每个参与者从联盟中得到的收益不少于单独经营所获得的收益，合作博弈能增进整体的利益。在控制器和交换机的映射问题中，参与博弈的对象是各个控制器，博弈的商品是交换机，控制器之间通过合作博弈、改变交换机的映射关系来改进自身性能。合作博弈的核心思想是参与人如何结盟以及如何重新分配结盟的支付，当控制器自身的性能较差时，会通告其邻居控制器，然后和邻居控制器形成联盟，对交换机进行重新分配。控制器之间通过博弈获取个体自身性能的提升，在整个控制平面中，控制器通过不断形成联盟博弈，优化交换机的从属关系，最终控制器和交换机映射问题收敛到纳什稳定状态，实现了整个控制平面的全局优化。

为了设计合适的控制器联盟博弈模型，在控制器之间实施联盟博弈策略，首先对决策域、收益函数、控制器通信机制进行讨论。

决策域是为了达到迁移交换机目的而形成的决策区域。当网络中控制器性能较差时，它会通告邻居控制器其负载信息，并发出交换机迁移请求，同时，邻居控制器会选择接受或拒绝请求，接受请求的邻居控制器会建立决策域，并做出迁移策略。在网络中，多个决策域可以同时进行，决策域之间不存在重叠。为了构建决策域，本文引入控制器 c_j 的邻居集的概念。c_i 与 c_j 形成邻居集必须满足以下条件：c_j 的控制域至少有一个交换机与 c_i 中交换机相连，即 c_i 和 c_j 的控制域相连；交换机迁移只发生在 2 个相邻的控制器之间。如图 2 所示，控制器 c_1 的邻居控制器包括 c_2、c_3、c_4，由于控制器 c_1 过载，会向邻居控制器请求建立迁移决策域，由于控制器 c_3 过载，选择拒绝参与联盟决策，因此，控制器 c_1、c_2、 c_4 会建立决策域 $\Omega=\{c_1,c_2,c_4\}$。

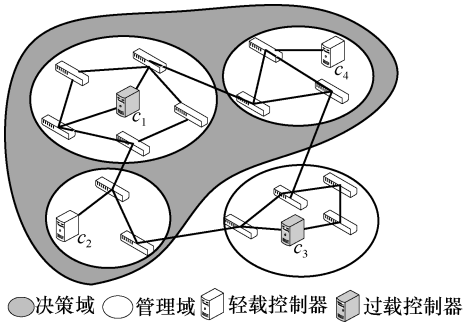


图 2　决策域

为了确定收益函数，由于优化式(9)是全局化目标函数，也是控制器实施交换机迁移的目的，为了设计分布式算法，需要对式(9)进行改进，定义控制器之间博弈的收益函数为

$$f_x=(1-\delta)(\tau_{\text{th}}-\tau_x)+\delta(\eta_{\text{th}}-\eta_x) \tag{13}$$

其中，τ_x 和 η_x 分别是当前控制器的平均流建立时间和控制开销。通过式(13)所示的收益函数，控制器之间通过合作、博弈，以获得更小的流建立时间和控制开销。每个控制器计算其收益函数并独立做出决策。随着网络负载的变化，活动控制器的利用率和时延也发生变化，从而获得的收益也不同。

对于给定的网络拓扑，控制器会向其邻居控制器通告交换机迁移请求，构建决策域并进行迁移交换机行为。多个决策域可以同时进行但并不重叠，

通过控制器之间的相互交流，最终达到全网控制器负载均衡的目的。在决策域中，多控制器做出迁移策略的行为，建模为 n 人合作博弈模型，交换机为商品，控制器为博弈参与者，通过各控制器的竞争博弈，来决定将要迁移的交换机和目标控制器，整个过程可分为 3 个阶段。

阶段 1 在 SDN 中，每个控制器实时计算自身和邻域控制器的状态，当某一控制器过载时，会通告邻域控制器交换机迁移请求，邻域控制器根据自身状态，选择接受或拒绝请求，选择接受的控制器会建立决策域。

阶段 2 在决策域中，迁出控制器会随机选择交换机，参与博弈的控制器会计算接受交换机所付出的收益函数。

阶段 3 若决策域中存在控制器接受交换机迁移，则启动交换机迁移机制，通告其他控制器交换机迁移活动。待交换机迁移完成后，控制器集合会更新自身资源利用率和平均流建立时间，返回阶段 1。

控制器之间通过联盟博弈来实施交换机的迁移，交换机迁移由单个控制器独立运行算法，多个控制器协同实现，这里需要对 OpenFlow 协议进行扩展，使之支持多控制器协同机制，控制器之间的通信过程如图 3 所示。当控制器 c_k 向邻居控制器发出请求时，邻居控制器 c_n、c_m 回复迁移请求，选择是否建立联盟决策域，当控制器 c_k、c_n、c_m 形成联盟决策域时，会计算收益函数，选择要迁移的交换机以及迁移的目标控制器，然后启动交换机迁移，通告其他控制器正在进行交换机迁移活动，待迁移结束后，控制器 c_k、c_n、c_m 更新自身资源利用率、控制开销以及流建立时间，并向其邻居广播更新。

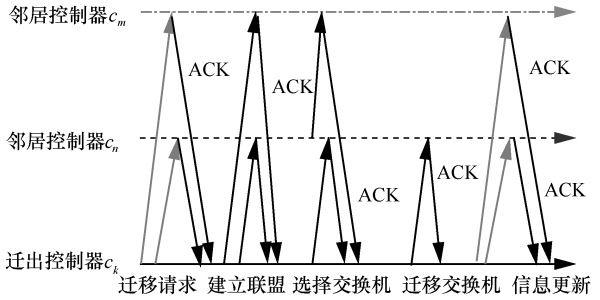


图 3 控制器之间的通信过程

算法 1 为基于联盟博弈的自适应交换机迁移机制。通过控制器博弈，每个控制器试图最小化流建立时间和控制开销。在此过程中，基于收益函数，一个或多个邻居控制器可以实施负载均衡，共同分担过载控制器的流请求（步骤 12)～步骤 19)）。当网络负载较低时，轻载控制器触发控制器休眠消息（步骤 6)），将其休眠以节约电力，如算法 2 所示。类似地，随着负载的增加，过载控制器可以卸载交换机到邻居控制器或触发控制器启动消息（步骤 8)），如算法 3 所示。待交换机迁移完毕，SDN 控制器互相通信共享网络状态信息，例如收益矩阵、资源利用率、流建立时间和控制开销（步骤 17)和步骤 18)）。

算法 1 基于联盟博弈的自适应交换机迁移机制

输入 控制器–交换机之间的初始关系 $X=[x_{ij}]$，控制器资源利用率阈值 $\gamma_{\max}$ 和 $\gamma_{\min}$，控制器流建立时间阈值 $\tau_{\max}$

输出 控制器–交换机之间新的映射关系 $X'=[x'_{ij}]$

1) 初始化 $X=[x_{ij}]$
2) repeat
3) for each C_i //对于每个控制器
4) 计算每个控制器 C_i 资源利用率 γ_j
5) if $\gamma_x<0.1$
6) （X'，Y）← 算法2
7) if $\gamma_x>0.9$
8) （X'，Y）← 算法3
9) end if
10) end if
11) end for
12) for each c_m //对于每一个控制器
13) $f_x=(1-\delta)(\tau_{\text{th}}-\tau_x)+\delta(\gamma_{\text{th}}-\gamma_x)$ //计算当前收益
14) if $s_i: c_m \to C_n$ and $\theta_n \leqslant \alpha\beta$ //保证控制器不过载，构建决策域
15) $f'_x=(1-\delta)(\tau_{\text{th}}-\tau'_x)+\delta(\gamma_{\text{th}}-\gamma'_x)$ //若迁移，计算当前收益
16) if $f'_x > f_x$ //若交换机迁移前后，当前收益增大
17) $s_i \to c_n$ //实施交换机迁移
18) $f_x=f'_x$ //更新新的收益
19) $\gamma_x=\gamma'_x$ //更新资源利用率
20) end if
21) end if

22)　end for

23) 系统没有任何交换机要求迁移，算法收敛

当控制平面资源利用率小于 0.1 时，轻载控制器会运行算法 2 以触发相关控制器休眠，节省控制资源。此时，网络流量较少，这说明控制器足以处理这些数据流请求，出于节能的考虑，控制平面会休眠一些控制器，以节省控制资源。由于一些控制器的休眠，其上的交换机要迁移到其他控制器上，因此，会造成其他控制器的资源利用率和收益产生变化。因此，算法 2 的核心思想是选择合适的控制器进行休眠（步骤 6)），同时将其上的交换机迁移到其他的控制器上，并更新网络资源状态，包括控制器–交换机映射关系、每个控制器新的资源利用率和产生的当前收益（步骤 9)～步骤 11)），一旦获得迁移所有交换机的解决方案，轻载控制器就会触发自我休眠（步骤 12)），然后返回算　法 1 执行负载均衡算法，以最小化控制开销和流建立时间。

算法 2　资源利用率较低场景下的交换机迁移

输入　控制器–交换机之间的原始初始关系 $X=[x_{ij}]$，控制器资源利用率阈值 $\gamma_{\max}$ 和 $\gamma_{\min}$，控制器流建立时间阈值 $\tau_{\max}$

输出　控制器–交换机之间新的映射关系 $X'=[x'_{ij}]$

1) 初始化 $X=[x_{ij}]$

2) repeat

3)　for　each　C_i //对于每个控制器

4)　　计算每个控制器 C_i 资源利用率 γ_j

5)　if　$\gamma_x<0.1$

6)　　select　min_payoff(C_m)//选择当前收益最小的控制器以休眠

7)　　if　$s_i: c_m \to C_n$　and　$\theta_n \leqslant \alpha\beta$ //构建决策域迁出所有交换机

8)　　$f'_x=(1-\delta)(\tau_{\text{th}}-\tau'_x)-\delta(\gamma_{\text{th}}-\gamma'_x)$ //若迁移，计算对其邻居控制器产生的收益

9)　　$s_i \to c_n$ //实施交换机迁移

10)　　$f_x=f'_x$ //更新新的收益

11)　　$\gamma_x=\gamma'_x$ //更新资源利用率

12)　　off controller　C_m　//休眠控制器

13)　　back　算法 1 //执行负载均衡算法

14)　end if

15)　end if

16)　end for

当控制平面资源利用率大于 0.9 时，过载控制器应启动算法 3，确定其所管理的一组交换机，将这些交换机卸载到相邻的控制器上。在此过程中，控制器执行 2 个过程：触发新控制器，将过载控制器管理的交换机卸载到新的控制器（步骤 6)～步骤 12)）。首先，该算法选择距离过载控制器较近的休眠控制器启动，并优先选择过载控制器中流请求较大的交换机进行迁移（步骤 7)～步骤 9)），为了检查最佳的交换机迁移方案，它会计算收益函数，以衡量卸载交换机对其邻居控制器的收益可能产生的影响。然后，控制器请求卸载交换机到影响最小的控制器。最后，返回算法 1 执行负载均衡算法，以最小化控制开销和流建立时间。

算法 3　资源利用率较高场景下的交换机迁移

输入　控制器–交换机之间的原始初始关系 $X=[x_{ij}]$，控制器资源利用率阈值 $\gamma_{\max}$ 和 $\gamma_{\min}$，控制器流建立时间阈值 $\tau_{\max}$

输出　控制器–交换机之间新的映射关系 $X'=[x'_{ij}]$

1) 初始化 $X=[x_{ij}]$

2) repeat

3)　for each　C_i //对于每个控制器

4)　　计算每个控制器 C_i 资源利用率 γ_j

5)　if　$\gamma_x>0.9$

6)　　select　distance_priority(C_n)//选择距离较近的休眠控制器启动

7)　　on controller　C_n //启动交换机

8)　　select　r_max(C_m)//选择利用率较大的控制器

9)　　select　flow_priority(s_i)//选择流请求较大的交换机

10)　　if　$s_i: c_m \to C_n$　and　$\theta_n \leqslant \alpha\beta$ //构建决策域迁出所有交换机

11)　　$f'_x=(1-\delta)(\tau_{\text{th}}-\tau'_x)-\delta(\gamma_{\text{th}}-\gamma'_x)$ //若迁移，计算对其邻居控制器产生的收益

12)　　$s_i \to c_n$ //实施交换机迁移

13)　　$f_x=f'_x$ //更新收益

14)　　$\gamma_x=\gamma'_x$ //更新资源利用率

15)　　back　算法 1//执行负载均衡算法

16)　end if

17)　end if

18)　end for

6 仿真结果分析

6.1 仿真环境设置

本节对所提机制的性能进行仿真验证，实验环境设置如下。

本文选择 RYU[22]作为实验控制器，并使用 Mininet[23]作为测试平台。考虑到 RYU 和 Mininet 之间的性能干扰，本文将 Mininet 和 RYU 分别安装在不同的物理设备上。本文配备 6 台具有相同实验配置的机器，英特尔酷睿 i7@、3.6 GHz，4 GB RAM，2 Gbit/s 网卡和 Ubuntu14.04 LTS。其中 5 台机器运行 RYU 控制器，另一台则运行 Mininet，Iperf 用于在主机之间生成流量。

仿真采用拓扑 Interllifiber，取自 the Internet topology zoo[24]，其中包含 73 个节点、93 条链路。实验中每个交换机接入 2 个主机。为了模拟动态的网络流量，Iperf 生成动态流量，数据流在主机上的生成和消失服从泊松分布。

6.2 仿真结果分析

将本文所提 CGSM 机制与控制器–交换机静态映射（SSC, static switch-controller）机制、基于贪心背包算法的控制器–交换机映射（DCP-GK, dynamic controller provisioning using greedy knapsack）机制和控制器利用率感知的交换机迁移（UMS, utilization-aware migrating switch）机制进行对比。其中，SSC[25]为控制器部署按照交换机的距离进行聚类分析，并划分管理域，一旦控制器部署完成便不会改变控制器–交换机关联关系；DCP-GK[15]基于贪心背包算法进行交换机迁移；UMS[26]优先将交换机迁移到利用率最低控制器上。

6.2.1 实验 1

实验 1 对 4 种方案的控制器资源利用率进行对比，如图 4 所示。从图 4 中可以看出，随着流请求速率的增大，4 种方案的控制器资源利用率也在增大；在不同的流请求速率下，本文所提 CGSM 机制的控制器资源利用率均高于其他方案。CGSM 能适应网络流量，动态地增加或减少控制器资源，保证控制平面资源利用率在 0.1～0.9，实现了控制器资源的合理利用。

6.2.2 实验 2

图 5 和图 6 分别显示了 4 种方案的控制开销和流建立时间。从图 5 和图 6 中可以看出，UMS 优先将交换机迁移到利用率较低的控制器，算法单一，控制开销相对较低，但由于迁移的控制器距离较远，导致流建立时间偏大；DCP-GK 基于贪心算法选择迁移策略，难以实现最优控制；CGSM 通过控制器之间相互合作、联合博弈以最大化控制开销和流建立时间的加权和，与 DCP-GK 相比，CGSM 能减少约 19%的控制开销和 30%的平均流建立时间。

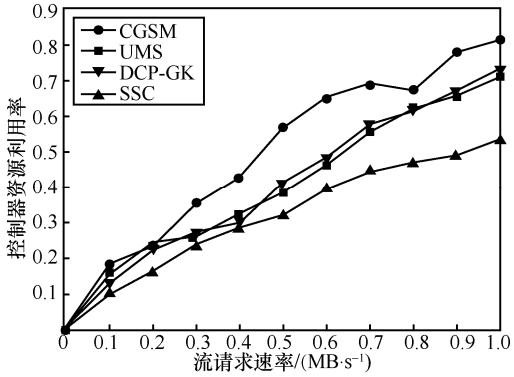


图 4 4 种方案的控制器资源利用率

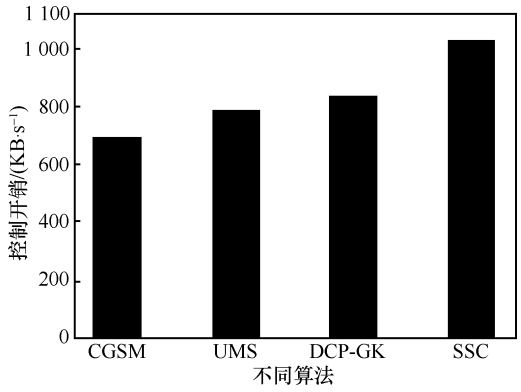


图 5 4 种方案的控制开销

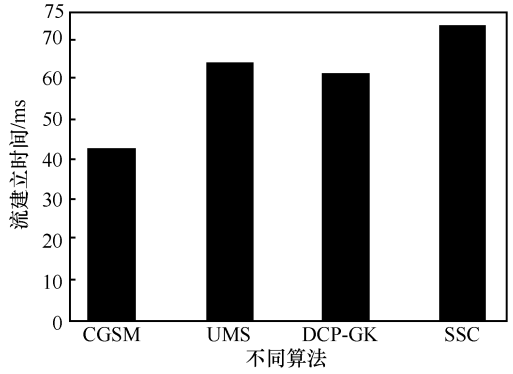


图 6 4 种方案的流建立时间

6.2.3 实验 3

为了验证 CGSM 机制的性能，实验 3 从 the Internet topology zoo 网络拓扑中选取 4 个规模差异

较大的网络拓扑进行仿真实验，网络拓扑及参数设置如表 1 所示。

表 1　　网络拓扑参数

网络拓扑	节点数/个	链路数/条	控制器数/个
OS3E	34	42	4
Germany50	50	88	5
Interllifiber	73	93	6
Interoute	110	149	7

图 7 和图 8 显示了不同网络拓扑下 4 种方案的归一化的控制开销和流建立时间。从图 7 和图 8 中可以看出，由于 UMS 交换机迁移选择僵化和 DCP-GK 基于贪心算法迁移交换机，从本质上都是集中式算法，且都是贪婪算法，难以实现全局的最优控制。与 UMS 和 DCP-GK 相比，CGSM 采用分布式算法，通过控制器联合博弈，最大化自身收益，实现了控制器–交换机的优化映射，降低了网络中通信开销，减少了流建立时间。

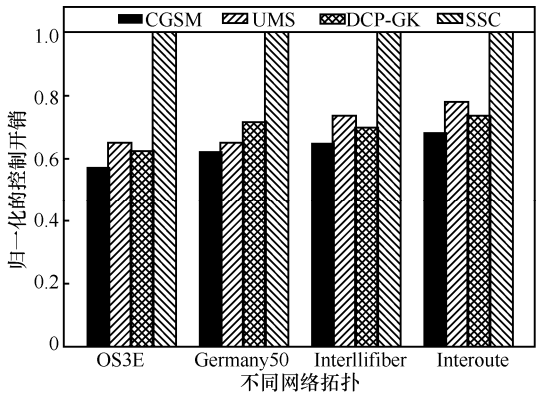


图 7　不同网络拓扑下 4 种方案的控制开销

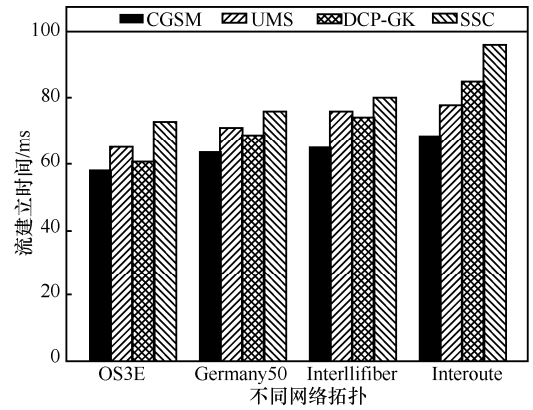


图 8　不同网络拓扑下 4 种方案的流建立时间

图 9 显示了不同网络拓扑下 4 种方案的控制器负载均衡率。从图 9 可以看出，较其他算法，CGSM 的控制器负载更均衡，SSC 由于采用静态部署，不适应网络流量变化，其负载均衡性能最差。UMS 和 DCP-GK 本质上都是贪婪算法，难以实现全局优化，负载均衡性能有限。CGUM 可以动态调整控制器状态，保障控制器资源的动态供应，以及控制器和交换机动态的映射关系，能更好地适应流量特征，实现了更合理的交换机迁移机制和较好的负载均衡率。

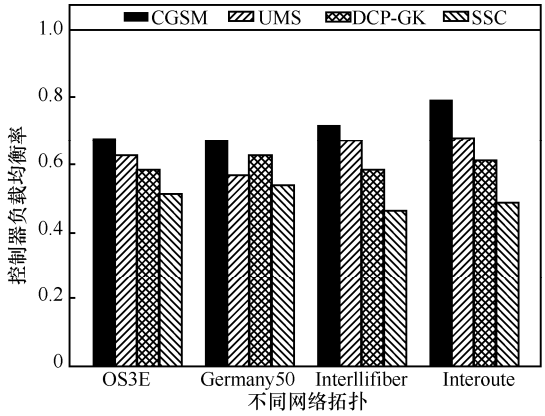


图 9　不同网络拓扑下 4 种方案的负载均衡率

7　结束语

本文对 SDN 控制器和交换机映射不合理导致控制平面性能低下的问题，提出了一种基于联盟博弈的自适应交换机迁移机制——CGSM。CGSM 采用分布式算法，将控制器和交换机映射问题建模为组合优化问题，并设计了自适应交换机迁移机制，通过控制器协调合作、联合博弈以最大化自身利益，实现了全局优化的控制器–交换机映射方案。仿真结果表明，相比现有的交换机迁移算法，该机制同时减小约 19%的控制开销和 30%的平均流建立时间，获得了较高的控制器资源利用率和良好的负载均衡。

参考文献：

[1] NUNES B A A, MENDONCA M, NGUYEN X N, et al. A survey of software-defined networking: past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634.

[2] ZHANG Y, CUI L, WANG W, et al. A survey on software defined networking with multiple controllers[J]. Journal of Network and Computer Applications, 2017, 103: 101-118.

[3] 胡涛, 张建辉, 孔维功, 等. SDN 中基于过程优化的交换机竞争迁移算法[J]. 通信学报, 2017, 38(8): 213-222.
HU T, ZHANG J H, KONG W G, et al. Switch competing migration

algorithm based on process optimization in SDN[J]. Journal on Communications, 2017, 38(8): 213-222.

[4] DIXIT A, HAO F, MUKHERJEE S, et al. Towards an elastic distributed SDN controller[J]. Computer Communication Review, 2013, 43(4): 7-12.

[5] KARAKUS M, DURRESI A. A survey: control plane scalability issues and approaches in software-defined networking (SDN)[J]. Computer Networks, 2017, 112(15): 279-293.

[6] ZHOU N, HU T, HU Y X, et al. An adaptive switch migration strategy for balancing loads in software-defined networking[J]. Application of Electronic Technique, 2019, 12: 91-95.

[7] TOOTOONCHIAN A, GANJALI Y. HyperFlow: a distributed control plane for OpenFlow[C]//Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. Berkeley: USENIX Association, 2010: 1-14.

[8] TEEMU K, MARTIN C, NATASHA G, et al. Onix: a distributed control platform for large-scale production networks[C]// Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2010: 1-15.

[9] BERDE P, HART J. ONOS: towards an open, distributed SDN OS[C]//ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'14). New York: ACM Press, 2014: 1-6.

[10] ZHANG J H, HU T, ZHAO W. DDS: distributed decision strategy based on switch migration towards SDN control plane[C]//2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Piscataway: IEEE Press, 2017: 486-493.

[11] 胡涛, 张建辉, 孔维功, 等. SDN 中基于双向匹配的多控制器动态部署算法[J]. 通信学报, 2018, 39(1): 159-169.
HU T, ZHANG J H, KONG W G, et al. Dynamic deployment algorithm for multi-controllers based on bidirectional matching in software defined networking[J]. Journal on Communications, 2018, 39(1): 159-169.

[12] XIAO H, HU B, ZHOU L, et al. DMSSM: a decision-making scheme of switch migration for SDN control plane[C]//IEEE 7th International Conference on Computer Science and Network Technology. Piscataway: IEEE Press, 2019: 348-352.

[13] XU Y, CELLO M, WANG I, et al. Walid dynamic switch migration in distributed software-defined networks to achieve controller load balance[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 515-529.

[14] MYKOLA B, ANDRII P, OLEKSIY P. et al. Dynamic switch migration method based on QoE-aware priority marking for intent-based networking[C]//15th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering. Piscataway: IEEE Press, 2020: 864-868.

[15] BARI M F, ROY A R, CHOWDHURY S R, et al. Dynamic controller provisioning in software defined networks[C]//Proceedings of the 9th International Conference on Network and Service Management. Piscataway: IEEE Press, 2013: 18-25.

[16] ZHOU Y, ZHENG K, NI W, et al. Elastic switch migration for control plane load balancing in SDN[J]. IEEE Access, 2018, 6: 3909-3919.

[17] WANG C, HU B, CHEN S Z, et al. A switch migration-based decision-making scheme for balancing load in SDN[J]. IEEE Access, 2018, 5: 4537-4544.

[18] GRKEMLI B, PARLAKK A M, CIVANLAR S, et al. Dynamic management of control plane performance in software-defined networks[C]//In 2016 IEEE NetSoft Conference and Workshops. Piscataway: IEEE Press, 2016: 68-72.

[19] LI Z Y, HU Y X, HU T, et al. Dynamic SDN controller association mechanism based on flow characteristics[J]. IEEE Access, 2019, 7: 92661-92671.

[20] CHENG G, CHEN H, HU H, et al. Dynamic switch migration towards a scalable SDN control plane[J]. International Journal of Communication Systems, 2016, 29(9): 1482-1499.

[21] FILALI A, CHERKAOUI S, KOBBANE A, et al. Prediction-based switch migration scheduling for SDN load balancing[C]//2019 IEEE International Conference on Communications. Piscataway: IEEE Press, 2019: 197-210.

[22] ASADOLLAHI S, GOSWAMI B, SAMEER M, et al. RYU controller's scalability experiment on software defined networks[C]//IEEE International Conference on Current Trends in Advanced Computing. Piscataway: IEEE Press, 2018: 1-5.

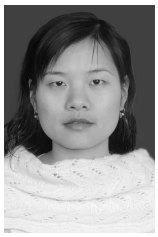
[23] PAL C, VEENA S, RUSTAGI R P, et al. Implementation of simplified custom topology framework in Mininet[C]//2014 Asia-Pacific Conference on Computer Aided System Engineering. [S.n.:s.l.], 2014: 48-53.

[24] KNIGHT S, NGUYEN H X, FALKNER N, et al. The Internet topology zoo[J]. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765-1775.

[25] 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. Piscataway: IEEE Press, 2016: 46-52.

[26] YAO G, BI J, LI Y, et al. On the capacitated controller placement problem in software defined networks[J]. IEEE Communications Letters, 2014, 18(8): 1339-1342.

[作者简介]



姚蓝（1982− ），女，河南信阳人，国家数字交换系统工程技术研究中心博士生，主要研究方向为软件定义网络。

兰巨龙（1962− ），男，河北张家口人，博士，国家数字交换系统工程技术研究中心教授、博士生导师，主要研究方向为未来信息通信网络关键理论与技术。