一种考虑软件定义网络控制节点故障的控制器部署和交换机迁移方法

伊 鹏 刘邦舟* 王文博 张少军
(国家数字交换系统工程技术研究中心 郑州 450002)

摘 要：在软件定义网络(SDN)中，若控制节点发生不可恢复的故障，则相关交换机需向其他控制节点迁移，这将导致网络性能下降。针对这一问题，该文提出一种考虑 SDN 控制节点故障的控制器部署和交换机迁移方法。与现有算法仅优化交换机迁移方法不同，该方法同时考虑控制器部署位置的影响。首先利用标号传播算法构造备选子集并划分双层子集，然后在每个子集中选取合适位置部署控制器，最后为子集内节点分配相应的 master 和 slave 控制器。实验结果表明，与现有算法相比，所提方法可有效解决交换机迁移后控制器超载的问题；通过调整参数，权衡控制器故障前后网络性能，可明显改善交换机迁移后的控制链路平均时延。

关键词：软件定义网络；控制器部署；交换机迁移；控制节点故障
中图分类号： TP393.2 **文献标识码**：A **文章编号**：1009-5896(2017)08-1972-07
DOI: 10.11999/JEIT161216

Controller Placement and Switch Immigration Strategy for SDN Controller Failure

YI Peng LIU Bangzhou WANG Wenbo ZHANG Shaojun
(*National Digital Switching Engineering & Technological Research Center*, *Zhengzhou* 450002, *China*)

Abstract: In Software-Defined Networking (SDN), if a controller has unrecoverable failure, the related switches immigrate to other controllers, which degrades network performance. Concerning the above problem, a strategy of controller placement and switch immigration is proposed for controller failure. Different from the present algorithms which only optimize switch immigration method, the proposed strategy also considers the influence of controller placement. Firstly, Label Propagation Algorithm (LPA) is used to construct alternate domains set and partition bilayer domains. Then, one controller is placed in each domain on properly selected situation. Finally, the switches are assigned to corresponding master and slave controllers. The experimental results show that controller overloading problem is well solved compared with the present algorithms. Network performance before and after failure can be traded off by adjusting parameters, which decreases average control path latency after switch immigration.

Key words: Software-Defined Networking (SDN); Controller placement; Switch immigration; Controller failure

1 引言

软件定义网络(Software Defined Networking, SDN)[1]将控制平面和数据平面分离，方便网络的管理和维护，但同时也导致了交换机对控制器的依赖性。SDN 网络中的组件主要包括通信链路、交换机和控制器。其中，控制器故障的影响最大，若无法及时从控制器获取信息，所属交换机将无法处理流表中未指定的网络数据流[2]。为减小控制器单点故障造成的影响，当前网络多采用分布式架构[3,4]，通过部署多个控制器提高网络的可靠性。交换机可以在必要时在控制器间迁移，实现控制器的故障备份。如何保证交换机迁移后的网络性能已经成为 SDN 控制层面设计的一大挑战。

近年来，控制器故障恢复问题成为研究热点。如文献[5]提出一种故障检测和恢复机制 SPIDER，可以快速检测 SDN 节点、链路故障并实现重路由；文献[6]提出 ElastiCon，通过动态调整控制器的负载，保证交换机迁移前后控制器的负载均衡；文献[7]提出通过构造更鲁棒的转发树，增强控制平面的

收稿日期：2016-11-10；改回日期：2017-03-24；网络出版：2017-05-02
*通信作者：刘邦舟 liubangzhou@163.com
基金项目：国家 973 计划项目(2012CB315901, 2013CB329104)，国家自然科学基金(61572519, 61502530)，国家 863 计划项目(2013AA013505, 2015AA016102)
Foundation Items: The National 973 Program of China (2012CB315901, 2013CB329104), The National Natural Science Foundation of China (61572519, 61502530), The National 863 Program of China (2013AA013505, 2015AA016102)

可靠性；文献[8]提出两种计算控制器备份列表的方法，其一是根据最短距离确定备份控制器，其二是根据控制器剩余容量确定备份控制器；文献[9]将原交换机迁移问题优化成为控制器的热备份及选举问题，综合考虑信息交互、失联性、负载失衡和跨域通信等代价，改善控制器负载均衡程度和交换机跨域通信问题。

过往研究大多通过优化交换机的迁移方法，降低控制链路通信时延或保证控制器负载均衡，但忽视了控制器的部署位置这一因素。如果仅对交换机迁移方法进行优化，而不考虑控制器部署位置的合理性，将使交换机迁移策略受到限制，同时也使优化效果受到影响。对此，本文提出一种考虑 SDN 控制节点故障的控制器部署和交换机迁移方法(Controller Placement and Switch Immigration Strategy for SDN controller failure, CPSI strategy)。CPSI 主要分为 2 个步骤，步骤 1 为网络双层子集的划分，首先利用标号传播算法[10](Label Propagation Algorithm, LPA)构造备选子集，然后选取符合条件的子集构成双层子集。步骤 2 为控制器部署和备份，先在子集中选取合适位置部署控制器，再为节点分配相应的 master 和 slave 控制器。基于公共拓扑 Internet2[11]和 topology zoo[12]的仿真可发现，与现有算法相比，交换机迁移后的控制器超载问题得到明显改善；通过调整参数，可以有效地权衡控制器故障前后控制链路平均时延。

2　模型定义

控制器在网络中存在 equal, master 和 slave 3 种不同身份[13]。其中，equal 仅作为一种过渡状态；master 控制器对于每个交换机仅有 1 个，可独立控制交换机。Slave 控制器对交换机仅有读权限，因此可作为 master 的备份，当 master 故障时，所属交换机迁移到相应的 slave 上。应用场景如图 1，假设当控制器 $c1$ 发生的故障时，需根据备份设定，将交换机 $s1\sim s5$ 迁移至 $c2$, $s6\sim s9$ 迁移至 $c3$，此时 $c2$ 和 $c3$ 成为相应交换机的 master 控制器。

将网络建模成一个无向图 $G=(N,L)$，其中 $n\in N$ 表示网络节点；$l\in L$ 表示通信链路；将网络节点划分成 m 个子域 $D=(D_1,D_2,\cdots,D_m)$；每个子域可包含任意 p $(1\le p\le n)$ 个交换机 $V_i=\left(v_{t_1},v_{t_2},\cdots,v_{t_p}\right)$, 1 个 master 控制器 $s_i^{'}$ 和 q 个 slave 控制器 $s_1^{''}$, $s_2^{''},\cdots,s_q^{''}$，因此每个子域的节点集合为 $D_i=\left(v_{t_1},v_{t_2},\cdots,v_{t_p},s_i^{'},s_{i_1}^{''},s_{i_2}^{''},\cdots,s_{i_q}^{''}\right)$；整个网络共包含 m 个控制器 $S=(s_1,s_2,\cdots,s_m)$；任意两个子域间互不重叠，并且任意交换机只属于某一个子域。

定义 1　交换机与 master 控制器连接矩阵 $\boldsymbol{X}$。

交换机与 master 控制器连接矩阵 $\boldsymbol{X}=[x_{i,j}]_{n\times n}$，表示交换机和相应的 master 之间的所属关系，若 $x_{i,j}=1$，则交换机 v_i 受到控制器 $s_j^{'}$ 的控制，反之则没有。

定义 2　交换机迁移矩阵 $\boldsymbol{A}$。交换机迁移矩阵 $\boldsymbol{A}_k=\left[a_{i,j}^k\right]_{n\times n}$，表示交换机 v_k 的控制器备份状况。若 $a_{i,j}^k=1$，则表示控制器 s_i 故障时交换机 v_k 可由 master s_i 迁移至 slave s_j，同时 slave s_j 升级为 master s_j。

定义 3　相邻节点间直线距离 $d(v_i,v_j)$。对于大规模网络，两点间距离需要考虑地面弧度。假设地球是半径为 r 的球体，节点 v_i 的经纬度分别为 α_i 和 β_i；且规定计算经度时，东经为正，西经为负；计算纬度时，南纬为 90° +纬度值，北纬为 90° – 纬度值，则相邻两点 (v_i,v_j) 间直线距离为

$$d\left(v_i,v_j\right)=r\cdot\arccos\left(\sin\left(\beta_i\right)\sin\left(\beta_j\right)\cos\left(\alpha_i-\alpha_j\right)+\cos\left(\beta_i\right)\cos\left(\beta_j\right)\right)\tag{1}$$

定义 4　控制链路平均时延 L_{avg}。大规模网络中节点间距离较远，传输距离是影响信号传播时延

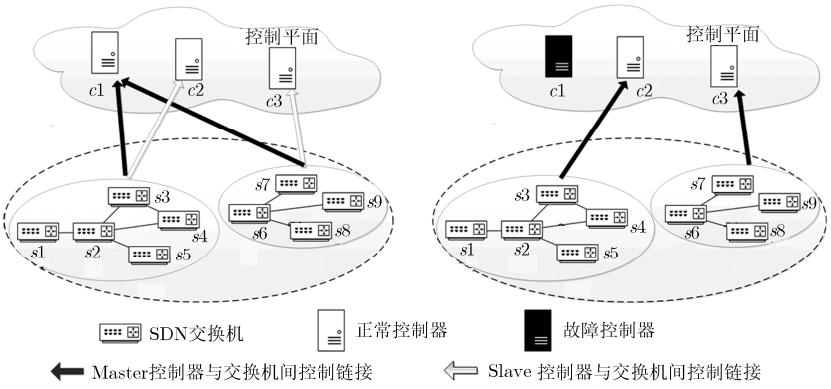


图 1 应用场景示意图

的主要因素，可用节点间距离衡量控制链路消息的传播时延。已知连接矩阵 $\boldsymbol{X}$，设式(4)中 $D(v_i,v_j)$ 为任意两节点间最短路径的长度，则控制链路平均时延 L_{avg} 可用网络中所有交换机 $v_i \in N$ 到相应控制器 $v_j \in S$ 最短路径长度的平均值表示：

$$D\left(v_i,v_j\right)=\min_{v_1,\cdots,v_k}\left(d\left(v_i,v_1\right)+d\left(v_1,v_2\right)+\cdots+d\left(v_{k-1},v_k\right)+d\left(v_k,v_j\right)\right) \tag{2}$$

$$L_{\text{avg}}\left(N,S,\boldsymbol{X}\right)=\frac{1}{n}\sum_{v_i\in N}\sum_{v_j\in S}D(v_i,v_j)\cdot x_{i,j} \tag{3}$$

定义 5 控制器容量冗余 R_q。 设交换机 v_p 的负载为 l_p，控制器 s_q 的容量为 C_q。网络中某一控制器发生故障，所属交换机迁移，给被迁移控制器增加额外的负载压力，为防止被迁移控制器超载，发生相继故障，控制器需要保留一定的容量冗余 R_q：

$$L_q=\left[l_1,l_2,\cdots,l_n\right]\cdot\boldsymbol{X}_{:,q} \tag{4}$$

$$R_q\left(\boldsymbol{X}\right)=C_q-L_q \tag{5}$$

针对控制器故障，交换机迁移后的网络性能优化问题，本文提出如下模型：

$$\min_{\boldsymbol{X},\boldsymbol{A}}\quad L_{\text{avg}}\left(N,S,\boldsymbol{X}'\right) \tag{6}$$

$$\text{s.t.}\quad \boldsymbol{X}'_{k,:}=\boldsymbol{X}_{k,:}\cdot\boldsymbol{A}_k;\forall k \tag{7}$$

$$L_{\text{avg}}\left(N,S,\boldsymbol{X}\right)<T \tag{8}$$

$$R_q\left(\boldsymbol{X}'\right)\geq 0,\ R_q\left(\boldsymbol{X}\right)\geq 0\ \forall q \tag{9}$$

$$\sum_j x'_{i,j}=1,\ \sum_j x_{i,j}=1\ \forall i \tag{10}$$

$$x'_{i,j}\in\{0,1\},\ x_{i,j}\in\{0,1\}\ \forall i,j \tag{11}$$

式(6)是模型的目标函数，最小化交换机迁移后网络的控制链路平均时延，其中决策变量为控制器连接矩阵 $\boldsymbol{X}$ 和交换机迁移矩阵 $\boldsymbol{A}$；式(7)给出了 $\boldsymbol{X}$、$\boldsymbol{A}$ 和 $\boldsymbol{X}'$ 三者间的函数关系，其中 $\boldsymbol{X}'$ 为交换机迁移后的连接矩阵；式(8)通过限制控制器故障前的控制链路平均时延，可实现对控制器故障前后控制链路平均时延的有效权衡；式(9)保证在控制器不发生超载；式(10)保证网络中的交换机在任何情况下都只分配给 1 个 master 控制器；式(11)约束了连接状态矩阵的值，保证其只存在 1 或 0，即连接或不连接两种状态。

3 算法分析

3.1 算法描述

本节介绍 CPSI 算法，分为 2 个步骤：一是双层子集的划分(表 1)；二是控制器的部署和分配(表 2)。此处引入子集的概念，本质上子集与子域相同，但为区分算法过程中使用的中间变量和最终的交换机划分结果，本文将 CPSI 算法步骤中出现的交换机集合命名为子集，而将最终结果中的交换机集合命名为子域。

步骤 1 CPSI 算法主要流程如下。利用 LPA 算法[10]构造备选子集，选取控制链路平均时延最小的子集构造双层子集。LPA 算法计算复杂度与网络规模仅呈线性关系，因此利用 LPA 可在有限时间内构造足够规模的备选子集。为控制两层子集间的相似度，现定义子集搜索范围参数 $w(0<w\leq 1)$ 和子集平均相似度 Sim_{avg}。其中，w 限制备选子集的选取范围；Sim_{avg} 衡量任一子集与相邻子集间的平均相似度，其中 c_1 和 c_2 表示重叠部分分别占两个子集的比例，$\text{Sim}\left(D_i,D_j\right)$ 表示任意两个子集间的相似度：

$$c_1=\frac{\left|D_i\cap D_j\right|}{\left|D_i\right|},\quad c_2=\frac{\left|D_i\cap D_j\right|}{\left|D_j\right|} \tag{12}$$

$$\text{Sim}\left(D_i,D_j\right)=c_1+c_2-c_1\cdot c_2 \tag{13}$$

$$\text{Sim}_{\text{avg}}\left(D_i\right)=\sum_{j=1}^{m}\text{Sim}\left(D_i,D_j\right)\Big/m \tag{14}$$

算法首先利用 LPA 算法，将网络划分成多个子集(行 2)，并选取负载不超过控制器最大容量的子集加入备选子集(行 3~4)。重复这一过程至备选子集达到设定的规模；然后将控制器按照容量大小降序排列(行 9)，并依次循环选取(行 11)，优先部署大容量控制器；将备选子集中的子集按平均相似度的大小升序排列(行 12)，并在前 w 元素中选取控制链路平均时延最小且负载尽可能大的子集加入双层子集(行 13~14)；将已经被包含两次的节点从备选子集和待选节点集合中删除(行 17~21)，保证每个节点只属于两个子集；重复以上步骤至所有的节点都已经被包含两次(行 10)；最后输出生成的双层子集划分结果(行 23)。如表 1 所示。

步骤 2 在子集中搜索使控制链路平均时延最小的位置来部署控制器(行 2)，从而保证交换机迁移后网络的性能；由于每个节点同时属于两个子集(行 6)，故分别计算到这两个子集中控制器的最短距离，选取其中距离近的控制器作为 master，距离远的作为 slave(行 7~11)，遍历网络中的所有节点(行 5)，完成控制器的分配；最后根据 master 和 slave 的分配情况构造连接矩阵 $\boldsymbol{X}$ 以及交换机迁移矩阵 $\boldsymbol{A}$ (行 13)。如表 2 所示。

3.2 复杂度分析

步骤 1 中，行 2 基于 LPA 算法对网络进行子域划分，其时间复杂度为 $O(n+m)$[10]，行 3~7 的最大循环次数为 n，行 1~8 的最大循环次数为 M，则行

表 1 划分双层子集

步骤 1　划分双层子集
输入： 网络拓扑图 $G=(N,L)$，交换机负载 l_p，控制器容量 C_q
备选子集规模 M，子集搜索范围参数 w
输出： 双层子集划分结果 D

while $\text{size}(D\_\text{set})<M$ **do**
  $\{D_1,D_2,\cdots,D_k\}\leftarrow \text{LPA}(G=(N,L))$
  for each $D_i\in\{D_1,D_2,\cdots,D_k\}$ **do**
    if $\text{load}(D_i)<\max\limits_q(C_q)$
      $D\_\text{set}=D\_\text{set}\bigcup D_i$
    end
  end
end while
C_sort $=\text{sort}(C,\text{'decrease'})$
while $N\neq\varnothing$ **do**
  $C_q=C\_\text{sort}\rightarrow\text{next}$
  $D\_\text{set}=\text{sort}(D\_\text{set},\text{'increase'})$ by Sim_{avg}
  $D\_\text{set}'=D\_\text{set}\left[1:\text{size}(D\_\text{set})/w\right]$
  select $D_i\in\left\{D_i\middle|\text{load}(D_i)=\max(\text{load}(D_i))\le C_q,D_i\in D\_\text{set}'\right\}$ that minimizes $L_{\text{avg}}(D_i)$
  $D=D\bigcup D_i$
  $D\_\text{set}=D\_\text{set}-D_i$
  for each $n\in N$ **do**
    if count$(n$ in $D)=2$
      $N=N-n$, $D\_\text{set}=D\_\text{set}-n$
    end
  end
end
output $D=(D_1,D_2,\cdots,D_m)$

表 2 控制器的部署及分配

步骤 2　控制器部署及分配
输入： 网络拓扑图 $G=(N,L)$，拓扑两层子集划分结果 D
输出： 交换机与 master 控制器的连接矩阵 $\boldsymbol{X}$，交换机迁移矩阵 $\boldsymbol{A}$

for each $D_i\in D$ **do**
  select $s_i\in D_i$ that minimizes $L_{\text{avg}}(D_i,s_i)$
  $S\leftarrow S\bigcup s_i$
end
for each $n\in N$ **do**
  find D_i and D_j that include n
  if $d(n,s_i\in D_i)<d(n,s_j\in D_j)$
    s_i is master to n, s_j is slave to n
  else
    s_j is master to n, s_i is slave to n
  end
end
construct $\boldsymbol{X}$ and $\boldsymbol{A}$

1~8 的时间复杂度为 $O(M(n+m))$；行 9 排序的时间复杂度为 $O(m\log_2 m)$；行 12 计算平均相似度并排序的时间复杂度为 $O(M^2)$，行 14 的时间复杂度为 $O(M)$，行 17~22 n 次循环的时间复杂度为 $O(n)$，行 10~22 的最大循环次数为 M，其时间复杂度为 $O\left(M\left(M^2+n\right)\right)$。故步骤 1 的整体时间复杂度为 $O\left(M\left(M^2+n\right)\right)$。步骤 2 中，行 1~4 分别在每个子集中选取合适位置部署控制器，时间复杂度为 $O\left(n^2\right)$；行 5~12 分别为每个节点分配控制器，时间复杂度为 $O(n)$，故步骤 2 的整体时间复杂度为 $O\left(n^2\right)$。综上，CPSI 算法的时间复杂度为 $O\left(M^3+n^2+Mn\right)$。

4 实验结果分析

4.1 实验环境及参数

针对本文实验的软硬件环境及相关参数设定，做出如下说明：

(1)硬件环境为基于 Intel Core i7-3770 CPU 3.40 GHz, RAM 4 GB 的个人台式电脑；相关算法均在 MATLAB 2013 软件上进行模拟仿真。

(2)为简化实验过程，在不影响实验结果的前提下，做出以下设定：每个交换机的负载大小仅影响其所在子域的负载，为其分配对应容量的控制器即可，对算法其他部分的计算无影响，因此可设所有交换机的负载均为 $l_p=1$；控制器容量仅影响子域与控制器的分配，对算法其他部分的计算无影响。因此，为保证任一控制器的故障均不会造成超载问题，所有控制器的总容量至少为当前网络总负载的 2 倍，故实验中可设每个控制器的容量均为 $C_q=2n/k$。交换机负载和控制器容量在实际应用中可根据实际情况具体设定。

4.2 仿真分析

本文将 CPSI 算法和其他几种算法进行比较，由于直接研究此方向的文献较少，故借鉴了相关的控制器部署算法和交换机迁移算法。文献[4,14]分别提出了两种控制器部署方案，其中文献[4]提出了最小切算法(Min-Cut Algorithm, MCA)，将网络划分成多个子域，使得子域边界上的通信链路切数最小；文献[14]提出最小 K 中心算法(Minimum K-median Algorithm, MKA)，遍历所有可选节点，选取其中 K 个节点部署控制器，并将交换机连接至距离最近的控制器。文献[8]提出了 2 种交换机迁移方案，一是基于近邻的启发式算法(Proximity-Based Heuristic, PBH)，交换机迁移至时延最短的控制器；

二是基于剩余容量的启发式算法(Residual Capacity-Based Heuristic, RCBH)，交换机迁移至容量最大的控制器。

基于以上文献，本文设计了 4 种对照算法，最小切—近邻算法(MC-PB)，最小 K 中心—近邻算法(MK-PB)，最小切—剩余容量算法(MC-RCB)和最小 K 中心—剩余容量算法(MK-RCB)，分别对应两种控制器部署方案和两种交换机迁移方案构成的 4 种不同组合。

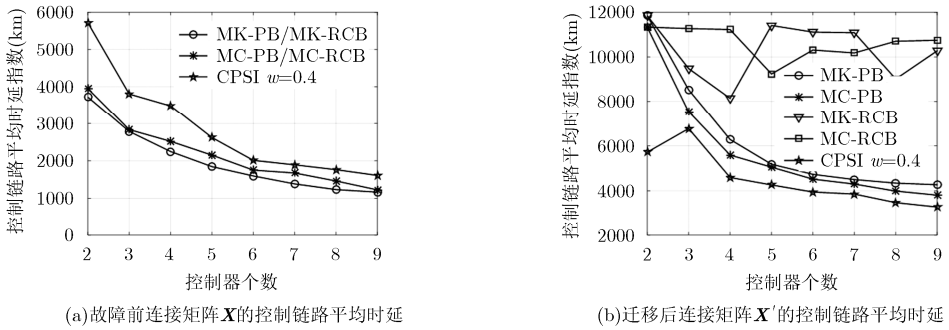
首先，针对交换机迁移后控制器负载的问题，在拓扑 Internet2, Geant2009 和 Canerie 上分别对上述 5 种算法进行研究。表 3 中 k 为控制器的数量，表中数据为当网络中任一控制器发生故障后，其余控制器超载数量之和。CPSI 算法中限制单个子域的最大负载(控制器作为 master 和 slave 连接的交换机负载总和)不超过控制器容量，因此不会发生超载状况。采用 RCBH 迁移算法的 MC-RCB 算法和 MK-RCB 算法会优先将交换机迁移至剩余容量最大的控制器，因此和 CPSI 算法一样都可以有效地解决交换机迁移后控制器超载的问题。但采用 PBH 迁移算法的 MC-PB 算法和 MK-PB 算法由于未考虑控制器容量，故某些控制器故障将导致相邻的被迁移控制器超载。

然后，基于 Internet2 拓扑比较 5 种算法的控制链路平均时延，其中 CPSI 算法选取 $w = 0.4$，保证故障前后网络性能相对均衡。图 2(a)为基于连接矩阵 $\boldsymbol{X}$ 的控制链路平均时延，通过比较可以发现，采用 MKA 算法的 MK-PB 和 MK-RCB 算法性能最好，这是由于 MKA 穷举了控制器部署的所有可能情况，并选择了最优解，但此算法带来巨大的运算量，不适用于复杂的网络。MCA 是一种启发式算法，以性能为代价，降低了运算量。CPSI 算法在此方面的性能略差，这是因为 CPSI 通过调整参数 w，权衡控制器故障前后的控制链路平均时延，通过牺牲一定故障前的网络性能，换取交换机迁移后网络性能的提升。

图 2(b)为基于迁移后连接矩阵 $\boldsymbol{X}'$ 的控制链路平均时延。MC-RCB 和 MK-RCB 算法因为只考虑了控制器的容量，而没有考虑距离因素，使得交换机可能被迁移到通信距离较远的控制器，导致平均时延远高于其他算法。MC-PB 和 MK-PB 算法在交换机迁移时选择最近的控制器，在一定程度上降低了平均时延，但是受到控制器部署位置的限制，使得优化的效果有限。CPSI 算法性能最佳，与其余 4 种算法中效果最好的 MC-PB 算法相比，其交换机迁移后的控制链路平均时延最大下降 49.5%，平均下降 16.4%。因为 CPSI 算法将网络划分成双层子集，每个节点同时属于两个子集，故交换机迁移后仍然属于其中一个子集。利用 LPA 构造备选子集，优化子集内的连通性，有效保证了迁移后控制链路平均时延。

表 3 超载控制器的数量

拓扑名	Internet2					Geant2009					Canerie				
	k														
	4	5	6	7	8	4	5	6	7	8	4	5	6	7	8
CPSI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC-RCB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MK-RCB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC-PB	0	1	1	0	1	1	1	1	2	0	1	0	2	1	1
MK-PB	1	2	1	0	2	1	1	0	1	1	1	1	1	2	2

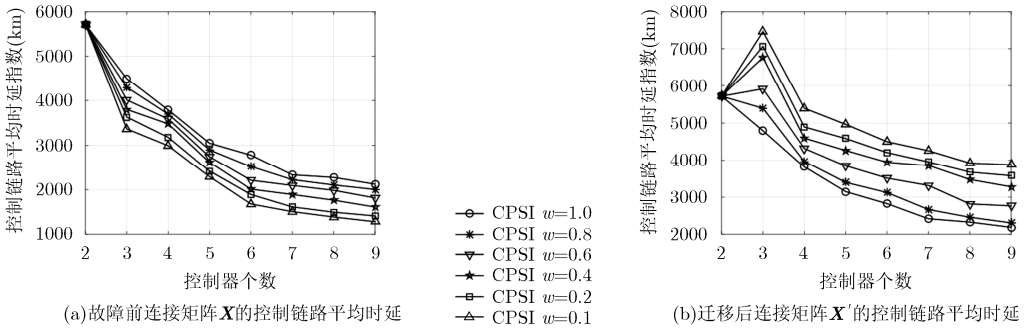


(a)故障前连接矩阵$\boldsymbol{X}$的控制链路平均时延　(b)迁移后连接矩阵$\boldsymbol{X}'$的控制链路平均时延

图 2 控制链路平均时延的对比

接下来，基于 Internet2 拓扑，比较不同 w 对算法结果的影响。图 3(a)和图 3(b)分别为不同 w 下，控制器故障前后网络的控制链路平均时延。通过比较可以发现，在相同控制器数量的前提下，w 越大，则故障前控制链路平均时延越大，而迁移后控制链路平均时延越小。这是由于 w 增大，导致选择备选子集时对子集相似度的限制松弛，使得子集间相似度更高，不同控制器的部署位置更接近。控制器部署位置相对集中，会导致处于拓扑边缘的节点远离控制器，从而增加整体的控制链路平均时延，但同时也保证控制器故障时交换机迁移距离更短，减小了迁移后控制链路平均时延。反之 w 减小，则会有相反的效果。因此，通过改变 w，可以在一定程度上权衡故障前后的控制链路平均时延。

最后，利用 Internet2 和在 topology zoo 中选取的 20 个拓扑对 CPSI 算法进一步验证。横坐标为实验拓扑的名称。计算当控制器个数分别为 $\lceil n/15\rceil \sim \lceil n/5\rceil$ (n 为拓扑节点数)时，上述 5 种算法基于交换机迁移后连接矩阵 $\boldsymbol{X}'$ 的控制链路平均时延的均值 $\overline{L}=\{\bar{l}_{\text{MK-PB}},\bar{l}_{\text{MC-PB}},\bar{l}_{\text{MK-RCB}},\bar{l}_{\text{MC-RCB}},\bar{l}_{\text{CPSI}}\}$，并做归一化处理 $\overline{L}'=\overline{L}/\max(\overline{L})$。实验结果如图 4，对于所选取的 21 个拓扑，CPSI 算法的平均时延都低于其余 4 种算法，对交换机迁移后控制链路时延的优化效果明显，说明了算法在不同拓扑上的适用性。



(a)故障前连接矩阵$\boldsymbol{X}$的控制链路平均时延　　(b)迁移后连接矩阵$\boldsymbol{X}'$的控制链路平均时延

图 3 比较 w 对 CPSI 控制链路平均时延的影响

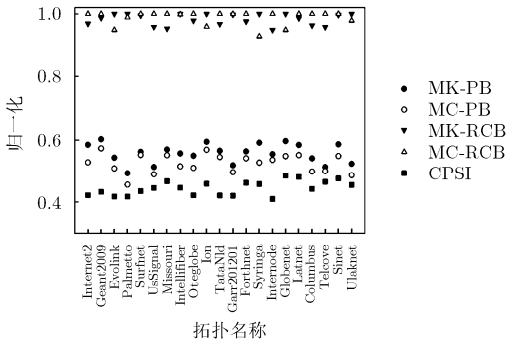


图 4 不同拓扑下基于迁移后连接矩阵 $\boldsymbol{X}'$ 的归一化控制链路平均时延

5　结束语

本文针对 SDN 控制器故障，交换机迁移，导致网络性能下降的问题，提出一种考虑控制节点故障的控制器部署和交换机迁移方法。与传统的解决思路不同，CPSI 算法不仅对交换机迁移方法进行优化，还对控制器部署策略进行了相应的优化。与现有算法相比，CPSI 算法通过权衡控制器故障前后的网络性能明显改善了交换机迁移后控制链路平均时延，同时有效解决了交换机迁移导致的控制器超载问题。本文在 SDN 控制节点故障恢复问题中考虑控制器部署位置的影响是一种新的尝试，为解决此类问题提供了新思路。

参 考 文 献

[1] RAWAT D B, DANDA B, and REDDY R. Software defined networking architecture, security and energy efficiency: A survey[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(1): 325–346. doi: 10.1109/COMST.2016.2618874.

[2] VISSICCHIO S, VANBEVER L, and BONAVENTURE O. Opportunities and research challenges of hybrid software defined networks[J]. *ACM SIGCOMM Computer Communication Review*, 2014, 44(2): 70–75.

[3] HASSA S YEGANEH S, and GANJALI Y. Kandoo: A framework for efficient and scalable offloading of control applications[C]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks, New York, NY, USA, 2012: 19–24.

[4] ZHANG Y, BEHESHTI N, and TATIPAMULA M. On resilience of split-architecture networks[C]. Global Telecommunications Conference (GLOBECOM 2011), Houston, TX, USA, 2011: 1–6.

[5] CASCONE C, POLLINI L, SANVITO D, *et al*. SPIDER: Fault resilient SDN pipeline with recovery delay guarantees[C]. 2016 IEEE NetSoft Conference and

Workshops (NetSoft), Seoul, Korea, 2016: 296–302.

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

[7] PENG Yuhuai, GONG Xiaoxue, GUO Lei, *et al*. A survivability routing mechanism in SDN enabled wireless mesh networks: Design and evaluation[J]. *China Communications*, 2016, 3(7): 32–38. doi: 10.1109/CC.2016.7559073.

[8] MULLER L F, OLIVEIRA R R, LUIZELLI M C, *et al*. Survivor: An enhanced controller placement strategy for improving SDN survivability[C]. IEEE Global Communications Conference, Austin, TX, USA, 2014: 1909–1915.

[9] 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. doi: 10.3969/j.issn.0372-2112.2016.04.023.

WANG Wenbo, WANG Binqiang, CHEN Feiyu, *et al*. The controller hot backup and election algorithms in software defined networks[J]. *Acta Electronica Sinica*, 2016, 44(4): 913–919.

[10] 刘邦舟, 汪斌强, 王文博, 等. 针对大规模软件定义网络的子域划分及控制器部署方法[J]. 计算机应用, 2016, 36(12): 3239–3243. doi: 10.11772/j.issn.1001-9081.2016.12.3239.

LIU Bangzhou, WANG Binqiang, WANG Wenbo, *et al*. Domain partition and controller placement for large scale software defined network[J]. *Journal of Computer Applications*, 2016, 36(12): 3239–3243. doi: 10.11772/j.issn.1001-9081.2016.12.3239.

[11] Internet2 Open Science, Scholarship and Services Exchange [OL]. http://www.internet2.edu/, 2016.10.

[12] 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.

[13] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, *et al*. OpenFlow: Enabling innovation in campus networks[J]. *ACM SIGCOMM Computer Communication Review*, 2008, 38(2): 69–74. doi: 10.1145/1355734.1355746.

[14] HELLER B, SHERWOOD R, and MCKEOWN N. The controller placement problem[C]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks. New York, NY, USA, 2012: 7–12.

伊 鹏： 男，1977 年生，教授，研究方向为宽带信息网、可重构柔性网络.

刘邦舟： 男，1992 年生，硕士生，研究方向为 SDN 控制器部署.

王文博： 男，1991 年生，硕士，研究方向为 SDN 弹性控制.

张少军： 男，1989 年生，博士生，研究方向为 SDN 控制平面可扩展性.