SDN 中基于分布式决策的控制器负载均衡机制

胡 涛[1],张建辉[1],邬 江[2],何为伟[3],江逸茗[1],赵 伟[1]
(1. 国家数字交换系统工程技术研究中心,河南郑州 450002;2. 中电长城网际系统应用有限公司,北京 102200;
3. 中国人民解放军 75775 部队,广东广州 510010)

摘 要: 针对 SDN 多控制器负载均衡过程中,控制器选取僵化和交换机迁移冲突问题,提出了一种基于分布式决策的控制器负载均衡机制,分为三个阶段进行实施:首先通过周期性收集网络信息,结合控制器负载状况构建分布式迁移决策域;然后在域中依据选取概率确定迁移交换机,综合权衡数据收集、交换机迁移和状态同步三种代价选择目标控制器;最后建立迁移时钟模型,完成交换机迁移和控制器角色转换. 仿真结果表明,与现有的负载均衡机制相比,降低了网络的通信开销,流建立时间平均缩短 0.14s,控制器资源利用率提高了 21.7%.

关键词: 软件定义网络; 控制器; 负载均衡; 交换机迁移; 分布式决策
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)10-2316-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.10.002

Controller Load Balancing Mechanism Based on Distributed Policy in SDN

HU Tao[1], ZHANG Jian-hui[1], WU Jiang[2], HE Wei-wei[3], JIANG Yi-ming[1], ZHAO Wei[1]
(1. *National Digital Switching System Engineering & Technology Research Center, Zhengzhou, Henan* 450002, *China*;
2. *GEG Cyber Space Great Wall, CS. LAB, Beijing* 102200, *China*;
3. *Troops* 75770 *of People's Liberation Army, Guangzhou, Guangdong* 510010, *China*)

Abstract: In the process of SDN multi-controller load balancing, controller selecting is rigid and switch migration is conflicting. This paper proposes a controller load balancing mechanism based on distributed policy, which is divided into three phases. Firstly, through collecting network information periodically, distributed migration decision domains are structured according to controller load conditions. Then the switches are migrated according to selecting probability. By balancing three costs included data collection, switch migration and state synchronization, the target controllers are determined. Finally, this mechanism sets the migration countdown to complete the switch migration and controller role conversion. Results show that, compared with the existing load balancing mechanism, this method could reduce the total network communication overhead, flow establishment time is shortened 0.14s averagely, controller resource utilization has been increased by 21.7%.

Key words: software defined network; controller; load balancing; switch migration; distributed policy

1 引言

软件定义网络[1](Software Defined Networking, SDN)作为一种新型网络体系架构,成功解决了传统网络的设计僵化问题,实现数据平面和控制平面的完全解耦. 随着网络规模的日益扩大,SDN 控制平面的单控制器设计已经不能满足现有的流量需求,因此业界相继提出了逻辑上集中,物理上分布的多控制器部署方案.

尽管多控制器架构提高了控制器的可扩展性和可靠性,但当控制器管理的交换机在某个时间段内发生流量激增或者瞬减时,很容易导致所属控制器出现热点(负载骤增)或者冷点(负载骤减)现象,不利于整个网络架构的稳定和控制器间负载均衡.

目前,关于控制器负载不均衡问题的解决方案可以分为两类:

收稿日期:2016-12-20;修回日期:2017-11-21;责任编辑:孙瑶
基金项目:国家网络空间安全专项课题(No. 2017YFB0803204);国家 863 高技术研究发展计划(No. 2015AA016102);国家自然科学基金创新研究群体科学基金(No. 61521003)

方案 1　通过改变控制器的配置方式和部署位置来避免控制器过载[2~6]. 这类方案无法根据实时流量状况对网络进行动态调整.

方案 2　通过迁移交换机来确保负载均衡[7~11]. 该方案虽然能够动态调整网络负载,但在交换机迁移过程中,关于目标控制器选取,仅考虑时延或控制器容量,容易导致目标控制器选取僵化.

针对上述问题,本文从交换机迁移的角度出发,提出了一种基于分布式决策的控制器负载均衡(Distributed Policy based Controller Load Balancing,DPCLB)机制. 本文的主要贡献和创新工作总结如下:

(1)结合交换机迁移思想,对 DPCLB 机制进行建模,考虑交换机选取概率、数据收集代价、交换机迁移代价和状态同步代价是影响控制器负载均衡的主要因素.

(2)构建分布式迁移决策域模型,在决策域中依据选取概率确定迁移交换机;基于贪婪算法选择目标控制器;建立迁移时钟模型,设定“迁移倒计时”,避免迁移冲突,有效提高交换机迁移效率. 多代价权衡和迁移时钟设定也保证控制器负载的均衡分布.

(3)从原理论证和仿真实验两个层面与现有机制进行比较,综合多种性能评价指标,基于仿真场景开展实验研究,验证 DPCLB 机制性能.

2　相关工作

在控制器负载均衡研究过程中,方案 1 作为一种先验式方案,主要有以下研究进展. Fu[2] 等人在多控制器部署的基础上,提出了一种控制器休眠模型,在流量负载较轻时,允许部分闲置控制器进入休眠状态. Guo[3] 等人提出基于负载方差同步的负载均衡方案,当控制器的负载超过预设阈值,控制器间实施负载同步,实现了无环路转发和良好的负载平衡性能. 张栋[4] 等人针对层次型多中心 SDN 的控制器部署问题,采用多层 k 路划分方法实现大规模 SDN 网络的区域划分,通过减少图划分的域间割边数以降低 SDN 跨域流数量. Santos[5] 等人提出了一种联邦控制器架构,使用聚类算法基于单个连接创建子网,优化了控制器部署位置的选取. Hock[6] 等人提出 POCO(Pareto-based Optimal Controller)框架,考虑不同的性能度量,使用 Pareto 优化配置,提升了网络的可靠性,但算法运行时间较长.

继 OpenFlow1.2[7] 协议提出之后,方案 2 逐渐受到研究者重视. Heller[8] 首次提出交换机迁移策略,考虑交换机和控制器之间的时延度量,通过迁出过载域中部分交换机实现控制器负载均衡. 不足之处在于该方案只是进行了数学分析,对于如何选取迁移交换机未做说明. 文献[9]中作者提出了一种 ElastiCon 架构,基于双门限值进行负载决策,根据控制器负载阈值将交换机就近迁移至邻居控制器. 但在该架构中,考虑因素仅限于单控制器过载状况,虽然减少了迁移时延,但当邻居控制器也处于高负载状况时,该迁移机制失效. 文献[10]中作者基于控制器剩余处理容量,提出了一种控制器负载均衡算法,迁移交换机至剩余处理容量最大的控制器. 王文博[11] 等人将交换机迁移优化成为控制器的热备份及选举问题,通过对控制器进行热备份,并设计相应的备份空间确定算法和主控制器选举算法,实现合理的网络构建.

综上所述,方案 2 相比方案 1,虽然改善了网络静态配属存在的缺陷,但控制器负载度量考虑单一,算法设计复杂,容易造成交换机迁移冲突问题.

3　分析与建模

3.1　问题分析

本文研究背景是 SDN 多域多控制器网络. 由于流量在时间和空间分布具有不均匀性和不确定性,因此,一旦某子域中部分交换机处于流量突发状态,很容易导致该子域控制器的处理资源被大量消耗,严重时有可能造成该控制器宕机. 另外一些子域的控制器需要处理的流请求较少,控制器资源利用率低下. 总而言之,控制器负载的不均衡分布导致控制器资源利用不合理,降低了网络整体的流传输与处理能力. 例如,在图 1 中,由于交换机 s_1 流量突发导致 c_2 成为过载控制器,但其余控制器仍处于负载正常状态.

通过对控制器负载不均衡问题进行深入分析可知,交换机和控制器之间的静态连接是导致控制器间负载无法有效转移的关键因素. 在 OpenFlow1.2 协议,交换机和控制器之间形成一种全新的连接关系,这使得通过交换机迁移调整控制器负载成为可能.

在图 1 中,当交换机 s_1 的 master 控制器 c_2 由于流量激增发生过载时,需要在 s_1 的 slave 控制器群(c_1,c_3,c_4)中选举一个控制器作为 master 控制器,完成交换机迁移任务. 但现有交换机迁移方案仍存在以下问题亟待解决,同时这些问题也是本文研究重点.

(1)如何构建 SDN 迁移域,确定迁移交换机和目标控制器;

(2)如何实施交换机迁移任务,避免交换机迁移冲突和控制器状态不一致问题.

3.2　DPCLB 建模

整个网络拓扑用无向图 $G=(V,E)$ 表示,V 和 E 分别表示网络中的节点集合和链路集合. 在网络中有 M 个控制器,控制器集合为 C,有 N 个交换机,交换机集合为 S,且 $|V|=|C|+|S|$. 假设控制器可以优化地部署在网络拓扑中[8],一个控制器管理多个交换机形成 SDN 子域. 设备间的跳数为 d,设交换机 s_i 和控制器 c_n 的连

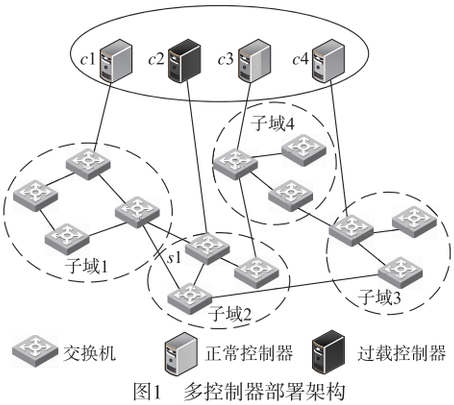


图1 多控制器部署架构

接关系为二进制变量 g_{in},其中 $g_{in}=1$ 表示第 i 个交换机和第 n 个控制器相连. 网络设备之间的连接关系构成连通矩阵 $\boldsymbol{H}=[g_{in}]_{N\times M}$. 交换机 s_i 的流请求(Packet-in 数据包)速率为 α_i. 控制器 c_n 管理的交换机数量为 U_n,处理容量为 β_n.

结合交换机迁移思想对 DPCLB 机制实施建模. 首先对迁移决策域进行介绍.

定义 1 迁移决策域 D_r,是指过载控制器 c_r 联合所有处于非过载状态的邻居控制器构成的控制器集合,$D_r=\{c_r,c_1,c_2,\cdots\}$. 例如,在图 1 中,$c_1$,$c_2$ 和 c_3 构成迁移决策域.

迁移决策域有两个基本特征:

特征 1 迁移决策域和过载控制器之间是一一对应关系,且呈现分布式部署.

特征 2 任意两个迁移决策域之间互不相交,不产生交集.

迁移决策域确定后,在过载控制器 c_r 的交换机集合 $S_r=\{s_1^r,s_2^r,\cdots,s_i^r,\cdots\}$ 中选取交换机 s_i 向目标控制器 c_k 进行迁移.

为了解决迁移决策域中迁移交换机和目标控制器选取问题,需要考虑如下因素.

(1)交换机选取概率

为了尽可能降低过载控制器的负载,选择迁出具有高流请求速率且距离过载控制器最远的交换机. 设集合 S_r 中交换机的选取概率为 φ_{ir},如式(1)所示,B 为计算参数,如式(2)所示,其中 λ_{ir} 表示交换机在控制器 c_r 上产生的资源利用率;d_{ir} 表示交换机到控制器的跳数. 选取概率 φ_{ir} 越大,则交换机 s_i 被迁移可能性越大.

$$\varphi_{ir}=\frac{e^{-B}}{\sum_{s_i^r\in S_r}e^{-B}} \tag{1}$$

$$B=\frac{\lambda_{ir}}{d_{ir}(1-\lambda_{ir})} \tag{2}$$

(2)数据收集代价

控制器需要和交换机进行周期性交互,收集跳数和流量信息. 设迁移决策域中控制器 c_r 的数据收集代价是 P_{data},式(3)所示,其中 v_r 是轮循交换机的平均比特速率.

$$P_{\text{data}}=\sum_{i\in S_r}d_{ir}\cdot v_r\cdot g_{ir} \tag{3}$$

(3)交换机迁移代价

交换机迁移过程中,控制器需要向交换机安装迁移规则,同时交换机也会和控制器进行通信,发出迁移请求. 因此交换机迁移代价包括迁移规则安装代价,交换机通信代价和迁移请求代价.

迁移规则安装代价 P_{rule} 是指控制器对域内迁移交换机安装 flow_mod 流规则所产生的代价,如式(4)所示,其中 δ_{rule} 是 flow_mod 数据包的平均大小.

$$P_{\text{rule}}=\delta_{\text{rule}}\cdot d_{ir}\cdot g_{ir} \tag{4}$$

交换机通信代价 P_{com} 是指迁移交换机和目标控制器正常传输数据过程中所产生的通信代价,如式(5)所示,其中 ε 是指交换机通信数据的传输速率. g_{ir} 和 g_{jk} 分别表示控制器 c_r 和 c_k 域内交换机和控制器的连接关系.

$$P_{\text{com}}=\varepsilon\cdot\Big(\sum_{i\in S_r}(g_{ir}\cdot d_{ir})+\sum_{j\in S_k}(g_{jk}\cdot d_{jk})\Big) \tag{5}$$

迁移请求代价 P_{req} 是指迁移交换机发送流请求到目标控制器所产生的代价,如式(6)所示,其中 $\min d_{ik}$ 表示迁移交换机到目标控制器的最短跳数.

$$P_{\text{req}}=(\min d_{ik})\cdot\varepsilon\cdot\Big(\sum_{i\in S_r}g_{ir}+\sum_{j\in S_k}g_{jk}\Big) \tag{6}$$

因此,通过对三种代价进行线性求和得到交换机迁移代价 P_{move},如式(7)所示:

$$P_{\text{move}}=P_{\text{rule}}+P_{\text{com}}+P_{\text{req}} \tag{7}$$

(4)控制器状态同步代价

在交换机迁移完成后,控制器之间需要进行状态同步,控制器 c_r 和 c_k 之间的同步代价如式(8)所示,其中 μ 指控制器间周期性共享的网络数据,与 ε 有关,但 $\mu<\varepsilon$,因为控制器不会共享所有子域信息,d_{rk} 指控制器 c_r 和 c_k 之间的跳数.

$$P_{syn}=\mu\cdot d_{rk}\cdot\Big(\sum_{i\in S_r}g_{ir}+\sum_{j\in S_k}g_{ik}\Big) \tag{8}$$

综合数据收集,交换机迁移和控制器状态同步三种代价,将迁移决策域中目标控制器选取问题转化为多代价混合线性规划问题进行求解. 目标函数如式(9)所示,其中 ω_1,ω_2,ω_3 分别是三种代价的权值.

$$P_{\text{object}}=\omega_1\cdot P_{\text{data}}+\omega_2\cdot P_{\text{move}}+\omega_3\cdot P_{\text{syn}} \tag{9}$$

$$\text{s.t.}\quad \forall i\in S,\quad j\in C,\quad \sum g_{ij}=1 \tag{10}$$

$$\forall c_i\in D_r,\quad d_{ir}\leqslant 1(i\neq r) \tag{11}$$

$$\forall r,k\quad D_r\cap D_k=\varnothing \tag{12}$$

式(10)保证每个交换机仅有一个 master 控制器. 式(11)表示迁移决策域内仅包含过载控制器和邻居控

制器. 式(12)表示两个迁移决策域间不存在交集.

4　DPCLB 机制

DPCLB 流程如图 2 所示,主要包含以下三个阶段:阶段 1,收集网络信息并构建迁移决策域;阶段 2,确定迁移交换机和目标控制器;阶段 3,交换机迁移和控制器角色转换. 当阶段 3 完成后进行负载判定,如果符合控制器资源利用率约束,则跳出 DPCLB 机制,否则跳转至阶段 2 中继续进行.

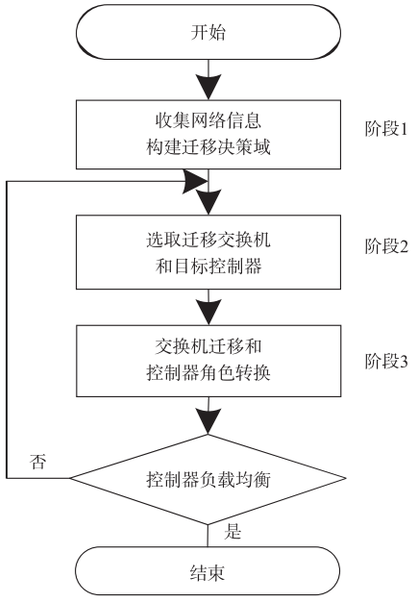


图2　DPCLB机制流程图

4.1　收集网络信息并构建迁移决策域

网络信息主要包括交换机-控制器之间的连接关系、控制器资源利用率和跳数信息. 控制器资源利用率的计算如式(13)所示,表示控制器 c_r 管理的交换机对控制器处理容量的占用程度. 迁移决策域的构建采用遍历式搜索,流程如下:收集网络的各类信息,类比文献[11],设定控制器过载条件,如式(14)所示,根据 η_r 对控制器进行过载判定,选取 $0.9 \leqslant \eta_r \leqslant 1$ 的热点控制器 c_r 作为过载控制器,并向所有邻居控制器发出交换机迁移请求,邻居控制器 c_i 接受到迁移请求之后有两种意向 < agree, discard > ,根据响应条件(如式(15)所示)对 c_r 做出响应,c_r 联合 response = = agree 的邻居构建迁移决策域 D_r.

$$\eta_r = \frac{\sum_{i=1}^{U_r} \alpha_i}{\beta_j} \tag{13}$$

$$\begin{cases} 0.9 \leqslant \eta_r \leqslant 1, & \text{controller } c_r \text{ is overload} \\ 0 < \eta_r < 0.9, & \text{controller } c_r \text{ is normal} \end{cases} \tag{14}$$

$$\text{response} = = \begin{cases} \text{agree}, & \eta_r < \sum_{j=1}^{M} \eta_j / M \\ \text{discard}, & \eta_r \geqslant \sum_{j=1}^{M} \eta_j / M \end{cases} \tag{15}$$

4.2　选取迁移交换机和目标控制器

在迁移决策域构建的基础上,阶段 2 的主要流程是:首先在过载控制器的子域集合中选取迁移交换机,然后在迁移决策域内选取目标控制器.

选取迁移交换机　在过载控制器 c_r 的交换机集合 S_r 中选择具有 $\max\varphi_{ir}$ 的交换机.

选取目标控制器　参考式(3)~(8),需要考虑数据收集,交换机迁移和控制器状态同步代价,将目标控制器选取优化为多代价混合线性规划,基于贪婪思想[12]设计目标控制器选取算法来求解,伪代码如算法 1 所示.

算法 1　目标控制器选取算法

```
Algorithm1:Object Controller Selection
输入:过载控制器 c_r
     迁移决策域 D_r
     迁移交换机 s_i^r
输出:目标控制器 c_obj
1:邻居控制器集合 N_r = {c_1, c_2, ⋯, c_k, ⋯}
2:计算跳数 d_ik
3:  while(min d_ik)
4:    尝试添加 s_i^r 到 c_k, S_k^* = S_k ∪ s_i^r
5:    if η_k^* > 0.9
6:        then N_r^* = N_r − c_k
7:        else 保留 c_k
8:    end if
9:  end while
10:选择 c_i ∈ N_r^*,计算 P_data, P_move, P_syn
11:  if c_i 有 max P_data
12:      then 移除 c_i
13:      else if c_i 有 min P_data
14:              then 保留 c_i
15:           end if
16:  end if
17:  重复 11~16 with P_move and P_syn
18:  对于 c_i 计算 P_object
19:    if current P_object < last P_object
20:     then 设置 min P_object = current P_object
21:  选择具有 min P_object 作为目标控制器 c_obj
22:  endif
```

算法的详细流程是:首先过载控制器检测其邻居控制器,得到邻居控制器集合(行 1). 基于跳数值,找到具有最小跳数的邻居控制器 c_k,尝试将交换机 s_i 迁移至

c_k(行3~4). 如果迁移完成后 c_k 的控制器资源利用率 η_k 超过阈值0.9,则将 c_k 从集合 N_r 中进行删除,否则保留 c_k(行5~7). 循环行3到行9过程,直至剔除完所有不符合要求的邻居控制器,形成新的邻居控制器集合 N_r^*,并计算相应的三种代价. 在此基于贪婪算法思想,如果交换机迁移至 c_i 使得网络中产生 $\max P_{\text{data}}$,则将 c_i 从 N_r^* 中移除,否则将 c_i 留在集合 N_r^* 中(行12~14). 整个过程尝试在每一次轮询中去除单方面代价最高的控制器,并使用最小代价配置作为循环开始点. 根据存储的控制器结果计算目标函数值(行18),选择具有最小函数值的控制器作为目标控制器(行21),算法结束.

该算法复杂度与控制器个数有关,算法将会执行 $|N_r| + (|N_r| - i) + \cdots + 1$ 次代价轮循,复杂度是 $O(m^2)$,其中 $m = |N_r| - i$. 在数据处理时,只需要进行对三种代价进行线性运算,最终算法快速收敛,具有实时性.

4.3 交换机迁移和控制器角色转换

阶段3主要流程是:过载控制器 c_r 选取迁移交换机 s_i^r 向目标控制器发出迁移信号 Move,目标控制器接收到信号 Move 后,回应 Move-Start 信号,并任意选取一个数 *num* 开始倒计时(待迁移交换机和邻居控制器数量越多,以及控制器资源利用率越高,则倒计时设置越短),在 *num* 减小至0时刻之前,如果完成交换机迁移任务,改变 s_i^r 的 master 控制器,同时将 c_r 变成 s_i^r 的 slave 控制器,解散迁移决策域. 如果超时或者迁移完成之后 c_r 仍处于过载状态($0.9 \leqslant \eta_r \leqslant 1$),则重置计数器,并返回阶段2.

$$num = \frac{\exp\left(\sum_{c_k \in N_r} \eta_k\right)}{|S_r| \cdot |N_r|} \tag{16}$$

5 性能评估

为了说明本文所提出的 DPCLB 机制性能,本节从原理论证和仿真实验两个层面将 DPCLB 机制与单控制器部署(Single Controller Deployment,SCD)机制、文献[11]中就近迁移机制(Nearest Migration,NM)、文献[4]中多控制器备份机制(Multi Controller Backup,MCB)进行对比,验证 DPCLB 机制的有效性和可行性. 各种机制性能描述如表1所示.

表1 机制比较

机制	性能描述
SCD	整个网络中仅部署一个控制器.
NM	交换机迁移至距离最近的控制器.
MCB	控制器备份池应对流量激增和瞬减.
DPCLB	基于分布式决策的控制器负载均衡.

5.1 原理论证

本小节从均衡原理和机制实施的角度出发,分别将 DPCLB 与其他机制进行对比.

(1)DPCLB 机制和 SCD 机制对比

SCD 仅在网络中部署单个控制器用于管理整个网络视图,尽管它的控制器资源得到充分利用,但一旦网络中发生流量突发,单控制器将立即过载. DPCLB 机制基于多控制器架构,通过交换机迁移完成负载调整,性能明显优于 SCD.

(2)DPCLB 机制和 NM 机制对比

NM 机制实施交换机迁移时,考虑流请求速率最高和跳数最小两方面因素. 迁移后的控制器负载方差 $LV_{\text{NM}} \propto (\max\alpha_t, \min d)$. DPCLB 机制实施交换机迁移时,通过计算选取概率选择迁移交换机,综合数据收集,交换机迁移和控制器状态同步三方面代价选择目标控制器. DPCLB 设定迁移完成后 $LV_{\text{DPCLB}} \propto (\max\varphi_t, \min d)$. 由于 φ_t 在选择迁移交换机过程中考虑了流请求速率 α_t,控制器可用处理容量 β_m 和跳数 d,同时目标控制器选取也经过贪婪算法优化,因此 DPCLB 确保不会出现交换机迁移完成后目标控制器成为新的过载控制器. 综上所述,$LV_{\text{DPCLB}} < LV_{\text{NM}}$,DPCLB 机制比 NM 机制具有更好的负载均衡性能.

(3)DPCLB 机制和 MCB 机制对比

MCB 在网络中设置控制器备份池来应对控制器过载状况. 只有当某些控制器发生过载时才会启用备份控制器去分担过载控制器负载. 尽管备份控制器在正常状态下未连接任何交换机,但仍会消耗网络资源. DPCLB 在负载调节过程中未新增控制器,迁移完成后负载方差 $LV_{\text{DPCLB}} < LV_{\text{MCB}}$. 因此,DPCLB 的负载均衡性能优于 MCB 机制.

对上述对比情况进行总结,就均衡原理和机制实施角度而言,DPCLB 的控制器负载均衡性能要优于 SCD,NM 和 MCB 机制.

5.2 仿真实验

5.2.1 仿真环境介绍

建立图3所示实验环境,做出如下说明:

(1)实验平台和物理设备介绍. 选取 OpenDaylight[13] 作为实验控制器,同时基于轻量级测试平台 Mininet[14]. 总共选取了8台具有相同实验配置的机器,Intel Corei7 3.2GHz 4GB RAM,装载 Ubuntu14.04 LTS 系统. 6台运行基于 DPCLB 的 OpenDaylight(No.1~6). 一台机器上仅安装 OpenDaylight 来模拟单个集中式控制器(No.7),另一台机器上运行 Mininet(No.8).

(2)拓扑选择. 使用两个真实的网络拓扑 OS3E[15] 和 Columbus[16] 验证 DPCLB 机制的有效性和拓扑适应性. OS3E 网络具有34个节点和42条链路. Columbus 网

络具有 70 个节点和 85 条链路.

(3)参数设定. 为了模拟真实的流量状况,所有的流都具有文献[17]中所示的流量特征,流大小呈现分布式,且具有不同的到达速率,平均流产生速率为 500K/s. 设定所有的控制器都具有相同的性能,处理容量的上限为 10000K,控制器资源利用率范围为 0 ~ 1,设交换机平均比特速率 v_r 为 50K/s,flow_mod 数据包 δ_{rule} 为 40 字节,交换机间通信速率 ε 为 20K/s,控制器之间共享的网络数据 μ 为 4K/s.

5.2.2　仿真结果分析

本文实验采用 OS3E 和 Columbus 网络,不同网络拓扑的流量分布如图 4 所示. 为了保证数据可靠性,排除无关因素干扰,进行多次重复实验,取实验平均值,并记录相关数据. 实验设计方面,实验 1 ~ 3 将 DPCLB 与表 1 中其他 3 种机制进行比较,分别评估了该机制在通信开销、流建立时间和控制器资源利用率方面的性能表现.

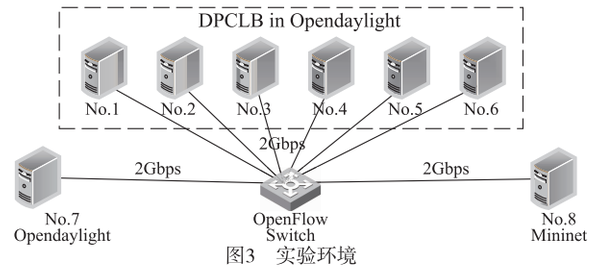
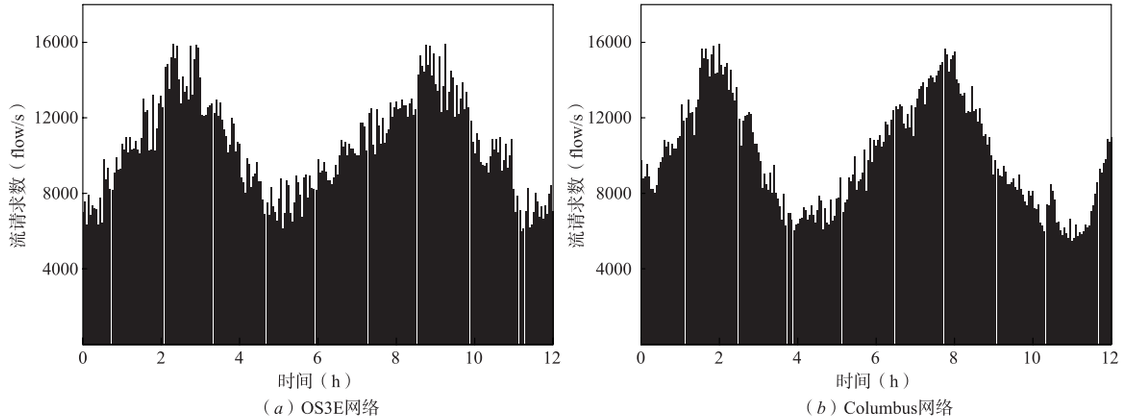


图3　实验环境

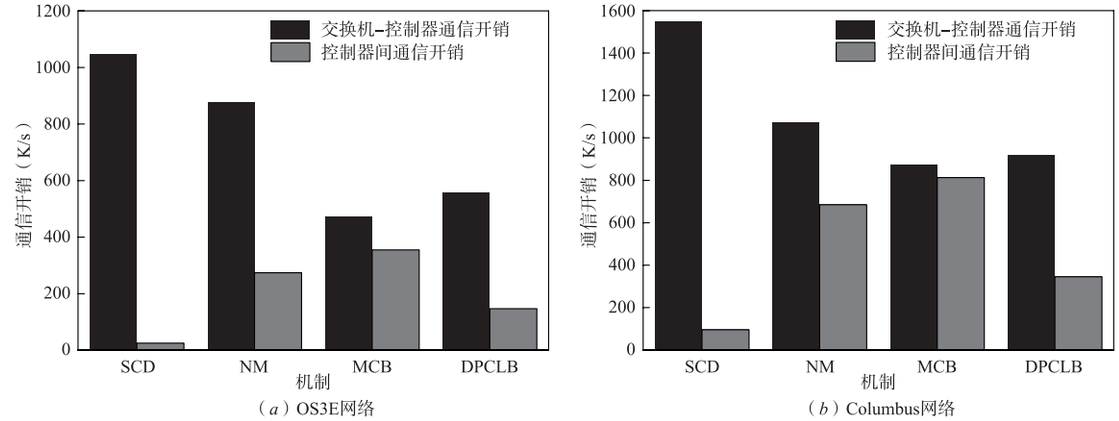
实验 1　通信开销

基于 OS3E 和 Columbus 的流量分布,对比不同机制的通信开销状况,实验结果如图 5 所示.



(a) OS3E网络　　(b) Columbus网络

图4　网络流量示意图



(a) OS3E网络　　(b) Columbus网络

图5　通信开销

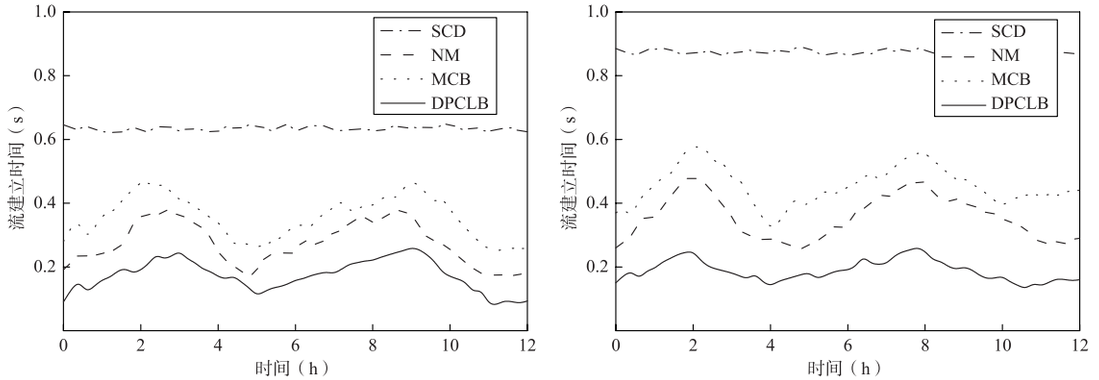
实验结果分析,因为 SCD 仅有一个控制器,所以控制器间通信开销基本为 0,流请求全部发送到该控制器中,控制器过载,交换机-控制器的通信开销最大. NM 将交换机迁移至距离最近的控制器,简化了目标控制器选取过程,控制器间开销较低,但多交换机同时涌入同一个距离最近控制器容易造成流量拥塞,加重了交换机-控制器通信开销. MCB 通过添加新控制器来均衡负载,交换机-控制器的通信开销最小. 新增控制器和所有控制器进行信息交互,因此控制器间通信开销最大. DPCLB 考虑多个负载代价,通过贪婪算法进行优化求解,在交换机-控制器通信开销方面和 MCB 结果基本相同(差值不超过 10%),但分布式迁移决策域的设定,减

少了无关控制器的信息交互,明显降低了控制器间通信开销,仅为 MCB 的 1/2,同时也小于 NM 机制.

在实现控制器负载均衡的基础上,4 种机制中 DPCLB 的网络总通信开销最少.

实验 2 流建立时间

如图 6 所示,SCD 均作为参照,单个控制器一直处于过载状态,因此流建立时间基本不随网络环境变化,一直维持在最高值. MCB 具有最剧烈的时间波动和高时间峰值,NM 次之,DPCLB 最小. 这是因为相比于交换机迁移,MCB 向网络中增加新控制器会造成网络状态的剧烈变化. NM 就近迁移交换机到邻居控制器,然而当邻居控制器也发生过载时,需要花费更多的时间去扩展迁移区域,流建立时间也因此增加. DPCLB 机制通过划分互不干扰的迁移决策域,同时进行多交换机迁移任务,对控制器设定“迁移倒计时”,避免交换机迁移冲突,因此流建立时间低于上述三种机制.



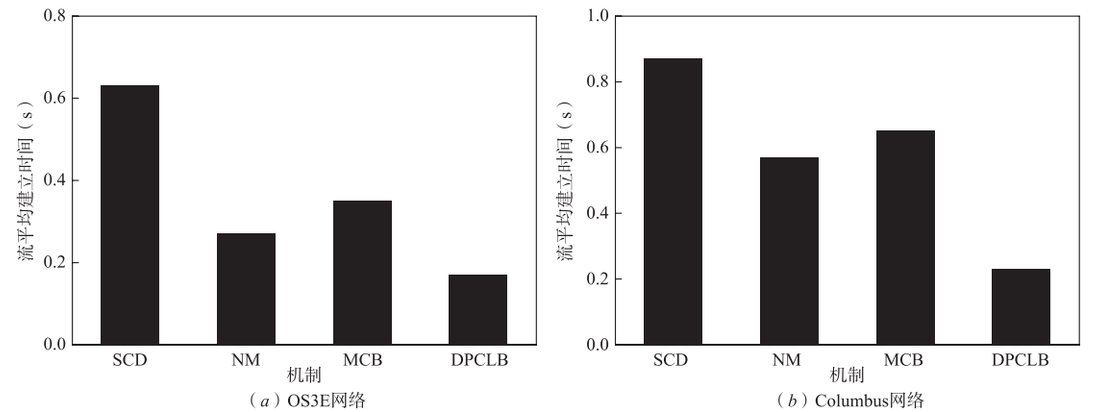
(a) OS3E网络 (b) Columbus网络

图6 流建立时间

对图 6 中实验结果进行处理,通过量化统计可以得出不同网络环境下 4 种负载均衡机制的流平均建立时间,如图 7 所示,DPCLB 机制的流平均建立时间相较于 SCD、NM 和 MCB,分别下降 51.3%,22.6% 和 31.4%,平均缩短了 0.14s.

实验 3 控制器资源利用率

控制器资源利用率作为负载均衡评价的重要指标,实验记录如图 8 所示. SCD 机制仅有一个控制器,因此控制器资源利用率接近 100%,处于完全过载状态. NM 就近迁移交换机到控制器,只是做到局部均衡,在全局网络中,控制器资源利用率仍低于 60%. MCB 和 DPCLB 都能较好地实现控制器负载均衡和资源的高效利用(约为 80%),但 DPCLB 比 MCB 部署了更少的控制器,控制器资源利用率提升了 21.7%.



(a) OS3E网络 (b) Columbus网络

图7 流平均建立时间

从拓扑角度进行横向对比,当网络规模扩大时(Columbus 比 OS3E 拥有更多的节点和链路数),Columbus 网络中 MCB 和 NM 机制的控制器资源利用率下降明显,但 DPCLB 机制仍维持在较高水平,这是因为随着网络节点数量增多,DPCLB 将会划分出更多的迁移决策域,从而提高交换机迁移效率,保证了控制器资源利用率.

6 结论

本文针对多控制器负载均衡过程中目标控制器选

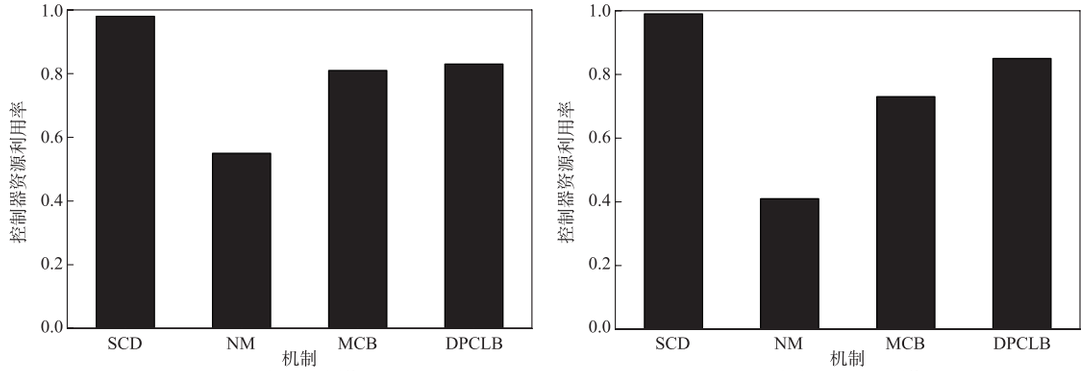


图8　控制器资源利用率

取困难和交换机迁移冲突问题,提出了一种基于分布式决策的控制器负载均衡(DPCLB)机制. 结合交换机迁移,构建分布式迁移决策域模型,并分为三个阶段进行实施,实现了交换机协调迁移和控制器角色转换. 在后续工作,搭建相关实验系统,部署 DPCLB 到真实环境中,进行性能评估.

参考文献

[1] MCKEOWN N, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. Computer Communication Review, 2008, 38(2): 69 – 74.

[2] FU Y, JUN B, WU J, et al. A dormant multi-controller model for software defined networking[J]. China Communications, 2014, 11(3): 45 – 55.

[3] Guo Z, Xu Y, et al. JumpFlow: Reducing flow table usage in software-defined networks [J]. Computer Networks, 2015, 92: 300 – 315.

[4] 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.
ZHANG Dong, GUO Jun-jie, WU Chun-ming. Controller placement based on hierarchical multi-center SDN[J]. Acta Electronica Sinica, 2017, 45(3): 680 – 686. (in Chinese)

[5] Santos J. Scalable design of SDN controllers for optical networks using federation-based architectures [A]. Proceedings of the 21st European Conference on Networks and Optical Communications (NOC) [C]. Lisbon Portugal: ISCTE-iUL, 2016. 70 – 75.

[6] Hock D, Gebert S, Hartmann M, Zinner T, Tran-Gia. POCO-framework for pareto-optimal resilient controller placement in SDN-based core networks[A]. Proceedings of the Network Operations and Management Symposium [C]. Krakow, Poland: IEEE, 2014. 1 – 2.

[7] OpenFlow Switch Specification, Version1. 2 [OL]. https://www. opennetworking. org. 2017 – 11 – 21.

[8] Heller B, Sherwood R, McKeown N. The controller placement problem[A]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN) [C]. New York, USA: ACM, 2012. 7 – 12.

[9] DIXIT A, HAO F, MUKHERJEE S, et al. Towards an elastic distributed SDN controller[A]. Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking [C]. New York, USA: ACM, 2013. 7 – 12.

[10] Yao G, Bi J, Li Y, Guo L. On the capacitated controller placement problem in software defined networks [J]. IEEE Communications Letters, 2014, 18 (8): 1339 – 1342.

[11] 王文博,汪斌强,陈飞宇,等. 一种软件定义网络中的控制器热备份及选举算法[J]. 电子学报,2016,44(4):913 – 919.
WANG Wen-bo, WANG Bin-qiang, CHEN Fei-yu, et al. The controller hot backup and election algorithms in software defined networking [J]. Acta Electronica Sinica, 2016, 44(4): 913 – 919. (in Chinese)

[12] Queyranne M, Spieksma F. A general class of greedily solvable linear programs [J]. Mathematics of Operation Research, 1998, 23: 892 – 908.

[13] OpenDaylight [OL]. http://www. opendaylight. org. 2017 – 11 – 21.

[14] Lantz B, Heller B. A network in a laptop: rapid prototyping for software-defined networks [A]. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets) [C]. New York, USA: ACM, 2011. 11 – 16.

[15] Internet2 Open Science, Scholarship [OL]. http://www. internet2. edu/network/ose/. 2017 – 11 – 21.

[16] 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.

[17] 杨洋,杨家海,秦董洪,等. DraLCD:一种新的数据中心流量工程方法[J]. 电子学报, 2017, 45 (5): 1261

–1267.
YANG Yang, YANG Jia-hai, QIN Dong-hong, et al. DraLCD: Another traffic engineering method for data center networks[J]. Acta Electronica Sinica, 2017, 45(5): 1261 –1267. (in Chinese)

作者简介

胡 涛 男. 1993 年 8 月生于陕西武功. 现为国家数字交换系统工程技术研究中心硕士研究生. 主要研究方向为新型网络体系结构.
E-mail: hutaondsc@163.com

张建辉 男. 1977 年 10 月生于河南平顶山. 现为国家数字交换系统工程技术研究中心副研究员. 研究方向为宽带信息网络, 网络安全.
E-mail: jhz@ndsc.com.cn

邬 江 男. 1987 年出生于山西岢岚. 现为中电长城网际系统应用有限公司副总. 主要研究方向为网络空间安全体系.

何为伟 男. 1981 年 3 月出生于湖北荆州. 现为中国人民解放军 75775 部队工程师, 研究方向为宽带信息网.

江逸茗 男. 1984 年出生于河南郑州. 现为国家数字交换系统工程技术研究中心助理研究员, 主要研究方向为宽带信息网络.

赵 伟 男. 1990 年出生于河南郑州. 现为国家数字交换系统工程技术研究中心硕士研究生, 主要研究方向为宽带信息网.