doi:10.11835/j.issn.1000-582X.2022.217

# 分布式软件定义网络中多域流量工程的路由优化方法

王 坤[1a,1b],吕光宏[2],胥 林[1a,1b],杨 晗[1a],邓 慧[1a]
(1. 西南石油大学 a. 计算机与软件学院,成都 610500; b. 数据挖掘与知识管理南充市重点实验室,四川 南充 637001; 2. 四川大学 计算机学院,成都 610065)

**摘要:**针对分布式软件定义网络(software-defined networking,SDN)中流量管理调度不均衡的流量工程问题,提出一种基于负载均衡的多控制域流量路由优化的解决方案。首先分析控制消息流量的组成、域内通信及域间通信规则;然后基于4种控制消息定义控制链路流量的构成,明确链路承载流量分为控制消息流量和业务流量,建立平衡控制器负载和最小化最大链路利用率的优化模型;最后基于域内通信和域间通信提出两层路由算法。为提高模型求解精度,进一步提出改进离散萤火虫算法求解最优路由。结合ABILENE网络和GEANT网络,分析控制消息流量、控制器负载和链路负载等评价指标。实验结果表明,优化模型能有效实现控制器和链路负载均衡,控制消息流量是流量工程重要组成部分。相比集中控制模式,扁平分布式控制模式的平均控制器负载降低47.3%,最大链路利用率相差不超过15%。

**关键词:**软件定义网络;流量工程;多控制域;离散萤火虫算法
**中图分类号:**TP393 **文献标志码:**A **文章编号:**1000-582X(2024)07-110-15

# Routing optimization method for multi-domain traffic engineering in distributed software-defined networking

*WANG Kun*[1a,1b], *LV Guanghong*[2], *XU Lin*[1a,1b], *YANG Han*[1a], *DENG Hui*[1a]
(1a. School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, P. R. China; 1b. Data Mining and Knowledge Management Key Laboratory of Nanchong City, Southwest Petroleum University, Nanchong, Sichuan 637001, P. R. China; 2. College of Computer Science, Sichuan University, Chengdu 610065, P. R. China)

**Abstract:** Addressing the challenge of unbalanced traffic management and scheduling in distributed software-defined networking (SDN) for traffic engineering, we propose a solution for traffic routing optimization across multi-control domains based on load balancing. Firstly, we define the composition of message traffic and the rules for intra-domain and inter-domain communication. Then, we establish an optimization model aiming at balancing controller loads and minimizing maximum link utilization. The model is based on the composition of control link traffic using four control messages, with link traffic divided into control message traffic and network traffic. Finally, we propose a two-layer routing algorithm based on communication rules. To improve the accuracy of the

**收稿日期:**2022-08-22 **网络出版日期:**2023-03-03
**基金项目:**国家自然科学基金资助项目(61373091);四川省南充市科技资助项目(19SXHZ0012)。
Supported by National Natural Science Foundation of China (61373091), and the Science & Technology Program of Nanchong, China (19SXHZ0012).
**作者简介:**王坤(1987—),男,主要从事软件定义网络研究,(E-mail)wk_scu@163.com。
**通信作者:**吕光宏,男,教授,(E-mail)lghong@scu.edu.cn。

model solution, we introduce an improved discrete firefly algorithm. Evaluating the model using the ABILENE network and GEANT network, we assess indicators such as control message traffic, controller load, and link load. Experimental results show that the optimization model effectively balances loads between controllers and links, emphasizing the significance of managing message traffic in traffic engineering. Compared to centralized control modes, the average controller load in the flat distributed control mode is reduced by 47.3%, with the the maximum link utilization difference not exceeding 15%.

**Keywords:** software defined networking; traffic engineering; multi-domain; discrete firefly algorithm

软件定义网络(software-defined networking,SDN)是将网络分为控制平面和数据平面的一种新型网络结构,能有效解决传统网络中流量拥塞、延迟、抖动等痛点,实现流量调度动态灵活、流量测量准确有效的高效流量工程模式[1]。单控制器集中式流量控制模式存在单点故障、负载过高等问题,多控制器分布式管理网络资源成为一种必要保障[2]。SDN 基于全局网络视图、网络状态和流量模式等特征能实现新颖的流量工程解决方案[3],特别是多控制器部署方案下的流量工程是研究热点之一[4]。

基于 SDN 的流量工程能提供可靠、全面、高效的流量调度策略,实现交换机之间的业务流量转发[5],通过多控制器之间互相协作规划最优路由。相关学者从集中式控制平面提出流量工程方案[6-10]。Agarwal 等[6]首次提出基于流量工程实现 SDN 网络动态路由算法,以达到时延和丢包率的最小化,降低网络链路的负载。Guo 等[7]基于混合 SDN 架构,提出了一个混合整数非线性规划问题,采用启发式算法 SDN/OSPF 流量工程(SDN/OSPF traffic engineering, SOTE)联合优化 SDN 节点的开放式最短路径优先(open shortest path first, OSPF)权值设置和流量分割比,提高 SDN 节点部署效益。Ammal 等[8]提出使用新的白蚁启发的优化算法优化网络链路利用率,实现网络动态路径分配,有效避免应用程序对带宽的弹性需求拥塞。Wang 等[10]研究如何在线对流量矩阵和流量工程联合优化,提出一种最大负载优先的流量规则生成策略,有效解决流量矩阵测量问题。多控制器解决方案是大规模 SDN 网络推广的必然,部分学者进一步研究多控制器协作规划路由[11-14]。Huang 等[11]实现一个多控制器的流量工程,建立流请求与多个交换机的映射关系,找到最小化流量设置时间的长期交换机–控制器映射和流量分布方案,实现预先计算备份控制器以及控制器故障时的流量分配。Sridharan 等[12]提出一种多控制器流量工程方案,实现控制器与交换机映射和流量分配,使流量设置时间最小化,能有效解决链路变化和控制器故障问题。Hua 等[14]考虑多域多层 SDN 架构,基于分层控制平面提出一种多域 SDN 流量工程方案,通过共享网络信息库,实现本地控制器规划路由和根控制器集中决策路由,有效解决层次控制架构下的流量工程。上述研究未讨论分布式控制器架构下的流量工程问题,忽略控制器控制路径产生流量对整个网络流量路由的影响,特别是控制链路拥堵、负载较高时,易引发网络控制崩溃的风险。

在扁平分布式控制平面中,控制消息是多控制器之间网络视图和链路信息同步的载体;对流量工程的研究不能忽略控制消息流量对链路流量的影响。同时,从查阅的相关研究中未发现同时探讨链路负载和控制器的负载均衡。基于此,笔者通过分析控制消息流量的生成规则,提出一种域间流量路由的流量工程方案。

1) 引入 4 种控制消息流量,分析控制消息流量所在路由对链路负载的影响。

2) 基于多控制器联合控制机制,讨论域内通信和域间通信的策略,建立优化控制器负载和链路负载的流量优化模型。

3) 提出两层路由(two-layered routing,TLR)算法求解流量的域内和域间路由。考虑到模型复杂度,进一步提出一种改进的离散萤火虫(discrete firefly algorithm,DFA)算法提高模型求解精度。

实验结果表明,流量优化模型能实现控制器和链路负载同时均衡的目标,证明控制链路流量对流量工程的重要性。

# 1 问题描述及模型

## 1.1 问题描述

扁平分布式控制平面中,各个控制器地位平等,共享网络拓扑和链路状态,各自决策控制域内路由。流量路由存在域内和域间2种情况,需要探讨控制消息流量和业务流量的路由策略,制定流量工程方案。

设SDN拓扑用图 $G=(V,E)$ 表示,$V$ 表示节点集合,$E$ 为连接节点的边集合,元素 $e(i,j)\in E$ 表示节点 $i$ 到节点 $j$ 的边,$S=\{s_1,s_2,...,s_m\}$ 表示 $m$ 个交换机节点集合。拟定控制器部署在交换机的位置,用 $C=\{c_1,c_2,...,c_n\}$ 表示 $n$ 个控制器节点集合,且 $C\subseteq S$。用矩阵 $\boldsymbol{R}_{m\times n}$ 表示交换机与控制器之间的从属关系,元素 $r(i,j)$ 表示交换机 $s_i$ 是否从属于控制器 $c_j$,有

$$r(i,j)=\begin{cases}1, & s_i\subseteq c_j;\\ 0, & \text{其他。}\end{cases} \tag{1}$$

SDN分布式控制机制中,数据处理采用被动模式,当开放式流量(OpenFlow)交换机接收到一个新数据流时,交换机在匹配失败后向所属控制器发送流请求消息(PACKET-IN),控制器根据网络状态确定路由后下发转发实体消息(PACKET-OUT)到相关路由交换机。为了维护网络状态一致性,控制器需要定期查询所属交换机状态,向控制域内交换机发送查询信息(ECHO),此外需要定期同步网络其他控制器的网络信息,相互之间传递同步状态消息(SYNCHRONIZATION)。对上述4种控制消息,约定消息大小分别为 $Q_1$、$Q_2$、$Q_3$、$Q_4$,其单位与流量单位相同。

**表1 符号定义**

**Table 1 Definition of notations**

| 符号 | 定义 |
|---|---|
| $G=(V,E)$ | 无向图 $G$,$V$ 表示节点集合,$E$ 表示连接节点边集合 |
| $S$ | 交换机节点集合 |
| $C$ | 控制器节点集合 |
| $e(i,j)$ | 节点 $i$ 到节点 $j$ 的边 |
| $\boldsymbol{M}$ | 流量矩阵集合 |
| $m(i,j)$ | 单位时间内交换机到交换机的业务流(origin-destination,OD)大小 |
| $p(o,d)$ | 源节点 $o$ 到目的节点 $d$ 的最佳路由 |
| $n_p(o,d)$ | 路径 $p(o,d)$ 的节点有序集合 |
| $x_{(o,d)}^{(i,j)}$ | 路由 $p(o,d)$ 是否包含链路 $e(i,j)$,若包含,则 $x_{(o,d)}^{(i,j)}=1$,否则为0 |
| $o(e(i,j))$ | 链路 $e(i,j)$ 上流量总和 |
| $\sigma(e(i,j))$ | 链路 $e(i,j)$ 容量上限 |
| $\mu(e(i,j))$ | 链路 $e(i,j)$ 利用率 |
| $U(j)$ | 控制器 $c_j$ 的容量 |

## 1.2 相关定义

### 1.2.1 控制域

控制域表示控制器和所控制交换机构成的一个区域,以更好地管理当前区域的网络设备和路由,如图1所示。图中host表示主机。

图1中网络划分为2个控制域:Domain01和Domain02,对应控制器分别为 $C_1$ 和 $C_2$。

用 $D(j)$ 表示控制器 $c_j$ 所控制交换机的集合,即

$$D(j)=\{s_i\in S|r(i,j)=1\}。 \tag{2}$$

交换机 $s_i$ 对应的控制器用rf($i$)表示,

$$\mathrm{rf}(i)=j,\ \exists r(i,j)=1,\ j\in C。 \tag{3}$$

如果存在任意2个交换机分别属于不同控制域,且两者存在一条链路直接连接,则称这2个控制域是相

邻的。用$A(i,j)$表示控制域$i$和$j$是否相邻,

$$A(i,j)=\begin{cases}1, & \exists e(k,l)\in E,\ \forall k\in D(c_i),\ \forall l\in D(c_j);\\ 0, & \text{其他。}\end{cases} \tag{4}$$

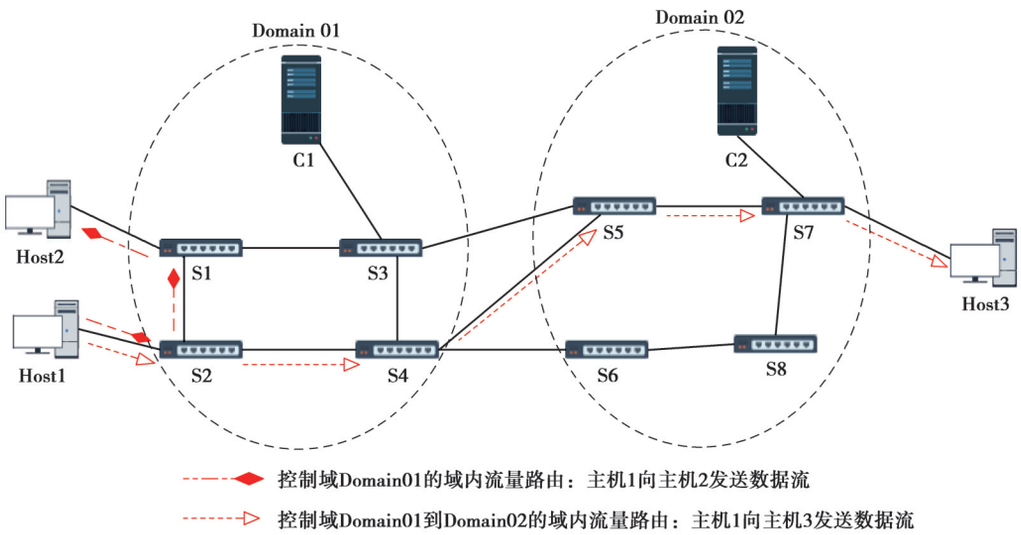


**图 1　域内流量与域间流量**

**Fig. 1　Intra-domain traffic vs inter-domain traffic**

1.2.2　域内通信

当一条业务流(origin-destination,OD)的源交换机和目的交换机处于同一控制域,则转发路由交换机也应属于当前控制域,不能出现其他控制域的交换机有转发情况;控制器会根据当前网络链路情况,为域内交换机之间的流量需求寻找最优路径。图 1 中主机 1 向主机 2 传递数据,其所属交换机分别为$s_2$和$s_1$,经过交换机$s_2$、$s_1$路由实现转发。此时$s_1$、$s_2$处于控制域 01,只需要控制器$c_1$计算最佳路由。

域内通信除了交换机之间流量转发路由,还包括交换机向控制器发送流请求、控制器下发转发实体到交换机和控制器轮询交换机状态 3 类控制路由。控制路由在控制器部署时将最佳路由写入交换机和控制器的流表,默认保持不变。如图 1 中,以 Domain01 为例,采用以传播时延为权重的 Dijkstra 算法实现域内最短路由,则控制器$c_1$与交换机$s_1$、$s_2$、$s_3$、$s_4$的控制路由分别为$\{<c_1,s_3>,<s_3,s_1>\}\{<c_1,s_3>,<s_3,s_1>,<s_1,s_2>\}$ $<c_1,s_3>\{<c_1,s_3>,<s_3,s_4>\}$。

1.2.3　域间通信

如果 OD 流对应的源交换机和目的交换机处于不同控制域,将发生跨域通信。当前控制器计算域内最佳路由,并确定下一个通过的控制域和接入交换机,重复进行,直到建立到目的交换机的路由。具体域间路由策略见后文。图 1 中主机 1 向主机 3 传递数据,其所属交换机分别为$s_2$和$s_7$,交换机$s_2$、$s_7$分别位于 Domain01 和 Domain02,控制器$c_1$、$c_2$根据网络情况确定各自最佳路由,其转发路径为$<s_2,s_4,s_5,s_7>$。

多控制器分布式机制中,各个控制器需要定期同步网络全局状态,维护网络一致性,这需要建立控制器之间的同步路由。同步路由与控制器位置有关,一旦控制器位置确定,在不考虑网络突发情况下,其路由保持不变。图 1 中,控制器$c_1$和控制器$c_2$需要定时更新网络状态,通过同步路由收集其他控制域内设备和链路状态。控制器$c_1$和$c_2$之间的最佳路由采用最短路径确定为$\{<c_1,s_3>,<s_3,s_5>,<s_5,s_7>,<s_7,c_2>\}$。

### 1.3　流量优化模型

对于流量矩阵中任意 OD 流,探讨源交换机和目的交换机是否处于相同控制域,产生域内通信和域间通信 2 种情况。在此基础上,分析控制消息流量和业务流量所在路由对链路负载情况的影响,构建优化模型。

定义二值变量$\eta(i,j)$描述交换机$i$到交换机$j$是否存在 OD 流:

$$\eta(i,j)=\begin{cases}1, & m(i,j)\neq 0;\\ 0, & \text{其他。}\end{cases} \tag{5}$$

1.3.1 域内流请求

约定所有OD流为新流，则源交换机均向所属控制器发送流请求。交换机$k$向所属控制器rf($k$)按照最佳路由$p(k,\mathrm{rf}(k))$传递流请求数据，则链路$e(i,j)$所承载的消息流量为

$$\mathrm{req}(e(i,j))=\sum_{k\in S}\sum_{l\in S}\eta(k,l)\,x_{(k,\mathrm{rf}(k))}^{(i,j)}Q_1。\tag{6}$$

跨域通信过程中，相邻域直接相连的交换机获得相邻域的转发数据后，会向所属控制器发送流请求。因此，数据流每到新的控制域时，都会产生流请求事件。用$\mathrm{ids}(k,l)$表示路由$p(k,l)$中经过的域间交换机集合。则流请求控制消息使得链路$e(i,j)$所承载的消息流量为

$$\mathrm{req}(e(i,j))=\sum_{k\in S}\sum_{l\in S}\eta(k,l)\,x_{(k,\mathrm{rf}(k))}^{(i,j)}Q_1+\sum_{k\in S}\sum_{l\in S}\sum_{\substack{v\in\mathrm{ids}(k,l)\\ \mathrm{rf}(k)\neq\mathrm{rf}(v)}}\eta(k,l)\,x_{(v,\mathrm{rf}(v))}^{(i,j)}Q_1。\tag{7}$$

1.3.2 域内转发实体

当控制器收到交换机流请求时，根据网络状态规划最优路径，并向路径上每个交换机下发转发实体消息。则链路$e(i,j)$所承载的消息流量为

$$\mathrm{ent}(e(i,j))=\sum_{k\in S}\sum_{l\in S}\sum_{v\in n_p(k,l)}\eta(k,l)\,x_{(v,\mathrm{rf}(v))}^{(i,j)}Q_2。\tag{8}$$

1.3.3 域内轮询状态

定期轮询控制域内所有交换机状态，有助于控制器制定准确的转发策略，域内交换机按控制路径向控制器报告当前状态。轮询与OD流量无关，约定在单位时间$t$内轮询一次。则链路$e(i,j)$所承载的消息流量为

$$\mathrm{que}(e(i,j))=\sum_{k\in S}x_{(k,\mathrm{rf}(k))}^{(i,j)}Q_3。\tag{9}$$

1.3.4 域间状态同步

为控制器之间规划最佳路由，需要每个控制器定时同步当前域内网络状态给其他控制器，确保网络状态一致性。状态同步与OD流量无关，约定在单位时间$t$内同步一次。则链路$e(i,j)$所承载的消息流量为

$$\mathrm{syn}(e(i,j))=\sum_{k\in C}\sum_{l\in C}x_{(k,l)}^{(i,j)}Q_4。\tag{10}$$

域内流请求、域内转发实体、域内轮询状态、域间状态同步均采用固定路由，其最佳路由采用Dijkstra算法实现。

链路$e(i,j)$所承载的控制消息流量

$$f(e(i,j))=\mathrm{req}(e(i,j))+\mathrm{ent}(e(i,j))+\mathrm{que}(e(i,j))+\mathrm{syn}(e(i,j))。\tag{11}$$

1.3.5 OD流路由

OD流根据最优路由传递数据，则链路$e(i,j)$所承载的OD流量为

$$g(e(i,j))==\sum_{k\in S}\sum_{l\in S}x_{(k,l)}^{(i,j)}m(k,l)。\tag{12}$$

综上，链路$e(i,j)$所承载的总流量$o(e(i,j))$为控制消息和非控制消息的链路流量之和，即

$$o(e(i,j))=f(e(i,j))+g(e(i,j))。\tag{13}$$

1.3.6 控制器负载

控制器$c_j$的容量为$U(j)$，标识控制器可以同时处理交换机的流请求大小，所有控制器容量上限均为$\hat{U}$。控制器负载等于所有OD流在域内通信和域间通信产生的流请求，即

$$U(j)=\sum_{k\in S}\sum_{l\in S}\eta(k,l)\,Q_1+\sum_{k\in S}\sum_{l\in S}\sum_{\substack{v\in\mathrm{ids}(k,l)\\ \mathrm{rf}(k)\neq\mathrm{rf}(v)}}\eta(k,l)\,Q_1。\tag{14}$$

1.3.7 链路利用率

链路利用率是链路承载的流量占链路总容量的比例，用$\mu(e(i,j))$表示链路$e(i,j)$的链路利用率，即

$$\mu(e(i,j))=\frac{o(e(i,j))}{\sigma(e(i,j))}\leqslant 1。\tag{15}$$

流量工程的路由优化目标是均衡控制器负载和链路负载，用最小化控制器极差率和最大链路利用率表示，即目标值

$$\Phi=\alpha*\frac{U_{\max}-U_{\min}}{\hat{U}}+(1-\alpha)\,\mu_{\max}, \tag{16}$$

式中:$U_{\max}$和$U_{\min}$分别表示所有控制器负载中最大负载和最小负载,两者越接近整个控制器负载越均衡;$\mu_{\max}$表示所有链路利用率最大值;$\alpha$表示平衡因子,$0 \leqslant \alpha \leqslant 1$。

模型优化目标是最小化$\Phi$,需要约束交换机与控制器从属关系、控制器处理流请求容量、链路容量和流量守恒等条件,结合一般流量工程问题求解方案[15],建立如下模型及约束:

$$\min\ \Phi \tag{17}$$

s.t.

$$\sum_{c_j\in C} r(i,j)=1,\ \forall s_i\in S; \tag{18}$$

$$\sum_{e(i,j)\in E} x_{(i,j)}^{(s,t)}-\sum_{e(j,i)\in E} x_{(j,i)}^{(s,t)}=\begin{cases}1, & i=s;\\ -1, & i=t,\ \forall i,j,s,t\in S;\\ 0, & \text{其他},\end{cases} \tag{19}$$

$$\mu(e(i,j))\leqslant u_{\max},\ \ e(i,j)\in E,\ \ \forall i,j\in S; \tag{20}$$

$$U(j)\leqslant \hat{U},\ \ \forall c_j\in C; \tag{21}$$

$$\mathrm{ids}(k,l)\subseteq D(\mathrm{rf}(k)),\ \ \forall m(k,l)\in \boldsymbol{M},\ \ \mathrm{rf}(k)==\mathrm{rf}(l)); \tag{22}$$

$$Q_1,Q_2,Q_3,Q_4>0。 \tag{23}$$

式(17)是最小化目标函数;式(18)表示一个交换机只从属一个控制器;式(19)表示流量守恒[15];式(20)表示链路的容量约束,链路容量利用率不超过其最大值;式(21)表示控制器容量不超过所属控制器处理上限;式(22)表示域内通信不允许跨域;式(23)相关变量为正数。

## 2　两层路由算法

两层路由算法旨在解决跨域通信中如何为OD流设计最佳路由,均衡控制器负载和链路流量负载,降低网络通信代价。基本思想是整个网络抽象为上层网络和下层网络的两层拓扑,上层网络将控制域抽象为一个节点,相邻控制域通过节点相邻表示,其中节点用控制域编号表示;下层网络为原始网络。上层网络的域间路由是均衡控制器的负载;下层网络根据上层网络计算的域间路由,确定对应控制域的域内路由,以均衡链路负载。

### 2.1　相关定义

#### 2.1.1　域间交换机

如果交换机$i$与交换机$j$分别处于相邻的2个控制域,且交换机$i$与$j$存在链路,则称为域间交换机。用$\mathrm{doms}_i$表示控制域$i$的域间交换机集合。如图1中,控制域01的域间交换机节点为$\mathrm{doms}_1=\{s_3,s_4\}$,控制域02的域间交换机节点为$\mathrm{doms}_2=\{s_5,s_6\}$。

#### 2.1.2　域间链路

将相邻控制域中域间交换机之间的链路称为域间链路,相邻控制域可以通过它们实现域间路由。用$\mathrm{doml}(i,j)$表示控制域$i$与$j$的域间链路集合,如图1中,控制域01与控制域02存在3条域间链路,$\mathrm{doml}(1,2)=\{<s_3,s_5>,<s_4,s_5>,<s_4,s_6>\}$。

#### 2.1.3　相邻域路由

相邻域路由指2个相邻域的路由,计算方法约定为某个域内的源交换机到其相邻域的域间交换机的最短路由,且其相邻域的域间交换机满足这2个相邻域的域间链路条件。如图1中,计算控制域01到控制域02路由,假定选择交换机$s_2$为源交换机,即计算$s_2$到控制域02的域间交换机$s_5$和$s_6$的最优路由集,表示为$p(1,2)=\{(<s_2,s_4>,<s_4,s_5>),(<s_2,s_4>,<s_4,s_6>),(<s_2,s_1>,<s_1,s_3>,<s_3,s_5>)\}$。

#### 2.1.4　路由设计原则

1)减少跨域次数。跨域通信会增加相应控制器的负载,增加控制通信代价,影响全局网络的稳定性。

2)域内路由独立规划。每个控制器独立地规划域内路由和域间路由的下一个转发控制域,符合分布式

控制平面的要求。

### 2.2 TLR算法

TLR算法包括上层网络路由(upper-layer network routing,ULNR)和下层网络路由(lower-layer network routing,LLNR),通过均衡控制器流请求量和全局链路容量利用率,实现整个网络的负载均衡。ULNR算法在建立域间路由时,优先选择负载低的控制器作为下一个转发控制域;LLNR算法实现域内交换机路由,将链路流量低的交换机作为下一个转发节点。

#### 2.2.1 ULNR算法

抽象控制域、相邻控制域分别为上层网络的节点和边,则上层网络表示为图 $G_{up}(V_{up},E_{up})$,其中 $V_{up}=\{dom_1,dom_2,\cdots,dom_n\}$,控制域编号表示顶点;集合 $E_{up}$ 中元素表示相邻控制域存在一条边连接。在计算控制域之间路由时,将控制器负载作为节点权重,使用深度优先遍历(depth first search,DFS)算法,使得路由上的对应控制器负载之和较低。

**算法1** 上层网络抽象算法

输入:网络拓扑 $G(V,E)$、控制域 $\{dom_1,dom_2,\cdots,dom_n\}$。

输出:上层网络 $G_{up}(V_{up},E_{up})$。

Step 1 初始化 $\boldsymbol{A}_{n\times n}$ 矩阵。

Step 2 遍历控制域 $i$、$j$,$i=1,\cdots,n$,$j=i+1,\cdots,n$。

Step 3 判断控制域 $i$ 中的任意交换机 $o$ 与控制域 $j$ 中的任意交换机 $d$ 是否存在边,若存在,则标记邻接域矩阵 $\boldsymbol{A}[i][j]=\boldsymbol{A}[j][i]=1$。

Step 4 如果控制域遍历未结束,则继续Step 2。

Step 5 定义 $G_{up}(V_{up},E_{up})$,其中节点集 $V_{up}$ 用控制域编号表示,其权重值初始为0,边集合 $E_{up}$ 通过矩阵 $\boldsymbol{A}_{n\times n}$ 表示控制域相邻关系。

Step 6 算法结束。

控制域的域间交换机集合、域间链路集合计算方法与算法1中Step 3相同,分别建立控制域和交换机、链路的映射关系。

**算法2** ULNR算法

输入:上层网络 $G_{up}(V_{up},E_{up})$、控制域 $i$ 和 $j$。

输出:最优路由 $p_{up}(i,j)$。

Step 1 初始化路由 $p_{up}(i,j)$。

Step 2 使用DFS算法递归执行,其中参数为起点 $i$、终点 $j$、节点权重之和load、路由集合 $p_{current}(i,j)$。算法在选择下一个遍历节点时,贪婪地选择节点(控制器域)负载较低的节点作为下一个选择节点,并运用剪枝方法加速优化。

Step 3 DFS算法递归执行,直到起点等于终点控制域 $j$,输出路由 $p_{current}(i,j)$。

Step 4 判断当前路由 $p_{current}(i,j)$ 是否最优,与 $p_{up}(i,j)$ 进行比较,是否满足:

**条件a** 路由节点数少(跨域个数少)、节点权重(控制器负载)之和小;

**条件b** 节点权重与路由节点数比值小。

选择满足条件a,或者不满足条件a但满足条件b的路由更新为最优路由 $p_{up}(i,j)$。

Step 5 若递归未结束,则继续Step 2。

Step 6 输出最优路由 $p_{up}(i,j)$。

Step 7 算法结束。

#### 2.2.2 LLNR算法

针对跨域OD流量,根据ULNR算法求得域间路由序列,依次计算相邻域路由,最终形成域间路由。首先提出相邻域路由(adjacent domain routing,ADR)算法,再基于此实现LLNR算法。在计算域内路由和相邻路由时,考虑链路负载均衡情况,定义链路总容量等于已占用部分与剩余部分的总和。而链路剩余容量越多,选择作为转发链路的可能性越大。从而将链路中已占用容量作为链路权重,基于Dijkstra算法实现链路

负载率较低的最短路由。为了遵循域内通信原则,计算域内最短路由时,将不属于当前控制域的链路更新为无效。

ADR算法实现控制域$i$内交换机$o$到控制域$j$所有域间交换机中最短路由作为域间路由,先计算交换机$o$到控制域$i$的域间交换机的最短路由,再分别加上控制域$i$和$j$的域间链路,选择链路负载率最低的路由作为最终路由。

**算法 3** ADR算法

输入:网络部署拓扑关系、控制域$i$和$j$、控制域$i$域内交换机$o$。

输出:最优路由$p(o,d_{\min})$、域间交换机$d_{\min}$。

Step 1　初始化最优路由$p(o,d_{\min})$。

Step 2　遍历$\mathrm{doml}(i,j)$域间链路$<u,v>, u \in \mathrm{doms}(i), v \in \mathrm{doms}(j)$。

Step 3　用Dijkstra算法计算交换机$o$到$u$域内最短路由。若路由$p(o,d_{\min})$为空,则更新$p(o,d_{\min}) = p(o,u) \cup e(u,v)$, $d_{\min} = v$。

Step 4　否则,新的路由$p(o,v) = p(o,u) \cup e(u,v)$和路由$p(o,d_{\min})$比较,如果$p(o,v)$的链路权重之和小(链路负载率低),则更新$p(o,d_{\min}) = p(o,v)$,$d_{\min} = v$。

Step 5　若遍历未结束,则继续Step 2。

Step 6　输出最短路由$p(o,d_{\min})$、域间交换机$d_{\min}$。

Step 7　算法结束。

LLNR算法实现流量矩阵$\boldsymbol{M}$中每条OD流路由区分域内和域间通信。如果是域内通信,直接计算最短路由;否则,需要根据ULNR算法计算的域间路由序列,再采用ADR算法计算相邻域路由。为提高全局链路利用率,对流量矩阵$\boldsymbol{M}$中的OD流分类,计算时先域内流,再域间流(见算法4)。

**算法 4** LLNR算法

输入:网络部署拓扑关系、流量矩阵$\boldsymbol{M}$。

输出:每对OD流对应的最佳路由。

Step 1　遍历流量矩阵$\boldsymbol{M}$中的OD流$(m(o,d) \neq 0)$,用冒泡排序算法形成OD序列,域内OD流靠前,域间OD流靠后。

Step 2　根据式(9)(10)更新相应链路的负载。

Step 3　遍历OD流序列$(o,d)$,初始路由$p(o,d)$为空。

Step 4　根据$\boldsymbol{R}$矩阵计算交换机$o$与$d$分别属于的控制域编号$i$、$j$。

Step 5　如果$i$与$j$相等,则说明$o$与$d$处于同一个控制域,使用Dijkstra算法计算域内路由$p(o,d)$。根据式(6)(8)(12)更新相应链路负载,式(14)更新控制器$i$的负载,跳转到Step 10。

否则,用ULNR算法计算域间路由$P_{\mathrm{up}}(i,j)$。

Step 6　遍历$P_{\mathrm{up}}(i,j)$路由元素$<u,v>$,根据式(14)更新控制器$i$的负载。讨论起始、中间、终点控制域3种情况。

Step 7　如果$i==u$,则说明当前$u$是起点控制域,$o_{\min} = o$,根据式(6)更新相应链路负载,继续Step 6。

Step 8　如果$i \neq u, j \neq v$,则说明处于域间控制域。使用ADR算法计算相邻域路由$p(o_{\min},d_{\min})$,域间交换机$d_{\min}$作为新的源点$o_{\min} = d_{\min}$,更新路由$p(o,d)$为$p(o,d) = p(o,d) \cup p(o_{\min},d_{\min})$;根据式(7)(8)更新相应链路负载,继续Step 6。

Step 9　如果$j==v$,则说明到达终点控制域,用Dijkstra算法计算$o_{\min}$到终点交换机$d$域内路由$p(o_{\min},d)$,更新路由$p(o,d) = p(o,d) \cup p(o_{\min},d)$。根据式(8)更新相应链路负载。

Step 10　输出路由$p(o,d)$。

Step 11　若遍历OD没有结束,跳转到Step 2。

Step 12　算法结束。

TLR算法为一对OD流计算路由时,会更改相应链路负载和控制器负载,影响后续OD流的路由计算。从而计算流量矩阵$\boldsymbol{M}$中每对OD流路由时,其处理先后顺序不同,计算结果存在差异。为了提高算法精度,在TLR算法结果上采用启发式算法求解多目标优化问题。

## 3 优化算法

萤火虫算法(firefly algorithm, FA)是Yang等[16]提出的一种新型的群体智能领域算法,其算法能有效解决各类优化组合问题,结构简单且参数较少,易于实现和理解[17]。模型求解各个OD流对应最佳路由,路由号是整数编号,候选解由离散整数序列构成,而传统FA算法主要是解决连续性解空间问题,不能直接用于求解。因此,本文提出一种改进的离散萤火虫算法,通过改进算法求解流程,提升算法求解精度。算法输入变量为流量矩阵,输出结果是所有OD流对应的最佳路由。

### 3.1 相关定义

设流量矩阵$\boldsymbol{M}$中$m(i,j)\neq 0$的OD流个数为$z$。约定每个OD流对应一个路由集合,路由集合相互独立,集合元素为路由编号,代表一条路由。$z$个OD流对应$z$条路由,由$z$位整数构成的序列表示一种可行的路由方案。每个路由集合按路由跳数和链路负载递增排序,即编号越小,目标值$\Theta$越低。

#### 3.1.1 萤火虫位置

萤火虫表示一个候选解,即所有OD流的最佳路由。每位编码表示当前OD流对应的路由编码,$X_i=(x_{i1},x_{i2},...,x_{in})$表示第$i$个萤火虫,其中$x_{ik}$表示第$i$个萤火虫第$k$个OD流对应的路由编号。例如$X_i=(1,2,1,3,4)$表示存在1~5个OD流,第3个OD流对应的路由编号为1,即第3个OD流对应的路由为该路由集合中编号为1的路由。

#### 3.1.2 萤火虫亮度

萤火虫亮度反映其位置的优劣,促使亮度弱的萤火虫逐步向亮度强的移动。

用绝对亮度描述萤火虫初始位置的亮度,一般用目标函数衡量。优化目标函数为$\Theta$,其值越小,说明模型越优,即亮度越强。则第$i$个萤火虫的绝对亮度为

$$I_i=\frac{1}{\Theta(X_i)}。\tag{24}$$

约定不满足约束条件的$\Theta$无穷大,即对应的绝对亮度值为0。

用相对亮度描述2个萤火虫之间的吸引力,随距离增大而减弱,则萤火虫$i$对萤火虫$j$相对亮度表示为

$$I_{ij}=I_i e^{-\varsigma d_{ij}},\tag{25}$$

式中:$\varsigma$表示光吸收系数;$d_{ij}$表示萤火虫$i$到萤火虫$j$之间距离的平方。用路由编号差异性表示萤火虫之间的距离,编码位相同越多,说明距离越近,反之越远。则

$$d_{ij}=\left\| X_i\oplus X_j\right\|^2=\left\|(x_{i1}\oplus x_{j1},x_{i2}\oplus x_{j2},...,x_{in}\oplus x_{jn})\right\|^2,\quad x_{ik}\oplus x_{jk}=\begin{cases}1,\text{if } x_{ik}\neq x_{jk};\\0,\ \text{其他}。\end{cases}\tag{26}$$

例如$X_i=(1,2,1,3,4)$,$X_j=(3,2,7,3,5)$,则距离$d_{ij}=\left\| X_i\oplus X_j\right\|^2=\left\|(1,0,1,0,1)\right\|^2=3$。

#### 3.1.3 萤火虫吸引力

萤火虫亮度与吸引力大小成正比,不同位置的萤火虫间的吸引力存在差异,且随距离增加而减弱,则萤火虫$i$对萤火虫$j$的吸引力表示为

$$\beta_{ij}=\beta_0 e^{-\varsigma d_{ij}},\tag{27}$$

式中,$\beta_0$表示光源位置($r$=0)的初始吸引力,一般取值为1。

#### 3.1.4 萤火虫位置更新策略

亮度强的萤火虫$i$吸引亮度弱的萤火虫$j$向其靠拢,根据路由编号规则得知编号偏小,$\Phi$值偏小,说明最优解对应的路由编号偏小。路由编号皆为整数,则萤火虫位置$X_j$的第$k$维分量在第$t$+1次迭代更新公式为

$$x_{jk}(t+1)=x_{jk}(t)+\Delta\beta_{ij}(x_{ik}(t)-x_{jk}(t))+F(*);\tag{28}$$

$$F(*)=\begin{cases}0,\text{if } \lambda(\text{rand}-0.5)<0;\\-1,\text{if } 0\leqslant\lambda(\text{rand}-0.5)\leqslant\beta_{ij};\\1,\ \text{其他}。\end{cases}\tag{29}$$

式(28)中,$\Delta\beta_{ij}(x_{ik}(t)-x_{jk}(t))$表示第$k$维分量的差值$(x_{ik}(t)-x_{jk}(t))$随相对亮度$\beta_{ij}$变化,其返回结果为整

数。式(29)中,函数$F(*)$表示根据$\lambda(\text{rand}-0.5)$随机值与$\beta_{ij}$大小关系返回常量0、-1、1,其中$\lambda\in[0,1]$表示步长因子,rand为0~1均匀分布随机数。

当$x_{ik}\geqslant x_{jk}$,说明萤火虫$j$的第$k$个业务路由编码小,路由上的链路负载比萤火虫$i$的低,无需调整。否则,调整

$$\Delta(\beta_{ij}(x_{ik}(t)-x_{jk}(t)))=\begin{cases}\text{val}, & x_{ik}<x_{jk}\begin{cases}\text{val}\in\{\frac{1}{2}\lceil x_{ik}-x_{jk}\rceil,...,-1\}, & \text{rand}(0,1)<\beta_{ij};\\ \text{val}\in\{x_{ik}-x_{jk},...,\frac{1}{2}\lfloor x_{ik}-x_{jk}\rfloor\}, & \text{rand}(0,1)\geqslant\beta_{ij};\end{cases}\\ 0, & x_{ik}\geqslant x_{jk}。\end{cases}\tag{30}$$

萤火虫位置移动方向由亮度强弱决定,亮度最强的萤火虫$i$不移动,为了避免过早陷入局部最优,需要随机移动变异操作。则亮度最强的萤火虫位置$X_i$的第$k$维分量在第$t$+1次迭代更新公式为

$$x_{ik}(t+1)=x_{ik}(t)+F(*)。\tag{31}$$

上述萤火虫位置更新时,如果第$k$维分量值超过第$k$个OD流的路由编号范围,则取离其最近编号值。

### 3.2　DFA算法

通过重新定义萤火虫算法中萤火虫位置、亮度和移动策略等关键步骤,实现对流量优化模型的离散问题求解。

#### 3.2.1　种群初始化

使用TLR算法计算每个OD流对应的路由集合,用于萤火虫位置编码。编码原则是跨域个数越少、路由跳数越少,路由编码值越小。选择1 000个不同次序OD流的TLR算法结果构建路由集合。每个萤火虫的编码位取随机值,随机值的范围为对应路由集合中编号值,从而生成$H$个萤火虫作为初始种群。

#### 3.2.2　DFA算法流程

**算法5** DFA算法

输入:　网络部署拓扑关系、流量矩阵$\boldsymbol{M}$。

输出:　每对OD流对应的最佳路由。

Step 1　萤火虫种群数量$H$,光吸收系数$\varsigma$,最大吸引力$\beta_0$,最大迭代次数$T_{\max}$,迭代次数$t$=1。根据种群初始算法初始化$X=(X_1,X_2,...,X_H)$。

Step 2　计算第$t$次迭代中每个萤火虫$j$的绝对亮度$I_j(t)$,按从小到大顺序排列萤火虫,亮度最强萤火虫位置$X_{\text{best}}$。

Step 3　依次遍历每个萤火虫$j$,比较是否存在$I_i(t)>I_j(t)$,若不存在,则萤火虫$j$是所有萤火虫中绝对亮度最大的,则依照式(31)执行随机移动变异操作;否则,萤火虫$j$需要向萤火虫$i$移动。

Step 4　分别计算亮度较强的萤火虫$i$到萤火虫$j$吸引力,依照式(28)计算移动后的萤火虫$j$的新位置$X_j'$,从中选择对应绝对亮度最大的新位置作为萤火虫$j$的移动方向。

Step 5　依次比较每个萤火虫$j$移动前后的绝对亮度值,若$I_j(t)>I_j(t+1)$,说明萤火虫移动位置后绝对亮度变弱,不变更位置,否则进行最优方向移动。

Step 6　更新每个萤火虫的位置,更新亮度最强的萤火虫位置$X_{\text{best}}$;若$t<=T_{\max}$,则$t$=$t$+1,继续执行Step 2。

Step 7　输出最优萤火虫位置和最小目标值$\Phi(X_i)$。

Step 8　算法结束。

## 4　实验分析

### 4.1　评价指标

为了检验模型和算法的有效性,从控制链路流量、控制器负载和链路负载等进行实验分析。

#### 4.1.1　平均控制链路流量

平均控制链路流量用于衡量控制消息对流量矩阵的响应程度,与OD流个数相关,则$f_{\text{avg}}$表示为

$$f_{\text{avg}}=\frac{\sum_{k\in S}\sum_{l\in S}f(e(k,l))}{\sum_{k\in S}\sum_{l\in S}\eta(k,l)}。\tag{32}$$

4.1.2 控制器负载均衡

计算域间路由时,每个控制器接收流请求尽可能均衡,以均衡控制器之间的负载,强化网络的管控效率,用极差$\varphi$衡量:

$$\varphi=U_{\max}-U_{\min}。\tag{33}$$

为衡量极差$\varphi$的波动范围,通过控制器负载率$\varphi'$表示:

$$\varphi'=\frac{\varphi}{U_{\max}},\tag{34}$$

$\varphi'$值越小说明极差值越小,负载越均衡。

4.1.3 链路负载均衡

计算OD流量路由时,以当前链路已占用的流量为权重,计算一条链路负载较低的路由,有效避免链路拥塞。用链路利用率平方差$\delta$衡量,

$$\delta=\sqrt{\frac{1}{(|E|-1)}\sum_{k\in S}\sum_{l\in S}(\mu(e(k,l))-\bar{\mu})^2},\tag{35}$$

式中:平均链路利用率$\bar{\mu}=\frac{1}{|E|}\sum_{k\in S}\sum_{l\in S}\mu(e(k,l))$;$|E|$为链路条数。

### 4.2 数据集

为了验证优化模型和算法的可行性,使用开源ABILENE网络、GEANT网络的拓扑和流量数据[18]验证。网络拓扑信息如表2所示。使用控制器部署算法[19]将ABILENE网络分为2个控制域,GEANT网络分为4个控制域。

**表2 网络拓扑**

**Table 2 Network topology**

| 网络 | 节点数 | 链路数 | 流量矩阵数 | 采集流量时间范围 |
|---|---|---|---|---|
| ABILENE | 12 | 15 | 288 | 2004-08-01(00:00~23:55) |
| GEANT | 22 | 36 | 96 | 2005-05-05(00:00~23:55) |

### 4.3 实验环境

实验算法使用Java语言开发,在运行环境(CPU 8核,内存16 G,Win11)上测试。

流量模型参数结合文献[10]和运行结果经验值设置,如表3所示。

**表3 优化模型参数**

**Table 3 Optimization of model parameters**

| 参数 | 取值 |
|---|---|
| 链路容量上限$\sigma(e(i,j))$ | 40 Gbps |
| 控制器容量$\hat{U}$ | 512 Mbps |
| PACKET_IN消息$Q_1$ | 128 kbps |
| PACKET_OUT消息$Q_2$ | 256 kbps |
| ECHO消息$Q_3$ | 256 kbps |
| SYNCHRONIZATION消息$Q_4$ | 256 kbps |

DFA算法参数结合文献[16]和运行结果经验值设置,如表4所示。

TLR算法和DFA算法对流量矩阵$\boldsymbol{M}$运行多次,取目标值最低的路由作为最优解分析实验结果。SDN的

控制平面决策分为集中式和分布式,集中式是采用一个超级控制器管理网络和规划路由,本文的分布式决策是各个控制器共同管理网络和单独规划路由,均采用DFA算法求解最佳路由。

**表 4　DFA 算法参数**

**Table 4　Parameters of DFA algorithm**

| 参数 | 取值 |
|---|---|
| 萤火虫种群数量 $H$ | 200 |
| 最大迭代次数 $T_{max}$ | 500 |
| 初始吸引力 $\beta_0$ | 1 |
| 光吸收系数 $\varsigma$ | 1 |
| 步长因子 $\lambda$ | 0.25 |

### 4.4　控制器负载分析

控制器负载极差 $\varphi$ 衡量控制器之间负载的差距,值越小说明越均衡。图 2、3 分别比较 2 种网络的 24 h 流量矩阵的控制器负载结果,大多数情况下 DFA 算法的 $\varphi$ 均小于 TLR 算法,$\varphi$ 值分别平均降低 32%、28%,说明 DFA 算法对 TLR 算法优化精度有较好改进。

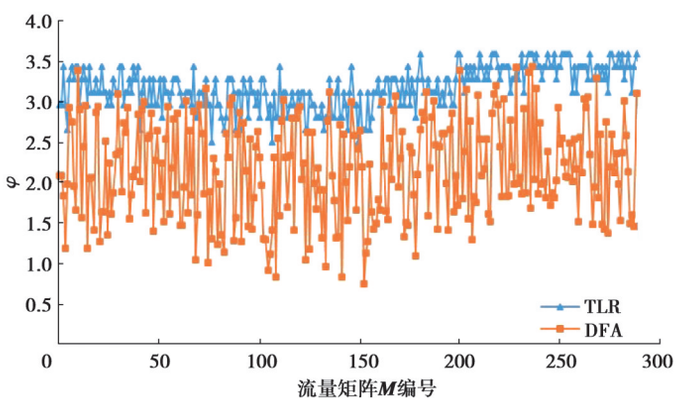


**图 2　ABILENE 网络控制器负载对比**

**Fig. 2　Comparison of $\varphi$ in ABILENE**

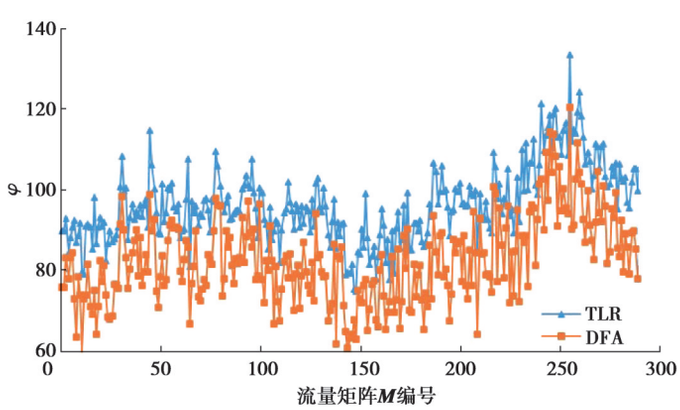


**图 3　GEANT 网络控制器负载对比**

**Fig. 3　Comparison of $\varphi$ in GEANT**

图 4 中控制器负载率 $\varphi'$ 均低于 19%,说明控制器之间负载差距较小,满足模型要求。再次说明基于 TLR 算法结果的启发式算法 DFA 求解结果相对较好。图 5 中分布式的平均控制器负载率相比集中式,降低约

47.3%,是因为多个控制器独立管理域内流请求和网络信息,处理流量数据量相对均衡。

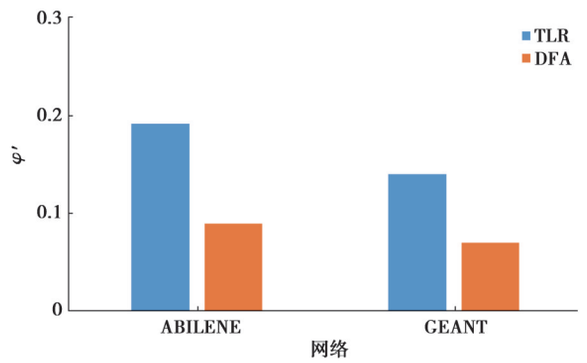


图 4 控制器负载率对比

Fig. 4 Comparison of $\varphi'$ in ABILENE and GEANT

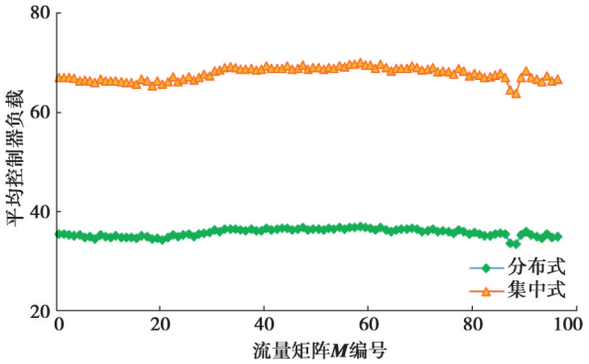


图 5 GEANT 网络平均控制器负载率对比

Fig. 5 Comparison of average controller load rate of GEANT

### 4.5 控制链路流量分析

图 6 显示控制流量 $f$ 随 OD 个数增加的变化过程,TLR 算法处理前 60 个 OD 流时,由于是域内通信不会产生域间流量,$f$ 变化幅度较小;反之,当大于 60,处理域间流量时,额外增加域间流请求和转发实体消息流量,$f$ 变化幅度相对较大。说明控制流量随 OD 个数增加,受域间流量的数量影响较大。图 7 显示平均控制流量与 OD 个数增加变化情况。由于 TLR 算法先计算域内状态轮询、域间控制器同步控制消息,产生较大初始控制流量;随着 OD 流量数增加,平均控制流量 $f_{avg}$ 逐渐降低,当 OD 个数大于 23 时,$f_{avg}$ 小于 1,说明控制流量平均产生流量增量平稳,控制器能及时响应 OD 流。

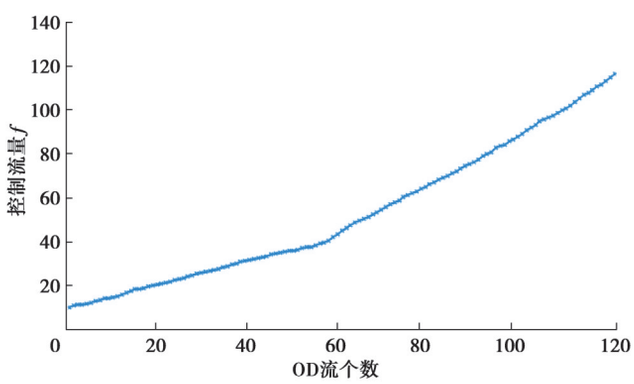


图 6 ABILENE 网络控制流量变化过程

Fig. 6 The change process of $f$ in ABILENE

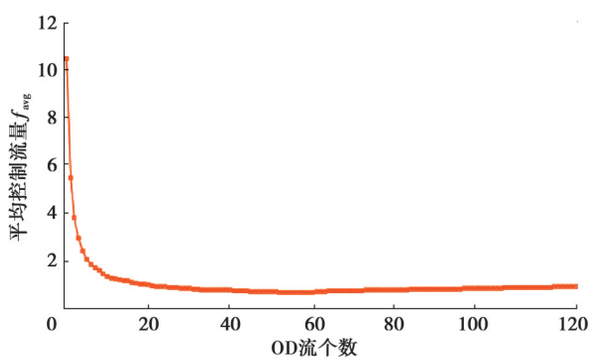


图 7 ABILENE 网络平均控制流量变化过程

Fig. 7 The change process of $f_{avg}$ in ABILENE

图 8 为 DFA 算法求得控制链路流量占整个链路流量比例,其平均值为 2.2%,平均流量为 110.8 Mbps,表明计算 OD 流路由时需要考虑控制消息流量,特别是链路容量不足情况下,易引发控制信息延迟和网络拥塞。此外,控制链路流量与 OD 个数和流量大小有关,OD 个数和流量越小,控制链路流量比重越大。

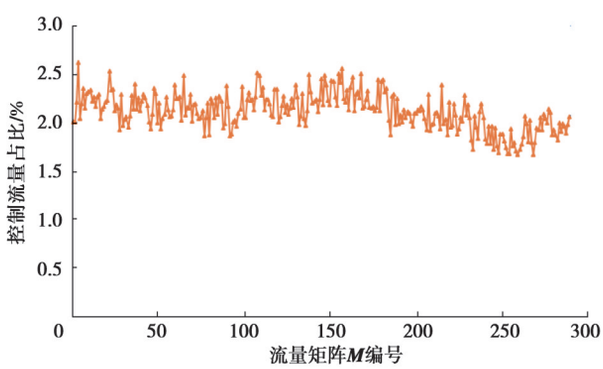


图 8 ABILENE 网络控制流量占比

Fig. 8 The proportion of network control traffic in ABILENE

### 4.6　链路负载分析

链路负载均衡通过链路利用率方差 $\delta$ 和最大链路利用率 $\mu_{max}$ 体现，$\delta$ 越小，链路负载越均衡。图 9 中分布式决策的 $\delta$ 值最大为 10.6%，说明 DFA 算法求解的链路之间负载相差不大，呈均衡效果。图 10 中分布式决策的 $\mu_{max}$ 最大值为 50.4%，大部分值在 40% 左右，是由于 GEANT 网络 OD 流较大，最大流量达到 3 259.6 Mbps。集中式决策的链路利用率方差 $\delta$ 和最大链路利用率 $\mu_{max}$ 均小于分布式决策结果，是由于整个网络视为一个控制域，所有 OD 流路由均为域内通信，计算结果最优，但控制器负载过大。而本文的分布式决策同时优化控制器和链路的负载，且链路负载值平均值与集中式相比，分别相差 2.7% 和 14.37%，整体链路负载均衡。

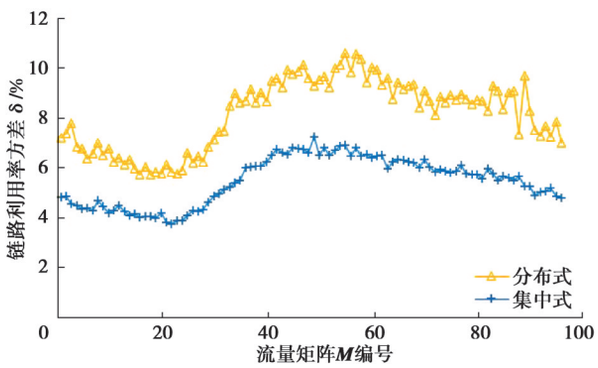


图 9　GEANT 网络链路利用率方差对比

Fig. 9　Comparison of $\delta$ in GEANT

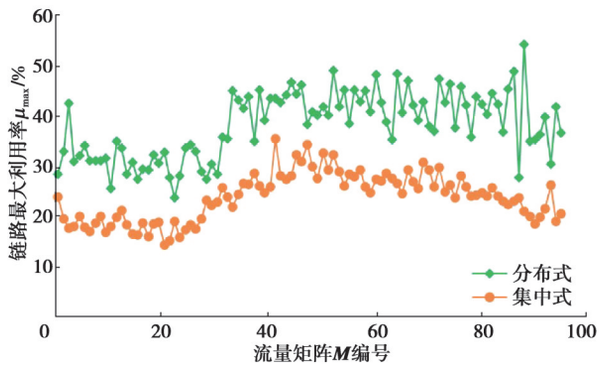


图 10　GEANT 网络链路最大利用率对比

Fig. 10　Comparison of $\mu_{max}$ in GEANT

### 4.7　控制域数量影响分析

图 11 表明在控制域划分确定的情况下，控制器负载极差 $\varphi$ 随控制域个数增加而升高，是因为控制域个数增加导致域间流请求数量增加，加大网络中心控制域的负载，边缘和中心控制器之间的负载不平衡加剧。

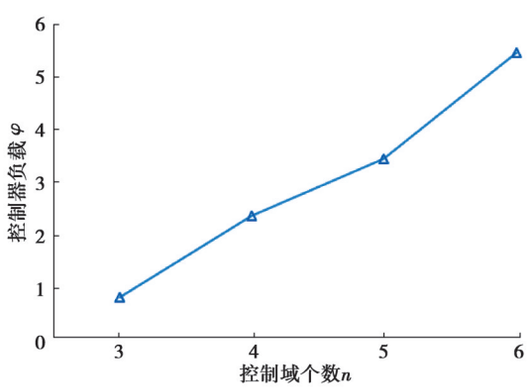


图 11　GEANT 网络控制器负载与控制域个数关系

Fig. 11　The relationship between $\varphi$ and $n$

## 5　结束语

引入控制消息流量研究 SDN 网络流量工程问题，建立优化控制器负载和链路负载的均衡性的目标模型，实现分布式控制平面的多域流量路由优化决策。为求解域间流量路由，提出一种 TLR 路由算法；为提高模型求解精度，提出一种改进的 DFA 算法优化结果。实验结果表明，与集中式控制机制相比，文中提出的分布式决策能实现控制器和链路的负载同时优化。目前研究局限于常见控制消息，需要进一步探索其他关键控制消息。不确定性流量更符合实际需求，需要进一步研究流量动态性和预测性。未来研究将深入分析控制平面消息构成，思考不确定集流量矩阵下的流量工程鲁棒性以及基于机器学习的流量预测。

## 参考文献


[1] 周桐庆, 蔡志平, 夏竟, 等 . 基于软件定义网络的流量工程[J]. 软件学报, 2016, 27(2): 394-417.
Zhou T Q, Cai Z P, Xia J, et al. Traffic engineering for software defined networks[J]. Journal of Software, 2016, 27(2): 394-417.(in Chinese)

[2] 张少军, 兰巨龙, 胡宇翔, 等 . 软件定义网络控制平面可扩展性研究进展[J]. 软件学报, 2018, 29(1): 160-175.
Zhang S J, Lan J L, Hu Y X, et al. Survey on scalability of control plane in software-defined networking[J]. Journal of Software, 2018, 29(1): 160-175.(in Chinese)

[3] Mohammadi R, Akleylek S, Ghaffari A, et al. Taxonomy of traffic engineering mechanisms in software-defined networks: a survey[J]. Telecommunication Systems, 2022(81): 475-502.

[4] Akyildiz I F, Lee A, Wang P, et al. Research challenges for traffic engineering in software defined networks[J]. IEEE Network, 2016, 30(3): 52-58.

[5] Mendiola A, Astorga J, Jacob E, et al. A survey on the contributions of software-defined networking to traffic engineering[J]. IEEE Communications Surveys & Tutorials, 2017, 19(2): 918-953.

[6] Agarwal S, Kodialam M, Lakshman T V. Traffic engineering in software defined networks[C]//2013 Proceedings IEEE INFOCOM. IEEE, 2013: 2211-2219.

[7] Guo Y Y, Wang Z L, Liu Z F, et al. SOTE: Traffic engineering in hybrid software defined networks[J]. Computer Networks, 2019, 154: 60-72.

[8] Ammal R A, Pc S, Ss V. Termite inspired algorithm for traffic engineering in hybrid software defined networks[J]. PeerJ Computer Science, 2020, 6: e283.

[9] Wang C H, Ni H, Liu L. A routing strategy with optimizing linear programming in hybrid SDN[J]. IEICE Transactions on Communications, 2022, E105.B(5): 569-579.

[10] Wang X, Deng Q, Ren J, et al. The joint optimization of online traffic matrix measurement and traffic engineering for software-defined networks[J]. IEEE/ACM Transactions on Networking, 2020, 28(1): 234-247.

[11] Huang J J, Chen Y Y, Chen C E, et al. Weighted routing in hierarchical multi-domain SDN controllers[C]//2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2015: 356-359.

[12] Sridharan V, Gurusamy M, Truong-Huu T. Multi-controller traffic engineering in software defined networks[C]//2017 IEEE 42nd Conference on Local Computer Networks (LCN). IEEE, 2017: 137-145.

[13] Zhao L P, Hua J Y, Liu Y Y, et al. Distributed traffic engineering for multi-domain software defined networks[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019: 492-502.

[14] Hua J Y, Zhao L P, Zhang S H, et al. Topology-preserving traffic engineering for hierarchical multi-domain SDN[J]. Computer Networks, 2018, 140: 62-77.

[15] Sanvito D, Filippini I, Capone A, et al. Clustered robust routing for traffic engineering in software-defined networks[J]. Computer Communications, 2019, 144: 175-187.

[16] Yang X S. Firefly algorithms for multimodal optimization[C]//International Symposium on Stochastic Algorithms. Berlin: Springer, 2009: 169-178.

[17] Li J, Wei X Y, Li B, et al. A survey on firefly algorithms[J]. Neurocomputing, 2022, 500: 662-678.

[18] Li D Y, Xing C Y, Dai N Y, et al. Estimating SDN traffic matrix based on online adaptive information gain maximization method[J]. Peer-to-Peer Networking and Applications, 2019 (12): 465-480.

[19] 王坤, 吕光宏, 胥林, 等 . 基于网络划分的SDN分布式控制器部署[J]. 重庆大学学报, 2020, 43(9): 81-92.
Wang K, Lu G H, Xu L, et al. Distributed controller placement in SDN based on network partitioning[J]. Journal of Chongqing University, 2020, 43(9): 81-92.(in Chinese)


（编辑　吕建斌）