基于最小代价路径的交换机迁移方法研究

赖英旭[1,2]，蒲叶玮[1]，刘静[1,3]

（1. 北京工业大学信息学部，北京 100124；2. 信息保障技术重点实验室，北京 100072；
3. 西安电子科技大学陕西省网络与系统安全重点实验室，陕西 西安 710071）

摘 要：针对如何保护控制器，尤其是骨干控制器免受安全威胁与攻击，提高 SDN 控制平面的安全性，提出一种基于最小代价路径的交换机迁移算法。在迁移模型中加入负载预测模块，预测模块执行控制器负载预测算法，得到负载预测矩阵，然后根据负载预测矩阵确定迁出、目标控制器集合。利用改进的迪杰斯特拉算法确定最小代价路径，根据控制器的负载状态和待迁移交换机的流量优先级，在最小代价路径中确定最优迁移交换机集合，同时针对迁移过程中可能产生的孤立节点问题给出了解决方案。实验结果表明，所提算法确定的迁移触发时机、迁出控制器和目标控制器更加合理，减少了迁移次数和代价，增强了控制器的安全性，提高了控制器性能。

关键词：软件定义网络；迪杰斯特拉算法；负载均衡；负载预测；交换机迁移

中图分类号：TP309

文献标识码：A

doi:10.11959/j.issn.1000−436x.2020030

Research on switch migration method based on minimum cost path

LAI Yingxu[1,2], PU Yewei[1], LIU Jing[1,3]

1. Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
2. Science and Technology on Information Assurance Laboratory, Beijing 100072, China
3. Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China

Abstract: In order to protect the controller, especially the controller in backbone network, from security threats and attacks, improve the security of the software-defined network (SDN) control plane, a switch migration algorithm based on minimum cost path was proposed. A load prediction module was added to the migration model, which executed a controller load prediction algorithm to obtain a load prediction matrix, and then a migration-target controller set was determined according to the load prediction matrix. The improved Dijkstra algorithm was used to determine the minimum cost path. According to the load state of the controller and the traffic priority of the switch to be migrated, the optimal migration switch set was determined. The problem of isolated nodes was solved that may occur during the migration process. The experimental results show that the migration timing of the algorithm is more reasonable, the selection of the migration controller and the target controller is more reasonable, the load balancing of the control plane is realized, the number of migrations and cost are reduced, and the performance of the controller is improved.

Key words: software-defined network, Dijkstra algorithm, load balancing, load forecasting, switch migration

---

收稿日期：2019−09−17；**修回日期**：2019−12−17

基金项目：国家自然科学基金资助项目（No.61872015）；北京市自然科学基金资助项目（No.19L2020）；青海省自然科学基金资助项目（No.2017-ZJ-912）；信息保障技术重点实验室基金资助项目（No.614211204031117）；陕西省网络与系统安全重点实验室开放课题基金资助项目（No.NSSOF1900105）；工业和信息化部 2018 年工业互联网创新发展工程基金资助项目（面向电子行业安全技术典型应用推广项目）

Foundation Items: The National Natural Science Foundation of China (No.61872015), The Natural Science Foundation of Beijing (No.19L2020), The Natural Science Foundation of Qinghai Province (No.2017-ZJ-912), The Foundation of Science and Technology on Information Assurance Laboratory (No.614211204031117), The Foundation of Shaanxi Key Laboratory of Network and System Security (No.NSSOF1900105), Industrial Internet Innovation and Development Project (Typical Application and Promotion Project of the Security Technology for the Electronics Industry) of the Ministry of Industry and Information Technology of China in 2018

1 引言

软件定义网络（SDN, software-defined network）将网络的转发和控制功能分离，更加方便地实现对网络的控制，具有比传统网络更灵活的优势，所以被业界用来解决传统网络结构僵化问题[1-2]。但当流量出现激增或者其他变化时，控制器与交换机之间无法迅速地适应这种负载改变，尤其是 SDN 部署在大规模网络时，会导致控制平面负载不均衡，控制器也更容易受到安全威胁[3]。

如何保护控制器，尤其是骨干控制器免受安全威胁与攻击，提高 SDN 控制平面的安全性，是一个非常重要的课题[4]。在 SDN 中，根据 OpenFlow 协议的描述，当流量到达时，若交换机的流表没有条目可以匹配该流量，则交换机将向控制器发送 Packet_In 数据分组来请求路由信息。然而该机制可能被攻击者利用，向控制器发起攻击，令控制器过载，导致系统崩溃，甚至导致控制器的级联故障[5]。

骨干控制器是 SDN 的核心，会接收到更多来自交换机的请求，处理大量的消息，这使它们更容易被恶意攻击者识别或被恶意流量吞没。为了保护骨干控制器，使其难以被识别，通过交换机迁移算法，将骨干控制器的流量迁移到其他控制器，降低骨干控制器被识别、被攻击的可能性，同时使控制器之间的负载尽可能达到均衡。

本文的主要贡献如下。

1) 为了减少骨干控制器被识别的可能性，同时减少迁移交换机所产生的代价，本文提出基于最小代价路径的交换机迁移算法（SMCP, switch migration algorithm based on minimum cost path）。通过改进的迪杰斯特拉算法获取最小代价迁移路径，根据控制器的负载状态，以及待迁移交换机的流量优先级来确定最优迁移交换机集合。本文算法可以确保迁移后控制平面负载有较好的均衡性，减少骨干控制器被监听、被识别的可能性，减少了迁移代价，同时保证重要流量被优先处理。

2) 本文提出的 SMCP 通过减少迁移次数来进一步降低迁移所产生的代价。对于过载控制器，采用一次均衡方法来降低其负载，在之后的每个均衡过程，本文算法都只进行一次迁移操作，减少了迁移次数，降低了迁移代价，从而提高了系统的稳定性。

3) 为了减少频繁迁移，规避不必要迁移操作，本文的迁移模型中加入了负载预测模块。传统方法仅通过控制器阈值来确定是否迁移，一旦负载值大于阈值就执行迁移，这样会产生大量的迁移代价并使系统的稳定性降低。比如控制器在当前时刻是超过阈值的，但是下一时刻控制器流量回归正常，此时的迁移是不必要的，并且会产生相应的不必要的代价。负载预测模块可以预测下一时刻的负载情况，从而确定目标控制器以及迁移触发时机，减少不必要的迁移，保证系统稳定性。

4) 本文针对迁移过程中可能产生的孤立节点问题给出了解决方案。迁移某个或者某些交换机时可能会造成孤立节点问题，孤立节点内的交换机会同时请求本域和目标域控制器，给 2 个控制器带来负担，降低控制器性能。本文提出孤立节点处理算法，解决迁移过程中出现的孤立节点问题，减少控制器响应时间。

2 相关研究

传统的 SDN 实现依赖于集中式控制器，并且具有与性能和可伸缩性相关的若干限制。相关研究工作指出，部署分布式控制器是一种解决问题的有效方法[6-7]。

为了提高 SDN 中控制平面的性能，应该考虑多方面的问题，例如，最大化控制器的性能，把控制器部分工作移交到转发设备，使一组控制器节点能够实现分布式控制平面。这些工作提出了一个逻辑上集中的控制平面，并着手解决全局视图和状态分布式控制平面的一致性，实现更好的可扩展性，但当大量的不均匀流量到达这些分布式控制器时，这种方法会导致控制器之间的负载不平衡[8]。

Cheng 等[9]提出了一种博弈决策机制，用于将交换机从过载控制器迁移到空闲控制器，解决了带约束条件的资源利用率最大化问题，但缺乏博弈触发机制，并且整个过程会额外产生较大的网络开销。Liang 等[10]通过构建控制器集群的交换机迁移机制来均衡控制器负载，动态地迁移交换机，实现控制器间的负载转移，但是该操作导致控制器响应时间增加。在分布式最近迁移算法[11]中，选择物理位置最近的控制器作为目标控制器，方便操作，不过该方法很可能带来新的负载不平衡。在最大化资源利用率迁移算法（MUMA, maximizing resource utilization migration algorithm）中[12]，当发生负载不平衡时，控制器随机选择一个交换机用于迁移，不过算法没有考虑负载均衡性，迁移后可能产生新

的过载情况。Wang 等[13]以提高迁移效率为目的，基于贪婪算法，通过构建迁移效率模型对迁移成本和负载平衡率进行优化。Ye 等[14]提出利用资源消耗模型来解决交换机迁移问题，将交换机看作资源消耗者，控制器作为资源提供者，用马尔可夫链求解该问题，能达到较高的负载均衡率，但是由于待迁移的交换机及目标控制器都是随机选择的，导致算法的效率较低。Li 等[15]提出了一种基于模糊多目标粒子群算法的动态切换迁移策略，通过求解多目标优化问题来解决控制器平面的负载均衡问题，算法综合考虑了代价、效率与均衡性，但算法的运算量和数据的存储量都很大，对网络本身的稳定性也有一定的负面影响。Kim 等[16]提出了一种称为渐进和声搜索（P-HS, progressive-harmony search）的启发式算法，用于控制平面的负载均衡，解决了分布式 SDN 控制器环境中的动态控制器供应问题，算法可以最大程度地减少交换机与控制器之间的通信时延，不过迁移过程所消耗的代价、迁移效率以及迁移后造成的新问题并没有考虑全面。

通过调查国内外的研究发现，控制平面负载不均衡问题普遍采用交换机迁移的策略来解决，但是迁移时机以及迁入迁出控制器的选择比较僵硬，大部分迁移算法都是超过固定的阈值后，选择负载较大的控制器下的交换机进行迁移，然后判断迁移后的过载控制器是否仍然过载，不断进行迁移操作直至恢复正常，虽然单次迁移效率较高，但是对于传输速率较大的网络来说，在迁移过程中容易产生更大的网络代价。对于控制器负载在当前时刻超过阈值，但是下一时刻回归正常，这种情况的迁移是不必要的，并且未考虑迁移后所产生的孤立节点对网络的影响，反而加重了控制器负担，使迁移策略并不完善。

为了解决上述问题，本文在迁移方法中加入了负载预测算法，从而解决迁移时机以及迁入迁出控制器的选择僵硬问题，通过预测控制器下一时刻的负载情况从而确定迁移触发时机和迁出控制器、目标控制器集合，减少频繁的迁移，提高网络的稳定性。通过交换机集合迁移的方式，减少迁移次数，缩短了迁移所花费的时间，同时划分流量优先级，保证重要流可以优先处理。通过孤立节点处理算法，解决迁移过程中出现的孤立节点问题，减少交换机跨域请求，提高控制器响应时间。

3　问题描述与迁移模型建模

本文研究背景是分布式架构下的 SDN。在 SDN 中，逻辑集中控制器（决策者）通过向交换机发送带有处理策略的数据分组来管理数据平面。由于在一定时间内流量具有不稳定性和不均衡特性，当控制器域内交换机流量突然增加时，会影响控制器对请求的响应，造成控制器过载。

3.1　问题描述

分布式多控制器架构如图 1 所示。当更多的交换机与控制器 B 连接，或控制器 B 聚合了大量的流请求时，控制器 B 处于负载过重状态。根据 OpenFlow1.3 协议的定义，如果交换机的主控制器处于过载或低效状态，此交换机可以从控制平面中选择一个其他控制器作为新的主控制器，因此该过程被描述为交换机迁移[17]。

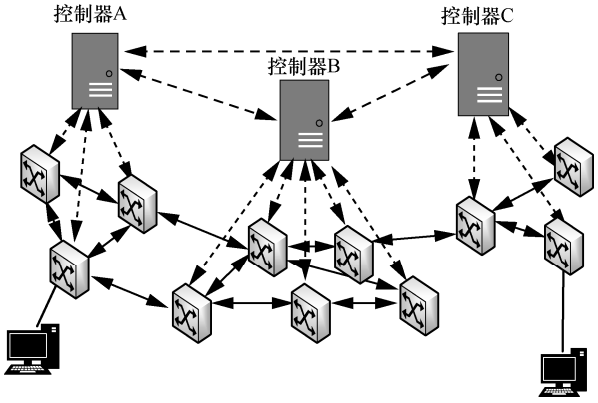

图 1　分布式多控制器架构

3.2　符号和定义

整个网络用图 $G=(V,E)$ 来表示，其中 V 和 E 分别表示节点集合和链路集合。控制器集合用 C 表示，交换机集合用 S 表示。$\boldsymbol{\Omega}=[x_{ij}]_{M\times N}$ 是所有网络元素的连接矩阵，其中 M 和 N 表示控制器和交换机个数，x_{ij} 表示交换机 S_i 和控制器 C_j 之间的映射关系，如式(1)所示。

$$x_{ij}=\begin{cases}1,\text{交换机}S_i\text{在控制器}C_j\text{连接下}\\0,\text{交换机}S_i\text{不在控制器}C_j\text{连接下}\end{cases} \tag{1}$$

1) 控制器的负载值定义为控制器接收来自其域内交换机 Packet_In 的数量，如式(2)所示。

$$\mathrm{LC}_j=\sum_{i=1}^{m}P_{S_iC_j} \tag{2}$$

其中，LC_j 表示控制器的负载值，$P_{S_iC_j}$ 表示控制器 C_j 从交换机 S_i 接收到的 Packet_In 数量。

2) 控制器状态定义为 φ_j，φ_j 的值如式(3)所示。

$$\varphi_j=\begin{cases}0, \mathrm{LC}_j \leqslant \mathrm{LC}_{\text{threshold}}\\ 1, \mathrm{LC}_j > \mathrm{LC}_{\text{threshold}}\end{cases} \tag{3}$$

其中，$\mathrm{LC}_{\text{threshold}}$ 为控制器阈值。当控制器负载值大于阈值时，设定为 1。

3) 当 $\varphi_j=1$ 时，需要通过迁移操作来降低其负载。迁移过程中会产生一定的代价，迁移代价主要由以下 3 个部分组成：消息交换代价、负载增加代价、规则部署代价。迁移代价定义如式(4)所示。

$$w=r_{\text{mc}}+r_{\text{lc}}+r_{\text{rc}} \tag{4}$$

其中，r_{mc} 是交换机迁移的消息交换代价，r_{lc} 是负载增加代价，r_{rc} 是规则部署代价。

消息交换代价 r_{mc}。迁移请求将不同的交换机消息传递给目标控制器，因此该过程将产生消息交换代价，如式(5)所示。

$$r_{\text{mc}}=\left(\sum_{S_i\in C_r}(x_{ir}h_{ir})+\sum_{S_j\in C_k}(x_{jk}h_{jk})\right)\varepsilon t_{rk} \tag{5}$$

其中，ε 表示交换机的平均通信速率，h_{ir} 表示交换机 S_i 到控制器 C_r 之间的跳数，h_{jk} 表示交换机 S_j 到控制器 C_k 之间的跳数，x_{ir} 和 x_{jk} 分别表示控制器 C_r 和 C_k 域内设备的连接关系，t_{rk} 表示控制器 C_r 域内的交换机与目标控制器 C_k 的时延总和。

负载增加代价 r_{lc}。负载增加代价定义如式(6)所示。

$$r_{lc}=\begin{cases}P_{S_k}h_{S_kC_j}-P_{S_k}h_{S_kC_i}, P_{S_k}h_{S_kC_j}>P_{S_k}h_{S_kC_i}\\ 0, P_{S_k}h_{S_kC_j}\leqslant P_{S_k}h_{S_kC_i}\end{cases} \tag{6}$$

其中，P_{S_k} 表示交换机 S_k 发送的 Packet_In 数据分组，$h_{S_iC_k}$ 表示交换机 S_i 到控制器 C_k 之间的跳数。

规则部署代价 r_{rc}。选择要迁移的交换机时，控制器必须在域内将迁移规则 flow_mod 部署到此交换机中，其定义如式(7)所示。

$$r_{\text{rc}}=\delta_{\text{rule}}h_{ir}x_{ir} \tag{7}$$

其中，δ_{rule} 是控制器发送的 flow_mod 规则分组。

4) 用控制器负载的标准差来表示负载平衡因子，如式(8)所示。

$$\sigma=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{LC}_i-\overline{\mathrm{LC}}\right)^2} \tag{8}$$

交换机 S_k 迁移之后，新的负载平衡因子如式(9)所示。

$$\sigma^*=\sqrt{\frac{1}{N}\left(\sum_{i=1,i\neq j}^{N}(\mathrm{LC}_i^*-\overline{\mathrm{LC}}^*)^2+(\mathrm{LC}_j^*-\overline{\mathrm{LC}}^*)^2\right)} \tag{9}$$

其中，$\mathrm{LC}_i^*=\mathrm{LC}_i-L_{S_iC_j}$，$\mathrm{LC}_j^*=\mathrm{LC}_j-L_{S_iC_j}$，$\overline{\mathrm{LC}}^*$ 是新的平均负载，$L_{S_iC_j}$ 是控制器 C_j 从交换机 S_i 接收到的 Packet_In 数量。

5) 迁移效率的定义如式(10)所示。

$$\eta_{S_iC_j}=\frac{\left|\sigma^*-\sigma\right|}{w} \tag{10}$$

其中，$\left|\sigma^*-\sigma\right|$ 表示控制器迁移前后负载平衡变化，w 表示迁移该节点的迁移代价。

6) 在迁出控制器 C_m 域中选择目标交换机时，C_m 以高迁移效率迁移部分交换机节点以减轻其负载；从时延的角度来看，C_m 优先放弃远离它的交换机。因此，定义目标交换机被选择的概率如式(11)所示。

$$\rho_{S_j}=\eta_{S_jC_k}\frac{\left|\overline{\mathrm{LC}}-(\mathrm{LC}_k-\mathrm{LC}_i)\right|\mathrm{e}^{(\max h_{S_jC_k})}}{\sum\limits_{S_i\in\text{path}[]}\mathrm{e}^{(\max h_{S_jC_k})}} \tag{11}$$

其中，$h_{S_iC_k}$ 表示交换机 S_i 到控制器 C_k 之间的跳数，LC_k 和 LS_i 分别表示控制器和交换机的负载。

7) 链路拥塞率，用吞吐量和链路的带宽容量的比值来表示。

$$q_{\text{link}}=\frac{B}{C} \tag{12}$$

其中，q_{link} 表示链路的拥塞率，B 表示链路的吞吐量。通过发送 PORT_STATS_REQUEST 消息到交换机，获取交换机端口接受和发送的字节数以及时间。链路的吞吐量用式(13)表示。

$$B=\frac{T_2-T_1}{t_2-t_1}+\frac{R_2-R_1}{t_2-t_1} \tag{13}$$

其中，T、R 分别表示发送、接收的字节，t 表示统计的时间。拥塞率可由式(12)和式(13)计算得到。

8) 通过计算交换机被选择的概率以及链路拥塞率，得到优先级因子，用来在最小代价路径中确定目标交换机。设 θ 为优先级因子，如式(14)所示。

$$\theta=\frac{\rho_{S_j}}{q_{\text{link}}} \tag{14}$$

3.3　流量类别分类

在迁移过程中，除了要考虑迁移代价和负载均衡度，还要考虑流量类别，需要让重要流具有较高的迁移优先级。为了保证重要流能够被优先处理，本文通过对流量进行识别，划分优先级。

SDN 结合深度分组检测（DPI, deep packet inspection）[18]技术可以做到对流量的识别。本文的流量划分方案如表 1 所示，通过划分比特率将流量分为 6 类。

表 1　　流量分类

流量类别	特点	适用
会话业务流（CSF, conversation service flow）	传输速率恒定，严格要求传输时延、抖动等	语音
实时流媒体流（rt-SMF, real time-stream media flow）	具有持续比特率和突发比特率	实时电视
非实时流媒体流（nrt-SMF, non real time-stream media flow）	不指定时延偏差最大值，可以存在少量信元丢失	电子邮件
背景业务流（BSF, background service flow）	不保证服务质量	尽力而为的服务
交互业务流（IBF, interactive business flow）	存在有保障的最小容量	文件传输
信控管理流（S&M)	保证网络正常工作	—

根据不同流量的特点，本文采用的负载优先级划分方案如表 2 所示。

表 2　　优先级划分

优先级	pri	类型
低	4	BSF
↓	3	IBF
↓	2	CSF，(rt- nrt)-SMF
高	1	S&M

3.4　孤立点问题

迁移策略决定将交换机从某个域迁移到另外一个域，以减少该域控制器的负载。如果有一个或者多个交换机连接在待迁移的目标交换机之下，此时迁移该目标交换机可能会出现孤立节点问题，即该节点不再属于任意一个域，而要请求 2 个域，导致控制器响应时间增加。迁移的目标交换机越多，这种额外的负担越大。

如图 2(a)所示，若控制器 A 超过阈值，假设要将图中的 S_1 节点迁移到控制器 B 中，迁移完成后如图 2(b)所示，此时 S_2 和 S_3 这 2 个节点成为孤立节点，孤立节点 S_2 和 S_3 请求控制器时，不仅要向本域内控制器 A 发出请求，还要向迁移目标域中的控制器 B 发出请求，这将给 2 个控制器带来负担，导致性能下降。

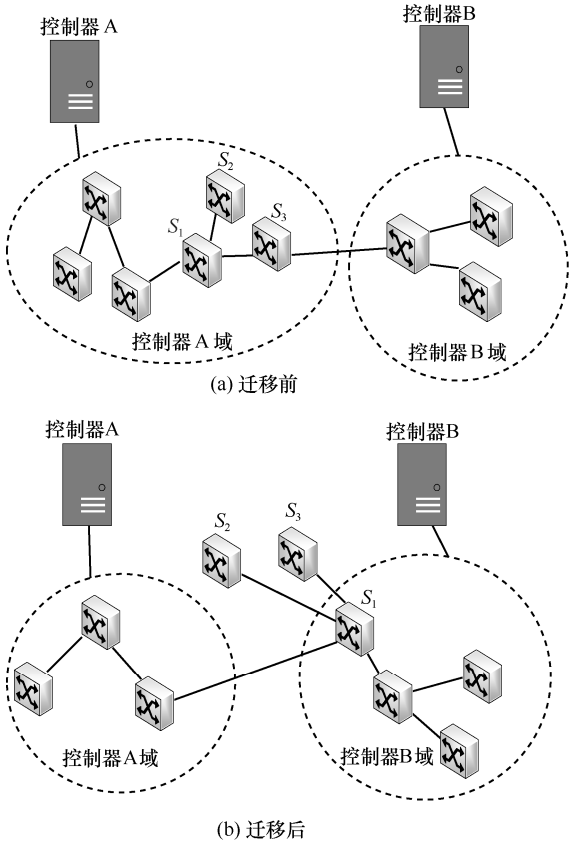

图 2　迁移前后的控制器集群对比

4　迁移模型

本文迁移模型如图 3 所示，包括 4 个模块：流量统计模块、流量预测模块、迁移策略决策模块及动作管理模块，每个模块对应功能具体如下。

1) 流量统计模块。该模块主要负责统计控制器接收来自自身域内交换机的流量信息，评估当前控制器的流量负载，并判断当前控制器在监测时间段内，流量是否超过预设的阈值。

2) 流量预测模块。该模块接收来自统计模块的负载统计信息与初步计算信息，根据预测模型预测控制器负载，得到负载预测矩阵，确定迁出控制器、目标控制器，判断是否需要迁移，避免频繁迁移。

3) 迁移策略决策模块。该模块作为本文方案的核心模块，负责根据网络信息，利用相应模型及工具，求解并决策具体交换机迁移策略。

4) 动作管理模块。该模块接收来自策略决策模

块的具体迁移策略，并负责向数据层交换机发送具体策略，实现交换机迁移过程。

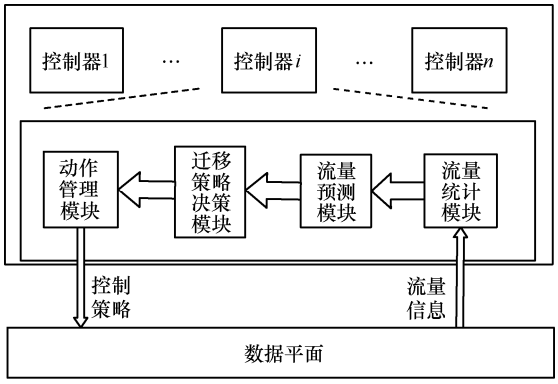

图 3 动态迁移模型

4.1 流量预测模块

现阶段的迁移算法大都是控制器超过阈值就进行迁移，这样会产生大量的迁移代价。比如控制器在当前时刻是超过阈值的，但是下一时刻控制器流量回归正常，则该时刻的迁移是不必要的，所以需要一种方法来避免这种不必要的迁移动作的发生。

自回归滑动平均（ARMA, auto-regressive moving average）模型可以有效分析短时间内流量稳定的网络数据，但对于存在网络异常的长时间网络流量数据却并不适合。其主要原因在于该模型建立的前提是被分析数据是平稳随机过程，而 SDN 流量具有非线性和不规则性的特点[19]，所以普通的 ARMA 模型不再适用。

差分整合滑动平均自回归（ARIMA，auto-regressive integrated moving average）模型[20]利用有限次差分操作后可以实现数据转化，使数据成为平稳数据，实现简单，所以在本文流量预测模块采用 ARIMA 模型。

ARIMA（p, d, q）模型的计算式为

$$\varphi(B)(1-B)^d X_t = \theta(B)\varepsilon_t \tag{15}$$

其中，B 为滞后算子，有

$$\varphi(B) = 1-\varphi_1 B-\varphi_2 B^2-\varphi_3 B^3-\cdots-\varphi_p B^p \tag{16}$$

$$\theta(B) = 1-\theta_1 B-\theta_2 B^2-\theta_3 B^3-\cdots-\theta_p B^p \tag{17}$$

ARIMA 建模首先需要判断被分析数据的平稳性，对于非平稳序列采取差分运算去掉其趋势或周期性，然后确定数据的自回归参数 p 和滑动平均参数 q，具体的建模及预测流程如下。

1) 采用迪基−福勒检验方法检测被分析数据的平稳性。

2) 数据平稳化。差分 d 次，若数据平稳则 d 为 0。

3) 参数分析。分别求得 ACF 和 PACF 曲线，观察并根据 AIC 判断拟合效果接近程度，确定 p 和 q。

4) 构建模型。利用上述步骤得到的参数构建 ARIMA（p, d, q）模型，并利用流量统计模块所得到的流量数据训练检测。

5) 预测分析。根据预测算法得到迁出控制器、目标控制器以及迁移触发时机。

算法 1 控制器负载预测算法

输入 Data

输出 controller < Overload, Target >

1) 接收统计模块的流量信息 Data_infomation

2) 初步判断

3) Overload controller C_0, Target controller C_1

4) 训练 ARIMA

5) → 控制器负载预测矩阵 $\boldsymbol{F\_C}$;

6) while $\mathrm{LC}_0' > \mathrm{LC}_{\mathrm{threshold}}$

7) C_0 → Overload; Move;

8) if (min($\bar{C}'$)== LC_1')

9) C_1 → Target;

10) else $\bar{C}$ → Target; Move_Start;

11) 更新 controller < Overload, Target >

算法 1 的详细流程如下。流量统计模块完成统计与初步评估之后，流量预测模块会收到来自流量统计模块的负载信息，初步判断过载控制器和目标控制器。通过构建的预测模型来预测下一时刻的负载情况，得到控制器负载预测矩阵，判断下一时刻过载控制器，如果依然过载，则选择该控制器为迁出控制器，并发出迁移触发信号 Move。通过负载矩阵判断下一时刻目标控制器是否是最低值，如果是，则选其为目标控制器；否则，选择矩阵中最低值控制器作为目标控制器，并发送 Move_Start 信号给过载控制器，响应 Move 信号，开始迁移操作。

4.2 迁移策略决策模块

该模块是整个迁移模型的核心，本文提出基于改进迪杰斯特拉的最小代价迁移算法，来减少迁移代价，保证控制器的性能。

4.2.1 确定最小代价路径

首先通过预测模型得到迁出控制器、目标控制器。起点 v_0 为迁出控制器，权值设置为迁移的代价。

在图 $G=(V,E)$中，初始化点 v_i 与点 v_j 的之间的权值 $w[i][j]$（由式(4)得出），即迁移该节点所产生的代价，寻找由顶点 v_0 到目标点的最小代价路径。

基于改进的迪杰斯特拉最小代价路径算法步骤如下。

1) 评估函数 $F(n)=f(n)+g(n)$，其中 $g(n)$为启发函数，表示从当前节点到目标节点的迁移代价估值；$f(n)$为前 n 个点所产生的代价。

2) 边的松弛。检查边(s,t)是否存在某一点 z，使$f((s,z)+(z,t)) <f(s,t)$，则更新最小路径。

3) 采用启发式搜索，在这里评估函数 F（n）用 $D[v]$表示，并且在函数中加入当前点到终点的欧氏距离。所以 $D[v]$定义为

$$D[v]=d[v]+\text{distance}(v,t) \tag{18}$$

其中，启发函数用欧氏距离表示，$g(v)$=distance(v,t)。$d[v]$表示起点到当前点 v 的路径长度，即所花费的迁移代价。

设起点、终点分别是 s、t，边为 $e(v,m)$，则松弛边 e 为

如果 $D[m]>d[v]+e(v,m)+\text{distance}(m,t))$，则更新 $D[m]$为

$$D[m]=d[v]+e(v,m)+\ \text{distance}(m,t)) \tag{19}$$

因为 $d[m]$是初始点 $s\rightarrow m$ 路径长，此时边 $e(v,m)$的另一个端点为 v，松弛该边为 $s\rightarrow v$ 和 $v\rightarrow w$，故

$$\begin{cases} d[m]=d[v]+e(v,m) \\ d[v]=D[v]-\text{distance}(v,t) \end{cases}$$

所以得到改进条件为

$$D[m]=D[v]-\text{distance}(v,t)+e(v,m)+\text{distance}(m,t)) \tag{20}$$

式(20)是采用启发式搜索的迪杰斯特拉算法。

本节获取预测算法所得到的迁出控制器、目标控制器，计算迁移交换机节点所产生的代价，然后通过改进迪杰斯特拉算法，得到一条最小代价路径 path[]。

4.2.2 对于迁移节点的选择

把 path[]这条路径上属于迁出控制器域的交换机节点加入待选择节点集合 MigNode{}中，并对集合中的节点进行判断，判断迁移目标交换机节点是否会出现孤立节点，把出现的孤立节点和目标交换机节点组成一个逻辑节点，并且将这个逻辑节点加入待选择节点集合中。

迁移优先级 S_Pri 与迁移因子 θ 成正比，与流量的 pri 成反比（优先级越高 pri 越小）。

$$\text{S\_Pri}_j=\theta_j\frac{1}{\text{pri}_j} \tag{21}$$

如果流的 pri=1，则优先处理该流，直接将该节点加入待迁移集合 MigNode{}中。

计算目标控制器能接受的流量 LC_{in}，LC_{in} 定义为目标控制器负载和控制平面控制器的负载均值的差值，如式(22)所示。

$$\text{LC}_{\text{in}}=\left|\text{LC}_{\text{target}}-\frac{1}{n}\sum_{i=1}^{n}\text{LC}_i\right| \tag{22}$$

其中，$\text{LC}_{\text{target}}$ 是目标控制器负载。

节点选择问题实际是动态规划问题，优化的目标是：在总容量 LC_{in} 的限制下，选择的节点的优先级最大。若存在 pri=1 的流，则优先处理该节点，剩余负载空间 $\text{lc}=\text{LC}_{\text{in}}-\text{Load}[\text{path}_{(\text{pri}=1)}]$。

1) 问题分析

在前 i 个交换机节点中，选择若干个节点加入待迁移集合{}内。

状态：在前 i 个交换机节点中，选择若干节点加入所剩负载空间为 lc 的待迁移集合 MigNode{}中获得最大优先级。

决策：是否选择第 i 个交换机节点加入 MigNode{}中。

2) 状态转移方程

设 $F(i,j)$表示负载容量为 lc 的待迁移集合加入前 i 个交换机节点后所占据的最大优先级，则有

$$F(i,j)=\max \begin{cases} F(i-1,\text{lc}-\text{Load}[i])+\text{S\_Pri}[i],\text{选择第}i\text{个节点} \\ F(i-1,\text{lc}),\text{不选择第}i\text{个节点} \end{cases} \tag{23}$$

其中，$F(0,j)=0$，Load[i]表示第 i 个节点的负载，S_Pri[i]表示第 i 个节点的迁移优先级。

4.2.3 对于孤立节点的处理

通过上述方法，将部分孤立节点和目标节点作为逻辑节点一起迁移，解决部分孤立节点，不是所有的孤立节点都可以跟随目标交换机节点一起迁移到目标控制器域中去。未被解决的孤立节点内的交换机在迁移完成后会同时请求本域和目标域控

制器，给 2 个控制器带来负担，造成控制器性能降低，这是亟待解决的一个问题。

从图 G 中的迁出控制器节点出发，访问该节点邻接顶点 V_{i1}，V_{i2}，…，V_{in}，然后访问 V_{i1} 相邻的 V_{21}，V_{22}，…，V_{2n}，直到遍历所有节点，得到广度优先遍历生成树。

存储方法：用孩子兄弟表示法来存储，如图 4 所示，每个节点由 3 个部分组成。

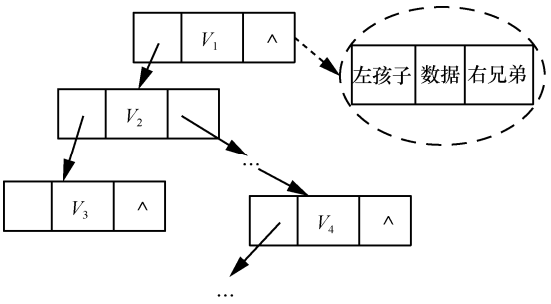

图 4 节点构成示意

孤立节点处理算法伪代码如算法 2 所示。

算法 2 孤立节点处理算法

输入 Tree tree

输出 $\Omega_{i,j}$

1) while L_{node} → firstchild !=null||L_{node} → rightbrother! =null)
2) if L_{node} 存在右兄弟
3) *p=L_{node}→rightbrother;
4) L_{node} → firstchild → rightbrother = p → firstchild;
5) p→firstchild==L_{node}→firstchild
6) //将其孩子连接到右兄弟上
7) end if
8) if L_{node} 没有右兄弟
9) 找到前驱节点
10) s→rightbrother||s→firstchild==L_{node};
11) if s 是兄弟节点
12) s→firstchild→rightbrother=L_{node}→firstchild ;
13) //将其孩子连接到兄弟节点上
14) else //s 是父节点
15) s→firstchild=L_{node}→firstchild ;
16) //将其孩子连接到他的父节点上
17) end if
18) end if
19) end while
20) delete(tree, L_{node});

算法 2 的详细流程如下。首先判断目标交换机节点是不是叶子节点，如果是叶子节点，则不会产生孤立节点，直接删除。当目标交换机节点的孩子节点不为空时，如果存在右兄弟，找到右兄弟的左孩子，并将目标交换机节点的孩子节点连接到右兄弟的孩子节点上；否则，找到其前驱节点，连接到其前驱节点上。该节点的孩子节点如果为空，证明已经处理掉该节点迁移所产生的孤立节点，删掉该节点。处理的原则就是尽量保持在和原先交换机一个结构层次上，即尽量将其孩子节点连接到同一级交换机下。

4.3 动作管理模块

在该模块中完成交换机迁移操作并更新网络元素的连接矩阵 $\boldsymbol{\Omega}=[x_{ij}]_{M\times N}$，改变控制器角色。如图 5 所示，首先通过预测模块确定的过载控制器和目标控制器，过载控制器选择要迁移的迁移交换机组，并将触发迁移信号 Move 发送到目标控制器 C_{target}。C_{target} 接收到 Move 信号后将响应迁移开始信号 Move_Start，开始执行迁移动作。如果交换机迁移完成，则迁移交换机的主控制器将从过载控制器转移到目标控制器，迁移活动中断，并且会将更新消息广播到整个网络。

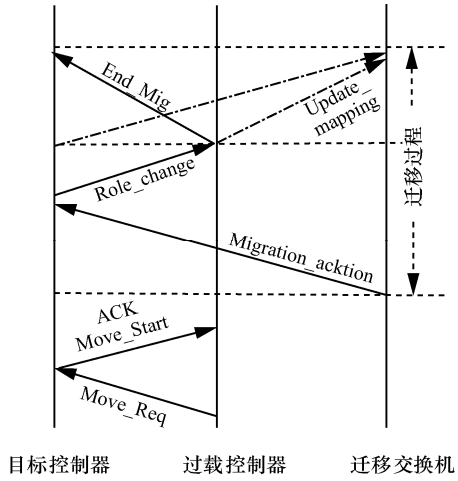

图 5 交换机迁移过程

基于最小代价路径的交换机迁移算法伪代码如算法 3 所示。

算法 3 基于最小代价路径的交换机迁移算法

输入 graph G

输出 migration plan P

1) 计算控制器负载情况

2) 判断 Overload controller C_0, Target controller C_1

3) Dijkstra(Graph G,int v),

4) 得到 C_0 到 C_1 之间的代价最小路径 path[]

5) 选择 path[i]$\in C_0$

6) $A[j]$←path[i]，计算 Load$A$;

7) if Load$A$ + $\mathrm{LC}_1 < \mathrm{LC}_{1\_thresthod}$

8) migration

9) if (Isolatednode( )==1)

10) 生成逻辑节点 $A[i]$, Inode[j] → LogNode[j]

11) 处理 InodeProcessing( )

12) else 计算 C_{in}

13) LogNode[j]→A[length+j]

14) 选择迁移节点 MigNode[]

15) migration

16) if (Isolatednode( )==1)

17) 生成逻辑节点 $A[i]$,Inode[j]→ LogNode[k]

18) 处理 InodeProcessing( )

算法 3 的详细流程如下。通过流量统计模块以及预测模块，首先得到过载控制器、目标控制器和确定迁移时机，通过改进的迪杰斯特拉算法得到 2 个控制器之间的最小代价迁移路径。计算 path[]路径上的属于过载控制器的节点负载之和，因为过载控制器一般是主干控制器，或者其他需要频繁处理更多消息的控制器，目标控制器的选择一般是在很长的一段时间内负载较少的控制器，所以如果不超过目标控制器的容量阈值，则直接进行迁移，并判断迁移后是否出现孤立节点，同时得到逻辑节点 LogNode，将逻辑节点加入待选择节点集合 A 中，处理孤立节点；否则，计算目标控制器所接受的负载，计算 A 中节点的最优负载组合 MigNode[]，进行迁移，并判断迁移后是否出现孤立节点，处理孤立节点。

5　实验结果分析

本文的实验环境为 Mininet+OpenDaylight。通过在 Mininet 中运行自定义的拓扑并连接 OpenDaylight 来搭建本文实验的网络环境，利用 iperf 工具模拟发流测试控制器的性能。3 个 ODL 控制器 C_0、C_1 和 C_2 部署在控制器集群中，12 个 OpenvSwitch 连接到 3 个控制器上。使用 iperf 模拟不同速率的数据分组。实验拓扑如图 6 所示。

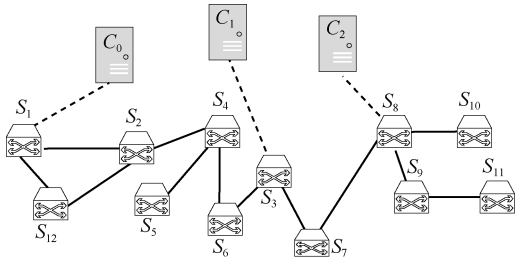

图 6　实验拓扑

1) 控制器负载

本文定义控制器负载占比为实际控制器收到 Packet_In 请求数据分组与控制器容量的比值，如式(24)所示。

$$\sigma = \frac{\mathrm{LC}}{C_m} \tag{24}$$

图 7 和图 8 清楚地显示在相同的流量发送情况下，使用 SMCP 算法和未使用任何负载均衡算法控制器负载占比的对比。

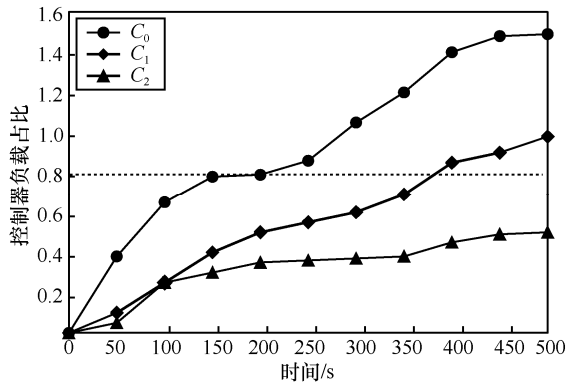

图 7　静态分配下控制器负载占比

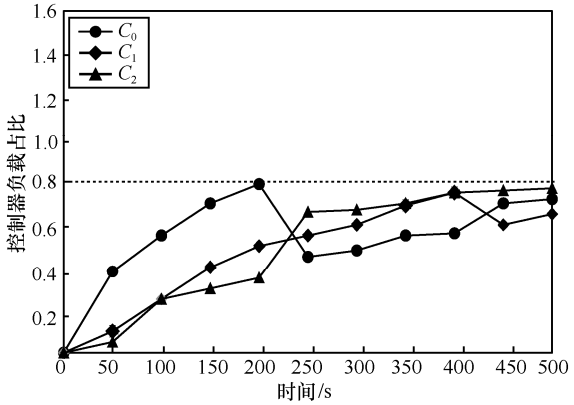

图 8　SMCP 下控制器负载占比

图 7 显示了没有负载平衡的 3 个控制器的负载占比。图 8 是在运行 SMCP 算法后的控制器负载占比。由图 8 可知，在实现 SMCP 算法后，负载只要高于阈值就会触发 Move_Req 信号，负载在下一时刻没有降低至阈值之下，则响应请求，通过迁移操作快速降低其负载，使控制器的负载基本维持在阈值以下；而在静态分配情况下，控制器负载超过阈值，随时间增加，过载情况继续加剧。

2) 响应时间

通过测试控制器的响应时间来衡量不同算法的负载均衡性，采用控制器负载均衡动态自适应算法（DALB, dynamic and adaptive algorithm for controller load balancing）[21]以及最大化资源利用率迁移算法（MUMA）来进行比较。

响应时间对比如图 9 所示。由图 9 可知，本文提出的 SMCP 算法的控制器响应时间较 MUMA 和 DALB 算法更短，其性能优于 MUMA 和 DALB 算法的控制器性能。图 10 表示发送速率不变，随着网络规模的增加，控制器的平均响应时间变化。实验表明，相比其他 2 种负载均衡算法，SMCP 算法的控制器响应时间更短，性能更优。

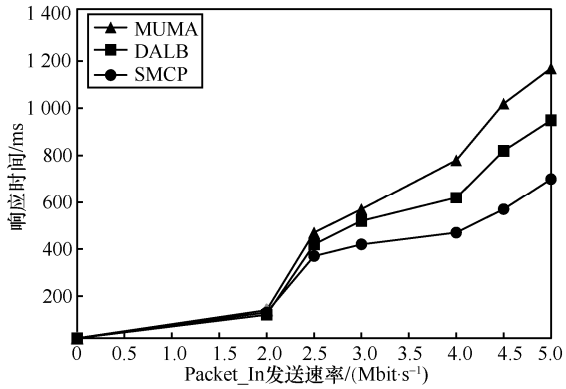

图 9 响应时间对比

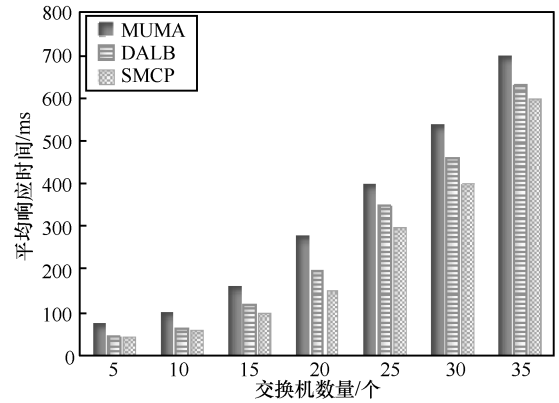

图 10 网络规模与响应时间关系

3) 均衡时间

在实验中增大发送速率，直到 C_0 过载，通过比较控制器的负载以及不同的迁移算法实现控制器均衡的时间来衡量算法的优劣性。

均衡时间对比如图 11 所示，反映了 MUMA、DALB 和 SMCP 中 C_0 的负载随时间变化的曲线。在 150 s 时，C_0 接收到大量来自本域交换机的请求，造成过载。在方案 DALB 中，需要 2 个均衡过程才能完成，在 300 s 时完成迁移。在方案 SMCP 中，控制器通过一个均衡过程降低负载，在 250 s 时完成迁移操作。而在 MUMA 中，由于交换机的选择是随机的，C_0 的负载降低时间比 DALB 和 SMCP 都要长。因此，本文提出的方案缩短了迁移时间，使控制平面快速完成负载均衡。

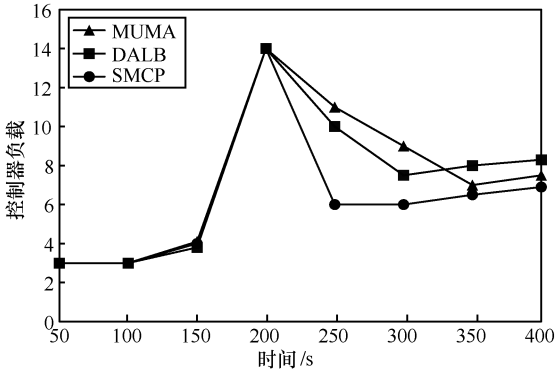

图 11 均衡时间对比

4) 系统稳定性

系统的稳定性与迁移次数有关，因为在迁移过程中，某些交换机可能无法及时处理新连接，从而造成迁移中断等问题。

在当前规模下，SMCP 算法只需要迁移 1 次，而 DALB 和 MUMA 算法则需要迁移 2 次。由表 3 可知，由于 SMCP 每次负载均衡都只需要迁移 1 次，随着交换机规模不断增加，这种差距表现出明显的不断扩大的趋势。

表 3 不同网络规模下的平均迁移次数

交换机数量/个	MUMA/次	SMCP/次	DALB/次
12	2	1	2
24	6	1	4
36	8	1	6

频繁的迁移还会破坏网络的稳定性，图 12 是随时间变化的迁移次数的变化曲线。由图 12 可以

看出，由于 MUMA 基于随机选择，无法保证迁移后控制平面是否均衡，因此迁移的频率较高。DALB 算法在一定程度上降低了迁移次数，但是迁移频率依然高于 SMCP。而 SMCP 通过迁移交换机集合，在每个负载均衡过程中只需要迁移 1 次，而且每次迁移操作都可以较好地实现控制平面的负载均衡。

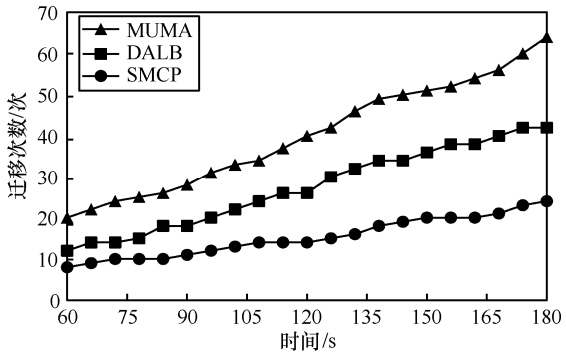

图 12　迁移次数对比

吞吐量也是衡量系统稳定性的一个重要指标。图 13 和图 14 是 3 种算法吞吐量的对比。由图 13 可知，第 200 s 开始，以 3:1:2 的比例向控制器 C_0、C_1、C_2 发送数据分组，Packet_In 请求增多，控制器间的负载不均衡导致吞吐量下降，MUMA 算法在 250 s 后吞吐量有所上升，但基本保持在一种比较低的状态；虽然 DALB 吞吐量恢复到正常水平，但是与 SMCP 算法相比，需要更长的时间；SMCP 算法能较快地迁移过载控制器 C_0 的部分负载，且迁移后控制平面有较好的均衡性，从而提高了系统吞吐量。在图 14 中，网络空闲时 3 种算法吞吐量都不断增长，随着负载的加重，MUMA 的吞吐量快速降低，SMCP 和 DALB 算法吞吐量也有所下降，但基本维持在较高的吞吐量，SMCP 下降速度要优于 DALB 算法，吞吐量也更高。

在衡量系统稳定性时，分组丢失率也是非常重要的一个指标。选择控制器 C_0 和 C_2 域内的 2 台主机进行传输，通过发送不同速率的 Packet_In 数据分组来对比 3 种算法的分组丢失率。从图 15 实验结果可知，当发送速率大于 2 Mbit/s 时，3 种算法的分组丢失率都开始增加，MUMA 算法不能很好地感知网络状态，迁移后也可能造成新的过载，分组丢失率随着发送速率的增加快速增加；SMCP 算法的控制器、交换机选取较灵活，并且迁移过程中充分考虑到时延、带宽和跳数等影响网络状态的因素，与 DALB 算法相比，分组丢失率在一定程度上有所下降。

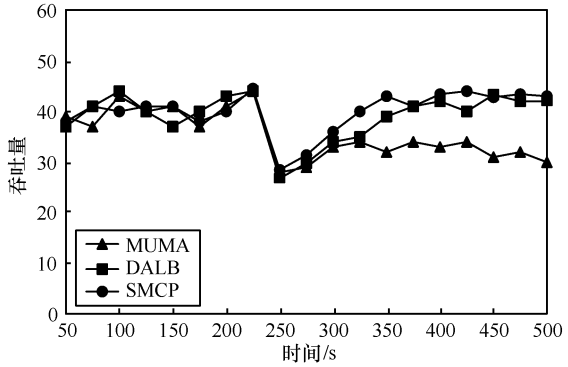

图 13　吞吐量随时间变化曲线

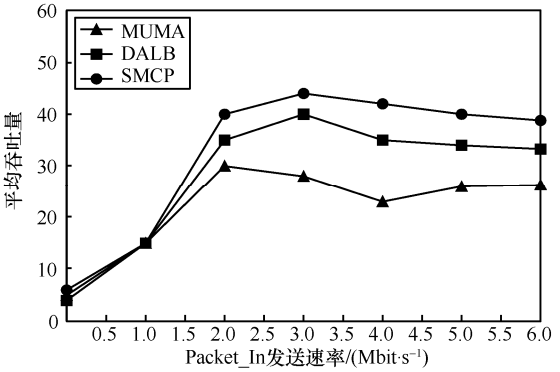

图 14　吞吐量随发送速率变化曲线

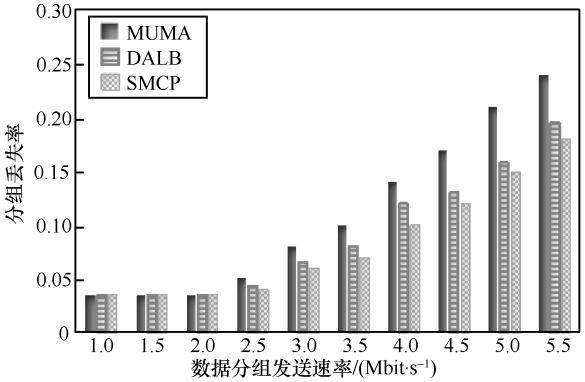

图 15　3 种算法的分组丢失率对比

6　结束语

本文就如何保护控制器，尤其是骨干控制器免受安全威胁与攻击，提高 SDN 控制平面的安全性问题，提出了基于最小代价路径的交换机迁移方法。通过获取数据平面负载信息，在预测模块实现控制器负载预测，得到确定迁出–目标控制器集合，根据最小代价路径算法，得到一条最小代价路径，对该路径上的节点通过节点选择算法，得到目标交换机集合，实施迁移操作，最后通过孤立节点处理算法解决孤立节点问题。本文方法的优势如下。

1) 基于最小代价路径的交换机迁移算法，得到一条最小代价迁移路径，结合控制器的状态，以及交换机负载的优先级确定迁移交换机集合，然后将迁移集合内的交换机迁移到目标控制器，可以减少迁移所产生的代价，并且保证重要流优先迁移。2) 通过交换机集合迁移的方式，减少迁移次数，缩短迁移所花费的时间，提高网络的稳定性。3) 在本文的迁移模型中还加入了负载预测模块，传统的方式是判断控制器负载情况，只要超过阈值就迁移，这样会出现大量不必要的迁移，本文通过预测控制器未来时刻的负载情况，根据负载预测矩阵确定迁出−目标控制器集合，以及触发迁移时机，减少频繁迁移，保证系统稳定性。4) 通过孤立节点处理算法减少控制器的响应时间，提高了控制器性能。通过实验验证，本文算法可以很好地实现控制平面的负载均衡，减少主干控制器被监听被识别的概率，并且减少了迁移次数，降低了迁移代价，提高了控制器性能。

参考文献：

[1] 互联网发展趋势报告（2017-2018 年）白皮书[R]. 北京: 中国信息通信研究院, 2017.
White paper of Internet development trend report (2017-2018)[R]. Beijing: China Academy of information and communication, 2017.

[2] ELLIOTT C. GENI: opening up new classes of experiments in global networking[J]. IEEE Internet Computing, 2010, 14(1): 39-42.

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

[4] 黄稻, 刘江, 魏亮, 等. 软件定义网络核心原理与应用实践[M]. 北京: 人民邮电出版社, 2014.
HUANG T, LIU J, WEI L, et al. Software defines network core principles and application practices[M]. Beijing: Posts and Telecom Press, 2014.

[5] CHENG T, WANG M, JIA X. QoS-guaranteed controller placement in SDN[C]//IEEE Global Communications Conference. IEEE, 2015:1-6.

[6] 邓志华, 吕光宏. SDN 网络可扩展性问题的研究[J]. 计算机技术与发展, 2016, 26(3): 14-17.
DENG Z H, LYU G H. Research on scalability of SDN network[J]. Computer Technology and Development, 2016, 26(3): 14-17.

[7] 杨思锦, 庄雷, 胡颖, 等. 一种动态自调节的 SDN 控制器负载均衡算法[J]. 计算机应用与软件, 2016, 33(12): 71-74.
YANG S J, ZHUANG L, HU Y, et al. A load balancing algorithm of SDN controller with dynamic self-regulation[J]. Computer Application and Software, 2016, 33(12): 71-74.

[8] 丁佳丽. 基于 OpenDaylight 的 SDN 控制器集群的设计与实现[D]. 南京: 东南大学, 2016.
DING J L. Design and implementation of SDN controller cluster based on OpenDaylight[D]. Nanjing: Southeast University, 2016.

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

[10] LIANG C, KAWASHIMA R, MATSUO H, et al. Scalable and crash-tolerant load balancing based on switch migration for multiple OpenFlow controller[C]//2nd International Symposium on Computing and Networking. 2014:171-177.

[11] ZHANG S J, LAN J L, SUN P H, et al. Online load balancing for distributed control plane in software-defined data center network[J]. IEEE Access, 2018, 6: 18184-18191.

[12] ALI N, NIMA N, MEHDI H. Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature[J]. IEEE Access, 2018, 6: 14159-14178.

[13] WANG C A, HU B, CHEN S Z, et al. A switch migration-based decision-making scheme for balancing load in SDN[J]. IEEE Access, 2017, 5: 4537-4544.

[14] YE X, CHENG G Z, LUO X G. Maximizing SDN control resource utilization via switch migration[J]. Computer Networks, 2017, 126: 69-80.

[15] LI J M, HU X L, ZHANG M Q. Research on dynamic switch migration strategy based on FMOPSO[C]//2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2018: 913-917.

[16] KIM S, EBAY K S, LEE B, et al. Load balancing for distributed SDN with harmony search[C]//2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2019: 1-2.

[17] 李婉, 沈苏彬, 吴振宇. 一种基于多 SDN 控制器的交换机迁移机制[J]. 计算机技术与发展, 2018,28(1): 89-99.
LI W, SHEN S B, WU Z Y. A switch migrationmechanism based on multiple SDN controllers[J]. Computer Technology and Development, 2018, 28(1): 89-99.

[18] DASILVA S, MACHADO C, BISOL V, et al. Identification and selection of flow features for accurate traffic classification in SDN[C]//Network Computing and Applications (NCA). IEEE, 2015: 134-141.

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

[20] DEEPAK N, BYRAV R, BRIAN B, et a1. Large data transfer predictability and forecasting using application-aware SDN[C]//2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). IEEE, 2018: 1-2.

[21] CHENG G, CHEN H, WANG Z, et al. DHA: distributed decisions on the switch migration toward a scalable SDN control plane[C]//IFIP Networking Conference (IFIP Networking). IEEE, 2015: 1-9.

[作者简介]

赖英旭（1973− ），女，辽宁抚顺人，博士，北京工业大学教授，主要研究方向为工业控制网络安全、软件定义网络安全等。

蒲叶玮（1994− ），男，山东淄博人，北京工业大学硕士生，主要研究方向为信息安全、软件定义网络等。

刘静（1978− ），女，北京人，博士，北京工业大学助理研究员，主要研究方向为工业互联网安全、可信计算等。