层次型多中心的SDN控制器部署

张 栋[1],郭俊杰[1],吴春明[2]
(1. 福州大学数学与计算机科学学院,福建福州 350108;2. 浙江大学计算机科学与技术学院,浙江杭州 310027)

摘 要: 软件定义网络(SDN)通过转发与控制分离,借助控制面的集中化实现网络的灵活性和开放性. 控制器部署是SDN部署运行的基础和前提. 针对层次型多中心SDN的控制器部署问题,该文采用多层 k 路划分方法实现大规模SDN网络的区域划分,将传统的SDN多控制器直接部署转化为区域划分和域内控制器部署,同时通过减少图划分的域间割边数以降低SDN跨域流数量以提高流表构建效率. 通过实验验证,较其他传统方法,该文提出的层次型多中心控制器部署方法可有效减少网络通信代价,降低流表构建代价.

关键词: 软件定义网络; 控制器放置; 多层 k 路划分
中图分类号: TP393.3 **文献标识码:** A **文章编号:** 0372-2112 (2017)03-0680-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.03.027

Controller Placement Based on Hierarchical Multi-center SDN

ZHANG Dong[1], GUO Jun-jie[1], WU Chun-ming[2]
(1. *College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian* 350108, *China*;
2. *College of Computer Science, Zhejiang University, Hangzhou, Zhejiang* 310027, *China*)

Abstract: Software defined networking (SDN) is a style of computer networking that separates the control plane from the data plane, shifting the control plane to a centralized controller in order to achieve network flexibility and openness. The controller placement is a key prerequisite to successful SDN. The current study examines the hierarchically distributed control plane controller placement problem, utilizing a multi-level k-way switch partition algorithm to divide large scale network topology. We also fix the traditional SDN controller placement problem, changing zoning and intra-domain controller placement by reducing the edge-cut in order to lower the number of inter-domain flows. Simulation results show that the multi-level k-way switch partition algorithm can effectively reduce control flow overhead and flow set-up time, compared with the other traditional algorithms.

Key words: software defined networking; controller placement; multi-level k-way switch partition

1 引言

传统网络中,路由控制和数据转发是紧密结合的. 随着网络规模的迅速扩大,作为网络核心的路由器,其承载功能不断扩展,如分组过滤、区分服务等. 路由器只能通过命令行接口等方式对外开放少量的功能,难以在真实网络中实验和部署新型网络技术架构. 针对现有互联网体系面临的困境,美国斯坦福大学clean slate研究组提出一种新型网络架构:软件定义网络(Software Defined Networking, SDN)[1],其核心思想是控制平面和数据平面相分离,控制功能集中到中心控制器,将数据转发放在支持OpenFlow的交换机上,实现数据域与控制域的解耦[2]. SDN为用户提供网络灵活拓展部署的能力,通过对SDN控制平面的操作,更容易部署新一代网络协议[3,4].

经典的SDN架构,依赖拥有全局视图的单一控制器[5]. 当交换机收到一条新的流时,需要问询控制器,控制器确定转发规则并将相应流表信息下发交换机,即流表建立过程[6]. 在大规模网络环境中,单一控制器难以及时处理大量新流请求,导致网络响应变慢,网络服务质量和扩展性受到制约. Kandoo[7], HyperFlow[8], 和Onix[9]等分别采用不同策略实现多个控制中心的

收稿日期:2015-06-29;修回日期:2015-08-07;责任编辑:梅志强
基金项目:国家重点基础研究发展计划973课题:(No. 2012CB315903);浙江省重点科技创新团队(No. 2011R50010-05, No. 2013TD20);国家863高技术研究发展计划项目:(No. 2015AA016103);国家自然科学基金项目:(No. 61379118);福建省引导性重点项目(No. 2016Y0060)

SDN 架构. Kandoo 将控制器划分成两层,第一层由一个 Root 控制器组成,第二层由多个控制器组成,如图 1 所示. Root 控制器拥有网络全局视图,并决定转发规则. HyperFlow 分配控制器的方式是基于 WheelFS 分布式文件系统. Onix 则通过 hash 表管理控制器.

在大规模 SDN 环境中,如何确定多控制器的部署位置,即控制器部署问题,是多控制器 SDN 部署的基础和前提. 文献[10]为减少流建立时间和控制流传输代价,动态调整控制器的数量和位置. 文献[11]根据网络节点间的最短距离选择控制器的部署位置,分析控制器数量和不同位置对传播时延的影响. 在此基础上,文献[12]考虑传输时延(PD)和发送时延(TD)对控制器部署的影响,提出基于时延的网络快速响应控制器部署方案;文献[13]通过帕累托优化确定 SDN 中需要放置多少个控制器才能使时延和网络恢复能力都达到最优. 文献[14]则通过设计一种基于帕累托优化的框架,综合考虑延迟约束、容错率和负载均衡对控制器部署的影响,通过采用模拟退火算法的设计思路选择控制器部署方式. 文献[15]设计一种可编程的控制层,通过这个控制层实现降低 SDN 分布式控制器平面因控制器状态、数据平面状态造成的编程复杂度. 文献[16]以优化控制器和交换机间的通信可靠性为目标,确定控制器放置位置. 上述研究主要针对平面型多中心架构,没有考虑层次型 SDN 控制器部署的需要;其次,集中式的部署策略难以满足大规模跨域 SDN 环境的需要. 文献[17]提到层次型控制器结构,但没有考虑合理的拓扑划分方式以有效减少流表构建代价. 本文针对上述问题,提出基于多层 k 路划分的方法,先划分区域再部署控制器以满足多控制器并行部署的要求;同时,在区域划分过程中,针对跨域流对流表建立时间和多控制器协同的延时影响,在分区过程考虑割边最少化.

本文将控制器部署问题抽象为交换机划分和控制器部署问题(Switch Partition and Controller Placement Problem,SPCPP). 根据 SDN 中各交换机链路间的带宽值、交换机处理流表的能力以及相互间的物理距离等因素划分区域,将一个大区域通过层次化方法,均衡划分成几个小区域. 在区域划分后,综合考虑通信代价和传输时延选择每个域的控制器部署位置,实现多区域的控制器并发部署. 本文通过多层 k 路划分算法,减少域间割边,降低跨域流产生概率,减少流表构建时间和跨域延时,同时选择合适位置部署控制器减少网络通信代价.

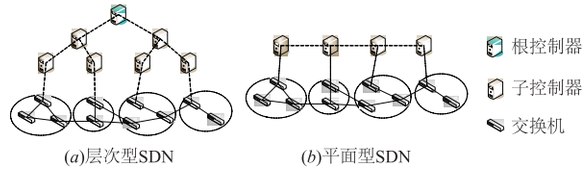

图1　SDN控制器放置架构

2　问题描述

将 SDN 网络表示为无向图 $G=(N,E)$,其中,N 表示交换机集合,E 表示链路集合. 使用 $F(F\subseteq N)$ 表示与控制器直连的交换机集合. 使用向量 $\boldsymbol{A}=[a_1,a_2,\cdots,a_{|F|}]$ 表示控制器每秒所能处理流的数量. 使用矩阵 $\boldsymbol{H}=[h_{ij}]_{|F|\times|O|}$ 表示层次型控制器的分配情况,h_{ij} 是一个二元组(α_{ij},β_{ij}),α_{ij} 表示控制器 i 在第 j 层所属的区域,β_{ij} 表示控制器 i 的父节点所在区域,其中 $|O|$ 表示控制层的层数. 使用矩阵 $\boldsymbol{X}=[\chi_{im}]_{|N|\times|F|}$ 表示交换机分配情况,当 χ_{im} 的值为 1 时,交换机 i 受控制器 m 管理,否则 χ_{im} 的值为 0. 使用 $\boldsymbol{T}=[\tau_{ij}]_{|N|\times|N|}$ 表示通信量矩阵,其中 τ_{ij} 表示每秒交换机 i 向交换机 j 发送的流的数量. 使用数组 D 表示控制器放置位置. n_i 发出的流的数量 $w(n_i)$ 来表示 n_i 的权重,根据式(1)计算:

$$w(n_i)=\sum_{j\in N}\tau_{ij} \tag{1}$$

r_{ij} 表示交换机 i 到交换机 j 的最短路径,r_{ij} 是边 e 的集合. 将经过路径 r_{ij} 的流数量总和作为路径 r_{ij} 的权重,用 $w(r_{ij})$ 表示,可以根据式(2)计算:

$$w(r_{ij})=\sum_{k\in N}\sum_{h\in N}\tau_{kh},\ r_{ij}\subseteq r_{kh} \tag{2}$$

将 n_i 到 n_j 最短路径的通信开销用 $l_{ij}=\dfrac{d_{ij}}{b_{ij}}$ 表示,其中 d_{ij} 表示交换机 n_i 与 n_j 间最短路径的长度(可以使用跳数衡量),b_{ij} 表示 r_{ij} 的带宽值.

区域负载(Load)为各个控制域中控制器负载的集合. 控制器负载指的是该控制器每秒处理的流数目总和,控制器 m 的区域负载表示为:

$$\mathrm{Load}_m=\sum_{i\in N}\sum_{j\in N}\tau_{ij}\chi_{im} \tag{3}$$

此时,$\mathrm{Load}=\{\mathrm{Load}_m\mid 1\leqslant m\leqslant|F|\}$.

流表构建代价是评价 SDN 控制器放置算法的重要标准. 流表构建包括三个阶段:交换机向控制器发出流请求的初始化阶段、控制器流表路径的计算阶段和控制器向交换机下达流表的安装阶段. 初始化阶段可以用式(4)表示:

$$C_p^I=\sum_{i\in N}\sum_{m\in F}\tau_{ii}\chi_{im}d_{im} \tag{4}$$

路径计算阶段表示控制器进行路径计算,可以使用 C_p^Q 表示,C_p^Q 是由控制器内部路径计算组件进行计算,通过控制器的拓扑管理组件获知网络情况,再根据拓扑进行路径计算.

流表安装阶段表示控制器计算完路径之后,向各个交换机下发流表的过程,可以使用式(5)表示:

$$C_p^L=\sum_{i\in N}\sum_{j\in N}\sum_{m\in F}\tau_{ji}\chi_{im}d_{im} \tag{5}$$

由此可得流表构建代价：

$$C_p = C_p^I + C_p^Q + C_p^L \tag{6}$$

网络通信代价(Total)表示某个时刻各交换机相互通信的开销,与流表构建代价,交换机之间的通信量和链路带宽等相关. 式(7)表示网络通信代价.

$$\text{Total} = C_p + \sum_{i\in N}\sum_{j\in N}\tau_{ij} * l_{ij} \tag{7}$$

SPCPP 的目标是：

$$\min(\text{Total}) \tag{8}$$

式(4)~(6)需要满足以下约束条件：

$$\forall_{i\in N}: \sum_{m\in F}\chi_{im} = 1 \tag{9}$$

$$\forall_{m\in F}: \sum_{i\in N}\chi_{im}\tau_{ii} \leqslant a_m \tag{10}$$

约束式(9)表示对于任意的交换机只属于某一个控制器管理,约束式(10)表示初始化阶段时,交换机 i 产生流的数量不能超过控制器所能处理的流数量上限.

3 层次型多中心 SDN 控制器部署

3.1 区域划分

本文使用层次型多中心部署算法(Hierarchical Multi-Center Placement, HMCP)对拓扑结构进行划分. HMCP 分为三个阶段,其流程如图 2 所示,第一阶段粗化阶段,一系列较小的图 $G_i(N_i,E_i)$ 是由原始图 $G=(N,E)$ 生成的,且 $|N_i|<|N|$. 由 $G_i(N_i,E_i)$ 的一组顶点合并成单个顶点粗化成 $G_{i+1}=(N_{i+1},E_{i+1})$,通过多次粗化操作将图缩小到合适的规模;第二阶段是初始划分,将第一阶段最终得到的图划分成 k 个区域;第三阶段则是将第一阶段产生的缩小图恢复成初始图 $G=(N,E)$,以此达到减少割边的效果.

3.1.1 粗化阶段

对较大的网络拓扑按传统方法划分,时间复杂度较高. 因此在划分前,对网络拓扑 $G=(N,E)$ 通过图匹配[18]和极大匹配[18]实现粗化. 图匹配是指边的一个集合,该集合不含环且任意两边皆不相邻. 图的极大匹配是边数最多的图匹配. 对点的粗化操作可以有效减少图的规模.

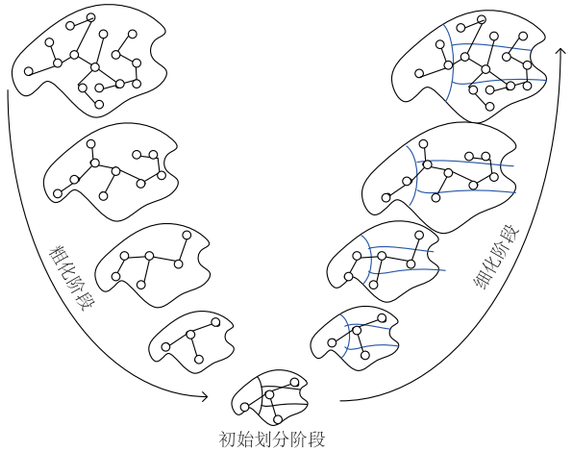

图2 层次型多中心放置算法

定义 1 顶点集合 N_i^n:一组需要合并的顶点集合.

定义 2 边的权重之和 $W(E)$:所有属于集合 E 的边的权重之和.

为更好反映原图的信息,本文算法将合并后得到的新顶点的权重值设置为 N_i^n 中所有元素权值之和,为保证原图的连通性,将新得到的边的权重值设为 N_i^n 中所有边权重值之和.

粗化阶段,使用重边匹配算法[18]将原始网络拓扑 $G=(N,E)$ 粗化成 $G_1=(N_1,E_1)$,接着 $G_2=(N_2,E_2)$、……$G_m=(N_m,E_m)$,$G_m=(N_m,E_m)$ 终止粗化的条件:$|N_m|\leqslant c$,其中 c 是一个常数.

使用重边匹配算法来寻找极大匹配. 假设一个匹配 M_i 用来粗化图 $G_i=(N_i,E_i)$,并且得到粗化后的拓扑 $G_{i+1}=(N_{i+1},E_{i+1})$. 则此时可得式(11)

$$W(E_{i+1}) = W(E_i) - W(M_i) \tag{11}$$

使用重边匹配算法得到图 $G_i=(N_i,E_i)$ 的极大匹配 M_i,如算法 1 所示.

算法 1 重边匹配(Heavy Edge Matching, HEM)算法

输入:$G_i=(N_i,E_i)$
输出:M_i
(1)初始化 M_i 为 NULL;
(2)随机遍历图中节点 n_i,若 $n_i \notin M_i$
(3)随机选取节点 n_i 的一个邻近节点 n_j,
  if n_j exist,
    将边(n_i,n_j)归为 M_i,且 n_j 标记为已匹配
  else n_j 不作处理
(4)直到遍历完所有节点
(5)再次随机选取图中节点 n_i,若 n_i 未匹配,此时选择 n_i 邻近的未匹配且 $w(r_{ij})$ 最大的节点 n_j
  if n_j exist,
    将边(n_i,n_j)归为 M_i,且 n_j 标记为已匹配
else n_j 不作处理
(6) 执行步骤(5),直到所有的节点都被匹配

3.1.2 初始划分阶段

定义 3 收益 g_v:顶点 v 从一个区域移动到另一个分区所减少的割边权重值.

在粗化阶段,原始网络拓扑 $G=(N,E)$ 最终被粗化为 $G_m=(N_m,E_m)$,其中 $G_m=(N_m,E_m)$ 的顶点数 $|N_m|$ 小于 c. 在初始划分阶段,针对规模较小的 $G_m=(N_m,E_m)$ 进行划分. 使用 P_m 记录图 $G_m=(N_m,E_m)$ 的划分情况,P_m 是 $\{k_{m0},k_{m1},\cdots,k_{mi}\}$ 的集合,k_{mi} 表示的是 $G_m=(N_m,E_m)$ 中点 i 所对应的分区,取值范围为 1 至 k. 划分的标

准是每个区域的顶点权重之和尽量为原图顶点之和的 $1/k$.

假设 P_m 是 $G_m=(N_m,E_m)$ 的初始划分,收益 g_v 可以用式(12)表示:

$$g_v=\sum_{(v,u)\in E\wedge P[v]\neq P[u]}t_{uv}-\sum_{(x,y)\in E\wedge P[x]=P[y]}t_{xy} \quad (12)$$

当 $g_v>0$ 时,表示顶点 v 移动到另一区域时,割边总权重减少,反之,$g_v<0$ 时,割边总权重增加.

采用 Kernighan-Lin(KL)[19]算法实现划分. KL 算法是一个基于迭代的算法,KL 算法如算法 2 所示.

算法 2　Kernighan-Lin(KL)算法

(1)输入 $G=(N,E)$

(2)执行划分算法(如算法 3 所示)

(3)得到 2 个子图,将其作为输入,执行步骤(1),直到移动了 f 个节点后,图的割边总权重不再减少后,结束算法

KL 执行的划分如算法 3 所示.

算法 3　划分算法

输入:P_m

(1)定义两个队列 q_0,q_1,分别保存二分图中的顶点,q_0,$q_1\leftarrow$unlocked.

(2)迭代每个 $q_0[v]$ = unlocked&&max(g_v).

(3)将这个顶点 v 移动到 q_1,更新 v 临近顶点的收益 g_v,$q_0[v]\leftarrow$locked.

(4)直到 q_0,q_1 没有未锁定的顶点时算法结束.

(5)输出更新后的 P_m

在执行完 KL 算法之后,可以得到粗化后的拓扑被划分的子图数 k,即将拓扑划分成 k 个区域.

3.1.3　细化阶段

将粗化网络分成 k 个区域后,逐步还原为原始拓扑,当 $G_m=(N_m,E_m)$ 还原为 $G_{m-1}=(N_{m-1},E_{m-1})$ 时,所有 $G_{m-1}=(N_{m-1},E_{m-1})$ 合并的顶点都被划分到了同一个区域. 此时,虽然 $G_m=(N_m,E_m)$ 的划分 P_m 已经达到最小划分,即划分的割边达到最小的目标,但对于 $G_{m-1}=(N_{m-1},E_{m-1})$,可能存在具有更小割边的划分方式,因此需要对 $G_{m-1}=(N_{m-1},E_{m-1})$ 进一步细化.

细化是针对每个 P_i(其中 $i\in(1,m)$)进行调整,使其达到最小局部划分的过程. 本文使用 Greedy Refinement(GR)算法[18]实现细化,如算法 4 所示.

算法 4　Greedy Refinement(GR)算法

输入:$G_i=(N_i,E_i)$

输出:P_{i-1}

(1)随机遍历 $G_i=(N_i,E_i)$ 中的节点

(2)获得节点 n_i,此时 $k_{ii}=a$,表示 $\chi_{ia}=1$. 如果节点 n_i 是内部节点,此时节点 n_i 不用移动

(3)如果 n_i 属于区域 a 的边界顶点. 令 P'为 P_i 的一个子集,当节点 n_i 满足以下两个条件之一则将 n_i 移动到区域 b 中:

①$ED[n_i]_b>ID[n_i]_b$ 和 $ED[n_i]_b$ 是 $b\in P'$中最大的

②$ED[n_i]_b>ID[n_i]_b$ 和 $w_i[a]-w_i[b]>w(n_i)$

(4)更新集合 ED 和 ID 的值

GR 算法中集合 ED 和 ID,ED 表示该节点与其他域节点的连接数,$ED[n_i]_b$ 表示节点 n_i 与区域 b 中相连节点的数量;$ID[n_i]_b$ 表示 n_i 所在区域 b 中与其相连的节点数量. ED 和 ID 的具体实例如图 3.

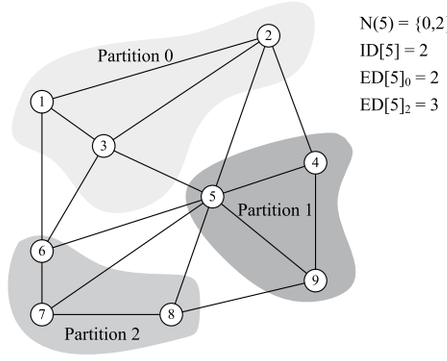

图3　ED和ID实例

3.2　控制器部署

通过 HMCP 算法,将交换机划分好区域后,需要找到控制器的适当位置,降低流表建立时间的代价. 新流建立的主要过程包括:(1)交换机向控制器发起流建立请求,(2)控制器计算流的路由路径,(3)控制器下放规则配置给交换机. 其中过程(2)的时间花费由控制器的性能及其负载等决定,而过程(1)和(2)的时间花费则和控制器的放置位置有关. 层次型控制器部署分为两个步骤,第一个步骤为在划分好的交换机区域内部署控制器,这个过程称为域内控制器部署;第二个步骤为域间控制器部署,此时需要确定上层控制器的放置位置.

3.2.1　域内控制器部署

域内控制器算法指的是经过 HMCP 算法将一个拓扑划分成固定的区域数之后,则需要在每个区域中选择一个合适的位置作为控制器放置位置. 本文采用 HMCP-MinMatch 算法来确定控制器放置位置. 在该算法中引入一个变量 λ,通过变量 λ 衡量网络通信代价和传播时延的权重. 算法的主要思想为综合考虑各个区域中网络通信代价和传播时延的情况下,选择一个合适的控制器放置位置,如算法 5 所示.

算法 5　HMCP-MinMatch 算法

输入:$G=(N,E)$

输出:D

(1) 初始化 $m=1$,初始化数组 t 的值全为 0

(2)选择 $\forall i \in \{1 \leqslant i \leqslant |N| \ \&\& \ P_i = m\}$,使用式(7)计算这个区域中 $t = \sum_{j=1}^{|N|} t_{ij} * l_{ij}$

(3)选择 $\min(t)$,并将此时的 i 赋值给 D_h

(4)对每个区域执行步骤(1)和(2),$m = m+1$,直到 $m \equiv k$ 时算法结束

3.2.2 域间控制器部署

在确定完域内控制器放置位置后,需要部署上层控制器,即域间控制器或根控制器(Root Controller),负责跨区的流表构建和所辖域内控制器的协调. 由于域间控制器主要进行信息交互,需要着重考虑传输时延(Propagation Delay, PD)[5],本文使用最小传播时延(MinPD)作为域间控制器放置算法. MinPD 算法如算法 6 所示.

算法 6 MinPD 算法

输入:$k, G=(N,E), D$,

输出:D'

(1) 初始化 $m=1$,初始化数组 t 的值全为 0

(2)选择 $\forall i \in \{1 \leqslant i \leqslant |N| \ \&\& \ P_i = m\}$,计算这个区域中其他节点到点 i 的传播时延 $t_i = \sum_{j=1}^{|N|} d_{ij}$

(3)选择 $\min(t)$,并将此时的 i 赋值给 D'

(4)对每个区域执行步骤(1)和(2),$m = m+1$,直到 $m \equiv k$ 时算法结束

MinPD 算法中输入参数 k 表示控制器所要划分的层次数量,D 表示下层控制器放置位置的数组,D' 表示上层控制器放置位置.

4 实验与结果

本文采用物理网络拓扑 Internet2 OS3E 和 50sw 拓扑进行实验,50sw 是由 GT-ITM 随机生成的一个网络拓扑结构. 网络流量模拟采用泊松随机过程生成通信量,能较好反映网络中流量生成情况,链路间的带宽值设置为 100Mbps.

OS3E 和 50sw 拓扑信息如表 1 所示.

表 1 OS3E 和 50sw 拓扑信息

网络拓扑	节点数	边数	主机数
OS3E	34	41	78
50sw	50	129	100

本文对比两种常见的网络拓扑划分算法,与 HMCP 相比较. 如表 2,这两种算法分别是随机放置算法和贪心放置算法,其输入包含网络拓扑结构 $G=(N,E)$,各个交换机的能力值集合 S,各个控制器能力值的集合 U,其中 $|S| = |N|, |U| = k$.

表 2 算法简述

算法名称	主要描述
HMCP-MinMatch	首先对网络拓扑进行粗化并划分,接着还原为初始网络;接着根据 MinMatch 算法选择控制器部署位置
Greedy-MinMatch	将问题抽象为一个背包问题对网络拓扑结构进行划分,接着根据 MinMatch 算法选择控制器部署位置
HMCP-MinPD	首先使用 HMCP 算法划分拓扑,接着在每个区域中选择一个传输时延最低的节点作为控制器部署位置
Greedy-MinPD	将问题抽象为一个背包问题对网络拓扑结构进行划分,接着在每个区域中选择一个传输时延最低的节点作为控制器部署位置

4.1 在 OS3E 拓扑上的验证

OS3E 拓扑由 34 个城市节点组成,控制器个数为 2,4,8,16,32,记录网络通信代价、流表构建代价.

4.2 在 50sw 拓扑上的验证

50sw 拓扑由 50 交换机节点组成,控制器个数分别为 2,4,8,16,32,记录网络通信代价、流表构建代价.

图 4、图 6 是控制器个数从 2 增加到 32 时,网络通信代价的统计结果. 从实验数据可知,随着控制器数量的增加,由于各个算法划分网络拓扑的方式不同,网络通信代价出现了不同的变化,接近线性的下降过程. HMCP 相较于其它三种算法,由于在 SPCPP 中考虑到了节点的权重和链路上的权重,每个划分的区域中节点和链路的权重都大体保持均衡,因此,在降低网络通信代价上具有良好效果. 图 5、图 7 是控制器个数从 2 增加到 32 时,流表构建代价的统计结果,从实验数据可知,随着控制器数量的增多,流表构建代价也会逐渐地减少,相较于其他三种算

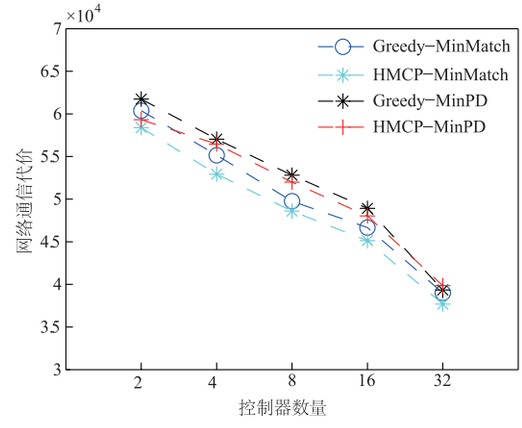

图4 网络通信代价

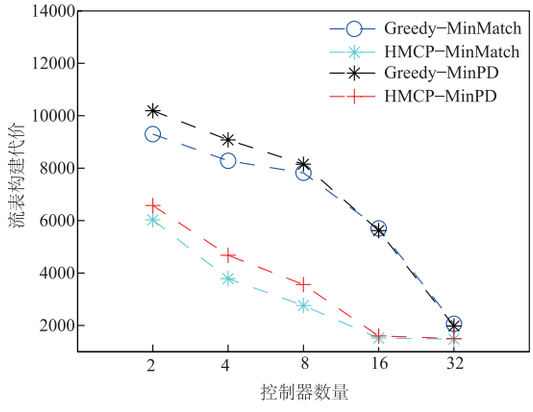

图5　流表构建代价

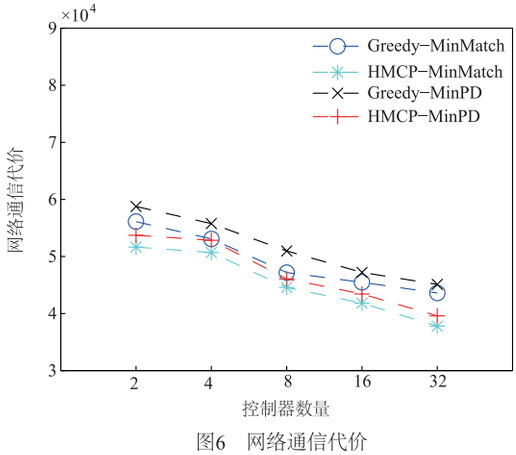

图6　网络通信代价

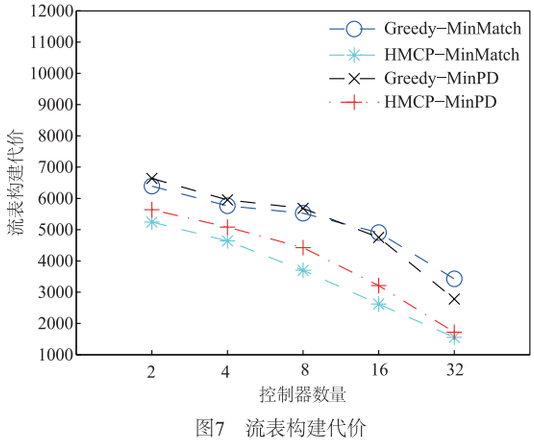

图7　流表构建代价

法,由于在划分拓扑时,HMCP 考虑到了减少域间割边数量,能够有效地减少跨域流产生的概率,因此,HMCP 在减少流表构建代价上具有较好地效果.

4.3　控制器层次型部署验证

图 8 和图 9 表示拓扑 OS3E 和 50sw 使用不同层数控制器放置方案获得的流表构建代价实验结果. 从图 8、图 9 可以看出对于不同拓扑而言,随着控制其数量地不断增加,流表构建代价则会逐渐减少. 从图 8 中可以

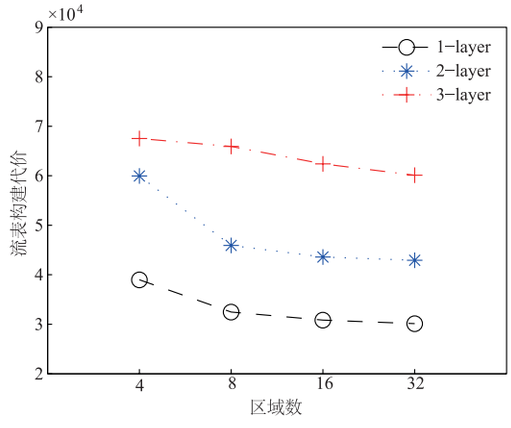

图8　os3e层次型控制器放置

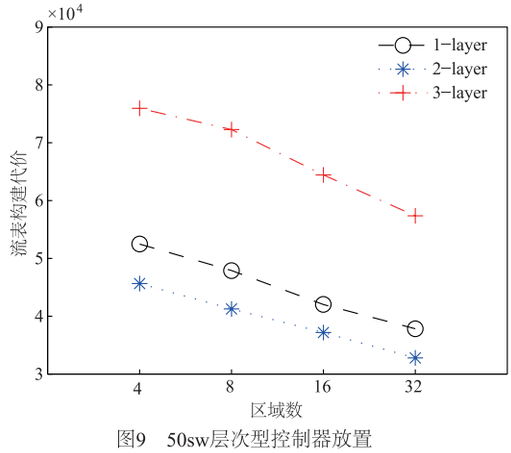

图9　50sw层次型控制器放置

看出,对于 OS3E 拓扑而言,当控制器层数为一层时,流表构建代价要小于层数为两层和三层的情况. 从图 9 中可以看出,对于 50sw 拓扑而言,当控制器层数为两层时,流表构建代价最小. 因此,针对不同拓扑的网络,需要合理分析控制器层数对流表构建代价的影响,选择合适的控制器层数,让流表构建代价达到最小.

5　总结

本文在 SDN 控制器放置问题中,将大规模 SDN 通过层次化方法,均衡划分成几个小区域. 在区域划分后,依据 HMCP-MinMatch 算法选择每个区域的控制器放置位置,实现多区域的分布式并发放置控制器. 在控制器数量相同的情况下,通过减少各个区域之间的割边数量,即减少跨域流产生的概率,能够有效地减少网络通信代价和流表构建代价;网络通信代价与流表构建代价随着控制器数量增多,也会相应地减少. 本文提出的 HMCP 算法在减少网络通信代价与流表构建代价上具有较好地效果.

参考文献

[1] Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enablinginnovation in campus networks[J]. ACM

SIGCOMM Computer Communication Review, 2008, 38(2):69 – 74.

[2] Jain S, Kumar A, et al. B4: Experience with aglobally-deployed software defined WAN[A]. The Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking[C]. Hong Kong: ACM, 2013, 3 – 14.

[3] 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, ChenFei-yu, et al. The controller hot backup and election algorithm in software define networks[J]. Acta Electronica Sinica, 2016, 44(4): 913 – 919. (in Chinese)

[4] 兰巨龙,莫涵,胡宇翔. 软件定义网络架构研究与实践[J]. 中兴通信技术,2013,19(5):11 – 15.
Lan Julong, Mo Han, Hu Yuxiang. SDN architecture: Research and practice[J]. Zet Technology Journal, 2013, 19(5):11 – 15. (in Chinese).

[5] B Heller, R Sherwood, N Mc Keown. The controller placementproblem[A]. Proceedings of HotSDN[C]. New York: ACM, 2013. 7 – 12.

[6] Bari M F, Roy AR, Chowdhury SR, et al. Dynamic controller provisioning in software defined networks[A]. International Conference on Network and Service Management[C]. Zuric: IEEE, 2013. 18 – 25.

[7] S Hassas Yeganeh, Y Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications[A]. Proceedings of HotSDN[C]. New York: ACM, 2012. 19 – 24.

[8] Mueller J, Wierz A, Vingarzan D, et al. Elastic networkdesign and adaptive flow placement in software definednetworks[A]. 2013 22nd International Conference on Computer Communications and Networks (ICCCN)[C]. Bahamas: IEEE, 2013. 1 – 6

[9] On Foundation. Software-Defined Networking: The New Norm for Networks[OL]. https://www. opennetworking. org, 2013. 12.

[10] A Tootoonchian, Y Ganjali. Hyper Flow: A distributed control plane for Open Flow[A]. Proceedings of NM/WREN[C]. San Jose: USENIX, 2010. 3 – 3.

[11] T Koponen, M Casado, N Gude, J Stribling, L Poutievski, et al. Onix: a distributed control platform for large-scale production networks[A]. Proceedings of OSDI[C]. Berkeley: USENIX, 2010. 1 – 6.

[12] 姚琳元,陈颖,宋飞,等. 基于时延的软件定义网络快速响应控制器部署[J]. 电子与信息学报,2014,36(12):2802 – 2808.
Yao Lin-yuan, Chen Ying, Song Fei, et al. Delay-aware controller placement for fast response in soft-defined network[J]. Journal of Electronics & Information Technology, 2014, 36(12):2802 – 2808. (in Chinese)

[13] David Hock, Matthias Hartmann, Steffen Gebert, et al. Pareto-optimal resilient controller placement in sdn-based core networks[A]. Teletraffic Congress (ITC)[C]. IEEE, 2013. 1 – 9.

[14] 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.

[15] Boyang Zhou, Chunming Wu, Ming Jiang. Programming network via distributed control in software-defined networks[A]. Communications (ICC)[C]. Sydney: IEEE, 2014. 3051 – 3057

[16] Yan-nan HU, Wen-dong WANG, Xiang-yang GONG, et al. On the placement of controller in software-defined networks[J]. The Journal of China Universities of Posts and Telecommunications, 2012, 19(2):92 – 97.

[17] HE Egilmez, Tekalp AM. Distributed QoS architecturesformultimedia[J]. Multimedia, IEEE Transactions, 2014, 16(6):1597 – 1609.

[18] George Karypis, VipinKumar. Multilevel, k-way partitioning scheme for irregular graphs[J]. Journal of Parallel and Distributed Computing, 1998, 3(11):96 – 129.

[19] BWKernighan, S Lin. An efficient heuristic procedure for partitioning graphs[J]. Bell System Tech J, 1970, 49(2):291 – 307.

作者简介

张 栋 男,1981 年 10 月出生,福建福州人,福州大学数学与计算机科学学院副教授、硕士生导师. 2005 年、2010 年分别获浙江大学工学学士、工学博士学位. 主要从事网络虚拟化、软件定义网络、网络服务质量等方面的研究.
E-mail: zhangdong@ fzu. edu. cn

郭俊杰 男,1990 年 6 月出生,福建南平人,福州大学数学与计算机科学学院硕士研究生. 主要从事软件定义网络方面的研究.