

# 基于加权图的链路映射算法

高明 周慧颖 焦海 应丽莉

浙江工商大学信息与电子工程学院 杭州 310018

(gaoming@zjgsu.edu.cn)

**摘要** 服务功能链(Service Function Chain,SFC)作为一种服务部署概念,为网络提供了更高的灵活性。文中研究服务功能部署中的映射问题,针对服务功能链的业务编排平面部署提出一种基于加权图的链路映射算法,来平衡功能服务节点部署到物理节点上的负载要求。给出了一种服务功能虚拟链路的映射算法,即先进行服务功能组合,随后针对实际的链路情况进行建模分析,利用效率矩阵求解初值,最后利用启发式算法对前者进行纠正。通过建模分析,并与降低链路带宽需求的图匹配策略的特征向量分解算法进行对比,该算法可以在链路节点负载和链路带宽均衡的情况下完成服务请求,并且在服务链长度不断增长和流量数增加的过程中,算法对于吞吐量的变化更加稳定,可以降低对于现有物理网络进行映射的代价。

**关键词**:网络功能虚拟化;软件定义网络;服务功能链;服务部署;映射

**中图法分类号** TP393

## Link Mapping Algorithm Based on Weighted Graph

GAO Ming,ZHOU Hui-ying,JIAO Hai and YING Li-li

School of Information and Electronic Engineering,Zhejiang Gongshang University, Hangzhou 310018, China

**Abstract** Service function chain (SFC), as a concept of service deployment, provides a higher flexibility for network. This paper studies the mapping problem in the service function deployment, and proposes a link mapping algorithm based on weighted graph for the service function chain's business choreography plane deployment, so as to balance the load requirements of the functional service nodes deployed to the physical nodes. And this paper presents a mapping algorithm for virtual link of service function, that is, the combination of service function is first carried out, and then the actual link situation is modeled and analyzed, the initial value is obtained by using efficiency matrix, and finally the former is corrected by using heuristic algorithm. Through the modeling analysis and comparison with Eigen decomposition of adjacency matrices (Eigen) of map matching strategy which reduces the link bandwidth demand, the algorithm can complete the service request under the condition of balanced link node load and link bandwidth, and in the growing service chain length and flow, the algorithm is more stable to changes in throughput and can reduce the cost of mapping for existing physical network.

**Keywords** Network function virtual, Software define network, Service function chain, Service deploy, Mapping

## 1 引言

软件定义网络(Software Defined Network, SDN)改变了传统网络的架构方式,将数据层面和转发层面分离,可以实现对网络流量的控制。网络功能虚拟化(Network Functions Virtualization, NFV)通过使用 x86 等通用性硬件以及虚拟化技术,可以降低网络昂贵的设备成本,并且通过软硬件解耦及功能抽象,使网络设备功能不再依赖于专用硬件,可以实现新业务的快速开发和部署。SDN 和 NFV 两种技术结合,可以有效提高网络资源部署的灵活性。服务功能链作为一种服务部署概念被提出,为未来的运营商网络提供了更高的灵活性和成本效益,让数据流可以在节点上自然流动,却不会被不同

节点上的不同服务干预。一个典型的 SFC 体系架构通常是在 NFV 平台的基础上部署 SDN 控制器, SFC 可以利用 SDN-Overlay 技术,当 SFC 请求到来时, SDN 控制器中的编排模块会根据服务功能链和底层物理网络的资源状态进行部署<sup>[1]</sup>。通过使用 SDN 和 NFV 可以使服务功能在不依赖于固定物理拓扑的前提下,实现服务功能的灵活增加、删除和更改。典型的动态 SFC 架构图主要由编排层、控制层、数据层<sup>[2-3]</sup>组成。图 1 是 SFC 体系根据网络服务进行的一个典型的实例: SFC 体系依据网络当前所能提供的服务功能,根据分类器或者网络管理软件平台进行分析,数据流量根据流量的分类获取数据流量的服务策略,之后根据服务路径计算进行指导,流量会经过合适的服务路径完成服务链的处理。例如

基金项目:浙江省基础公益研究计划(LGG20F010005);国家重点研发计划基金(2017YFB0803202);国家自然科学基金(61871468);浙江省重点研发计划基金(2019C01056)

This work was supported by the Research Plan of Basic Welfare of Zhejiang Province, China(LGG20F010005), National Research Plan Foundation of China(2017YFB0803202), National Natural Science Foundation of China(61871468) and Research Plan Foundation of Zhejiang Province, China(2019C01056).

通信作者:周慧颖(1435561370@qq.com)

从用户客户端发起的请求,要经过负载均衡(LoadBalance, LB)、深度包检查服务(Deep packet inspection, DPI)、入侵检测系统(Intrusion Detection System, IDS)、防火墙(Firewalls, FW)这些服务功能,编排如图1所示。服务链可以安排为两种,一条为LB→IDS→Firewall,另一条为LB→DPI→Firewall。

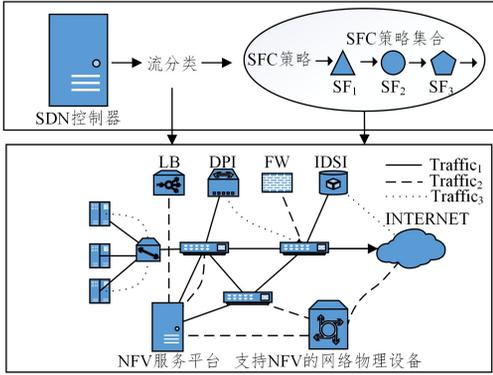


图1 基于SDN/NFV的SFC系统示意图

Fig.1 SFC system schematic diagram based on SDN/NFV

通过使用SDN和NFV,动态SFC不仅在数据流量和增值服务的层面上进行了演进,而且在网络体系架构的角度进行了变革。然而,随着当前网络服务的不断发展,越来越多的服务功能序列和大量的服务功能链导致当前现有的虚拟链路映射算法处理速度跟不上服务请求的增长规模;同时,存在节点和负载链路的平衡问题,使得映射算法不能够很好地满足服务质量的要求,并且服务功能链的部署方法也未能充分考虑网络资源利用率<sup>[4]</sup>。针对上述SFC链路映射中的问题,本文提出了一种基于加权图的链路映射算法,能够一定程度上解决上述问题,还能够提供冗余性和性能保障。

## 2 相关工作

SFC能够满足对于不同服务功能在不同数据流量下的业务处理要求,在SDN/NFV技术加持下的服务功能链映射的技术,是当前网络技术研究的前沿,被广泛研究。文献[5]提出了一种基于软件定义下的OpenBox功能映射架构,即可以通过多种服务请求下对相同服务功能的合并,减少同种服务功能实例的冗余即减少实例个数,同时实现服务功能实例的复用,再对缩减后的服务功能进行映射,但采用这种方式映射的效率不高。文献[6]提出了一种证明方法,即将虚拟功能的映射问题建模成一种整数线性规划问题,在实现最小虚拟功能实例个数的目标下,证明了虚拟功能映射问题是一种NP-hard问题。但这种方式对服务资源消耗过多。文献[7]提出虚拟网络嵌入问题(Virtual Network Embedding Problem, VNEP),它是NP-hard问题的一种。但采用这种方式解决上述问题时,在进行服务节点部署时需要进行一定的妥协,尤其是在解决大规模场景下的虚拟功能的映射问题时。为了解决这个问题,文献[8-9]分别基于贪婪算法、模拟退火算法等启发式算法对不同目标,例如对网络功能实例最小化或对最小化服务节点的数量,可以在较短的时间下得到解决方案的近似最优解。因此,针对虚拟功能映射效率较低以及不能实现有效的资源利用的问题,本文提出了一种基于加权图的链路映射算法,旨在通过充分利用虚拟资源灵活的分配特性<sup>[10]</sup>,

来提高映射性能和资源利用率。

## 3 服务功能映射模型

本文将服务功能映射模型定义为一种多条件约束的优化问题,将网络的物理拓扑建模为一个无向图 $G=(V,E)$ ,用 $V$ 代表物理网络拓扑上的节点,用 $E$ 代表连接这些节点的物理链路。 $C_v$ 表示每个节点 $v(v \in V)$ 具有的处理能力, $C_e$ 表示每条链路 $e(e \in E)$ 可以提供的带宽。 $R$ 表示物理设备为虚拟网络功能可以提供的资源,这些资源包括节点具有的计算能力以及存储资源等。其中对一组服务请求 $R=\{S_1, S_2, \dots\}$ ,第 $i$ 条SFC的服务请求用 $S_i=\{s_{i1}, s_{i2}, \dots\}$ 表示,其所需带宽用 $b_i$ 表示,用 $d_{ij}$ 表示第 $j$ 项服务所需要的处理能力。对于上述模型,我们整个映射过程的目标是将要求高的服务配置到链路情况较好的物理节点上,增大网络状况的鲁棒性,减少资源的无故损耗,从而支持更多服务请求节点的部署。我们将其定义为:

$$\max R \quad (1)$$

$$\text{s. t. } \sum_{i,j,p_{ij}=v_k} d_{ij} \leq C_v, \forall v_k \in V \quad (2)$$

$$\sum_{i,e_k \in l_i} b_i \leq C_e, \forall e_k \in E \quad (3)$$

其中, $p_{ij}=v_k$ 表示我们将功能 $s_{ij}$ 放置到节点 $v_k$ 上<sup>[11]</sup>,物理拓扑中存在链路 $e_k$ 。根据式(1),求 $\max R$ 是一个非具象的公式,需要对其进行具象化。

$$T = \text{var}(\{C_v - \sum_{i,j,p_{ij}} d_{ij} | v_k \in V\}) \quad (4)$$

$$Q = \text{var}(\{C_e - \sum_{i,e_k \in l_i} b_i | e_k \in E\}) \quad (5)$$

式(4)意味着每次部署到节点上的操作,需要去求解处理能力的方差值。式(5)意味着要求解带宽的方差值。为了实现映射时的效率最优,最简单的处理方法就是求解处理能力的方差和带宽的方差两者之和的最小值。但是这样做的问题在于两者的最小值,不一定代表系统的最优解。因此,本文提出一种加权图匹配算法来均衡两者的情况,生成映射方式。我们将最优加权图匹配<sup>[12]</sup>(Optimal Weighted Graph Matching, OWGM)进行如下定义: $G_1=(V_1, E_1)$ 和 $G_2=(V_2, E_2)$ 表示两个具备相同节点数的加权图。如果 $G_1$ 和 $G_2$ 之间存在一个映射函数 $P$ 使得 $G_1$ 通过映射之后与 $G_2$ 的差最小,称该函数为 $G_1$ 和 $G_2$ 的最优加权图匹配函数。其中映射函数是指矩阵中的每一行和每一列都只有一个元素为1,其他元素均为0;加权图的差是指两个加权图的邻接矩阵的差的F-范数<sup>[11]</sup>。假设存在两个图的邻接矩阵为 $E$ 和 $M$ , $\|x\|$ 表示矩阵 $x$ 的F-范数, $L(P)$ 代表经过映射后的差, $L(P)$ 可以用下式表示:

$$\text{Minimize } L(P) = \sqrt{\|PEP^T - M\|} \quad (6)$$

我们采用一种方式简化上述计算,即将加权图匹配问题转化成效率矩阵指派问题<sup>[13]</sup>,首先计算 $M$ 和 $E$ 矩阵的特征向量的绝对值矩阵,计为 $\overline{M}_1, \overline{E}_1^T$ 。然后利用 $H(X)$ 对效率矩阵 $x$ 进行求解,得到指派矩阵。

$$P = H(\overline{M}_1 \overline{E}_1^T) \quad (7)$$

## 4 算法描述

### 4.1 服务功能组合

对于虚拟链路映射,我们首先要实现对多条服务功能链的组合,使其成为虚拟拓扑图。虚拟拓扑是服务请求的链路情况的一种连通关系,我们需要对这个情况进行一定的处理,

连通图的链路表明所需要的带宽以及各自服务功能节点所需要的处理能力。图 2 展示了两个需要组合的服务功能链,我们对其链路和节点能力进行合并。合并方式有很多,这里列举一个普通情况,即合并相同的服务功能,因此需要对两条服务功能链所需要的带宽相加。两个节点之间的连线表示节点之间的带宽,节点后的数字表示节点具有的处理能力。考虑到实际情况,这种相加可能是一种理想情况,在实际情况中不需要满足。

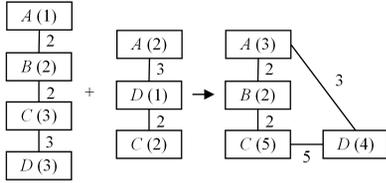


图 2 服务功能链组合成功能拓扑图

Fig. 2 Service function chains combined to form a functional topology

之后根据组合成的功能拓扑图,我们生成其对应的邻接矩阵  $E$ 。 $E$  的对角线元素表示物理节点需要为其服务所具备的处理能力,其余元素表示为其与相邻功能节点之间所需要的带宽能力。物理拓扑图的邻接矩阵称为  $M$ 。对于两个不同的矩阵,需要对  $E$  进行扩展,达到图匹配的顶点数相同的目的。对于  $E$ ,我们需要增加参数 0,将其扩展到和矩阵  $M$  相同的维度。具体如图 3、图 4 所示。

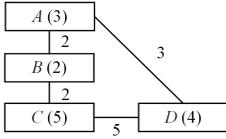


图 3 待匹配的功能拓扑

Fig. 3 Functional topologies to be matched

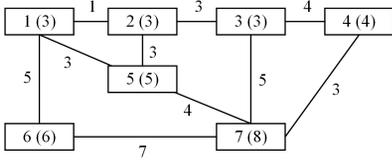


图 4 待匹配的物理拓扑

Fig. 4 Physical topologies to be matched

$$E = \begin{bmatrix} 3 & 2 & 0 & 3 \\ 2 & 2 & 2 & 0 \\ 0 & 2 & 5 & 5 \\ 3 & 0 & 5 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 2 & 0 & 3 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 5 & 5 & 0 & 0 & 0 \\ 3 & 0 & 5 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (8)$$

$$M = \begin{bmatrix} 3 & 1 & 0 & 0 & 3 & 5 & 0 \\ 1 & 3 & 3 & 0 & 2 & 0 & 0 \\ 0 & 3 & 3 & 4 & 0 & 0 & 5 \\ 0 & 0 & 4 & 4 & 0 & 0 & 3 \\ 3 & 2 & 0 & 0 & 5 & 2 & 4 \\ 5 & 0 & 0 & 0 & 2 & 6 & 7 \\ 0 & 0 & 5 & 3 & 4 & 7 & 8 \end{bmatrix} \quad (9)$$

#### 4.2 邻接矩阵替换

由于 4.1 节中对功能矩阵  $E$  进行 0 参扩展,然而对于物

理链路拓扑矩阵  $M$ ,为了完成节点选择物理链路的权值更新,需要对 0 参进行链路替换。因此可以根据最小带宽最大原则对非相邻节点计算权值,进行更新<sup>[14]</sup>,但这种方式没有更新邻接节点的权值,还加大了部分链路的转发跳数。可以采取一种改进的方式,即对任意两个节点重新计算路径,使得最小带宽和传输条数的比值最大,更新权值为带宽跳数值<sup>[15]</sup>。但这种方法改变了原本的链路评价体系,因此本节提出了一种新的算法,不利用  $K$  条最短路径算法,而是利用 Johnson 算法<sup>[16]</sup>进行路由更新。因为本质上, $M$  矩阵的权值更新,其实就是对其进行路由计算,对于解决全源最短路径算法主要的解法就是 Johnson 算法和 Floyd 算法。Floyd 算法可以检测图中的负环,解决给定的加权图中顶点间的最短路径。Johnson 算法虽然和 Floyd 算法类似,但其算法效率更高。

#### 算法 1 物理链路拓扑权值更新算法

Compute  $G'$ , where  $V[G'] \cup \{s\}$  and

$$E[G'] = E[G] \cup \{(s, v), v \in V[G]\}$$

If Bellman-Ford( $G', w, s$ ) = FALSE

Then print "the input graph contain a n re-weight cycle"

Else For each vertex  $v \in V[G']$

Do set  $h(v)$  to the value of  $\delta(s, v)$

Computed by the Bellman-Ford algorithm

For each edge  $(u, v) \in E[G']$

Do  $w'(u, v) \leftarrow w(u, v) + h(u) - h(v)$

For each vertex  $u \in V[G]$

Do run DIJKSTRA( $G, w', u$ ) to compute

$\delta'(s, v)$  FOR all  $v \in V[G]$

For each vertex  $v \in V[G]$

Do  $d_{uv} \leftarrow \delta'(s, v) + h(u) - h(v)$

Return  $D$

算法 1 给出了 Johnson 算法的描述,在给定的图  $G = (V, E)$  中增加一个新的顶点  $s$ ,使它指向图  $G$  中的所有顶点并建立连接,我们称新的图为  $G'$ 。随后我们对图  $G'$  中顶点  $s$  使用 Bellman-Ford 算法来计算单源最短路径。Bellman-Ford 算法是一种通过使用松弛函数对路径的不断松弛来获取最短路径的一种算法。通过该算法的计算,我们得到  $h[\ ] = \{h[0], h[1], \dots, h[V-1]\}$ 。之后对原图  $G$  中的所有边进行“re-weight”,即对每条边  $(u, v)$  计算其新的权值为  $w(u, v) + h[u] - h[v]$ ;最后移除新的顶点  $s$ ,对每个顶点运行 Dijkstra 算法求得最短路径。Bellman-Ford 算法首先创建源顶点  $s$  到图中所有顶点的距离的集合  $distSet$ ,为图中所有的顶点制定一个距离值,初始均为  $Infinite$ ,源顶点距离为 0;计算最短路径,执行  $v-1$  次遍历,对于图中的每条边,如果起点  $u$  距离  $d$  加上边的权值  $w$  小于终点  $v$  的距离  $d$ ,则更新终点  $v$  的距离值  $d$ ;然后检测图中是否有负权边形成了环,遍历图中的所有边,计算  $u$  到  $v$  的距离,如果  $u$  与  $v$  的距离存在更小值,则说明存在环。进行 Johnson 算法之后, $M$  矩阵的替换为:

$$M = \begin{bmatrix} 3 & 1 & 4 & 8 & 3 & 5 & 7 \\ 1 & 3 & 3 & 7 & 3 & 5 & 7 \\ 4 & 3 & 3 & 4 & 7 & 9 & 5 \\ 8 & 7 & 4 & 4 & 7 & 10 & 3 \\ 3 & 3 & 7 & 7 & 5 & 2 & 4 \\ 5 & 5 & 9 & 10 & 2 & 6 & 7 \\ 7 & 7 & 5 & 3 & 4 & 7 & 8 \end{bmatrix} \quad (10)$$

之后,对矩阵  $E$  的填充 0 进行非 0 替换。这是因为下一步的处理要求这两个矩阵近似同构。图同构指的是两个图有相同数目的边和顶点且对应的顶点具有相同的连接性。为了得到结果的最优化,我们需要将需求较多资源的链路匹配到物理拓扑剩余资源较多的链路上。但在下一步匈牙利算法求解初值时,求解不到最优解,需要用启发式算法进行校正。具体来讲,第一步,将  $E$  矩阵中的对角线元素,依据功能需求大小进行排序并进行替换,在  $E$  矩阵中随机分配“0”元素,以满足矩阵同构的要求。然后根据上述替换的原则,对矩阵的边进行替换,完成边的重构。下式是替换后的矩阵  $E$ 。

$$E = \begin{bmatrix} 3 & 2 & 0 & 3 \\ 2 & 2 & 2 & 0 \\ 0 & 2 & 5 & 5 \\ 3 & 0 & 5 & 4 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 5 & 7 & 3 & 7 & 2 & 4 & 3 \\ 7 & 4 & 10 & 4 & 7 & 3 & 8 \\ 3 & 10 & 8 & 7 & 5 & 3 & 4 \\ 7 & 4 & 5 & 6 & 2 & 7 & 9 \\ 2 & 7 & 5 & 2 & 3 & 4 & 3 \\ 4 & 3 & 3 & 7 & 4 & 3 & 5 \\ 3 & 8 & 4 & 9 & 3 & 5 & 3 \end{bmatrix} \quad (11)$$

#### 4.3 匈牙利算法求解初值

在上述操作之后,我们会得到两个同构的邻接矩阵,根据 Example for Weighted Undirected Graph Matching 方法求解出其效率矩阵。第一步求出各自矩阵的特征向量的矩阵,记为  $E_1, M_1$ ; 第二步,求出  $E_1, M_1$  的各自逆矩阵,记为  $\bar{E}_1, \bar{M}_1$ ; 第三步,求出带求解矩阵,  $X = \bar{M}_1 \bar{E}_1^T$ ; 第四步,对其采用式(7),求出其映射矩阵  $P$ 。

$$X = \bar{M}_1 \bar{E}_1^T$$

$$= \begin{bmatrix} 0.846 & 0.778 & 0.861 & 0.785 & 0.829 & 0.682 & 0.696 \\ 0.581 & 0.705 & 0.882 & 0.847 & 0.677 & 0.677 & 0.551 \\ 0.909 & 0.853 & 0.741 & 0.833 & 0.750 & 0.750 & 0.718 \\ 0.790 & 0.986 & 0.777 & 0.790 & 0.764 & 0.764 & 0.895 \\ 0.430 & 0.883 & 0.747 & 0.781 & 0.894 & 0.894 & 0.796 \\ 0.772 & 0.946 & 0.810 & 0.702 & 0.729 & 0.729 & 0.914 \\ 0.378 & 0.855 & 0.758 & 0.864 & 0.884 & 0.884 & 0.698 \end{bmatrix} \quad (12)$$

根据对效率矩阵的求解,将会得到一个初略的分布映射矩阵,即  $P$ 。其行出现数值为 1 的值,代表了功能节点将部署到相应的物理节点上。

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

#### 4.4 启发式算法轮询求解

通过上面的映射求解情况得到了一个简略的映射对应关系,但这种求解可能使某些链路上出现拥塞问题。求解的情况不一定是最优解,因此我们采用了启发式算法对其结果进

行修正。轮询的启发方式算法包括遗传算法(Genetic Algorithm, GA)、蚁群算法(Ant Clony Optimization, ACO)、模拟退火算法(Simulated annealing, SA)等,这里我们采用上述提到的 3 种算法来进行修正。根据分析得出可以依据迭代次数来更新数据  $P$ 。具体算法如算法 2 所示。

#### 算法 2 最优范数值修正算法

Compute  $L(P)$  depend on inital  $E$  and  $M$

for  $i=1$  to  $n$  do

parallel run GA ACO SA algorithm to get

updated  $L(P)$

if updated  $L(P) < \text{inital } L(P)$

update  $L(P)$  Return Minium  $L(P)$

之后我们可以根据更新后得到的  $L(P)$  范式来计算更新后的  $P$  值。

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (14)$$

#### 4.5 算法复杂度分析

我们根据上述 4 个过程,计算映射函数值  $P$ 。通过分析可知,本文主要体现在邻接矩阵替换算法和启发式算法轮询求解。对于前者,算法时间复杂度为  $O(nm \log m)$ 。而对于启发式算法轮询求解,时间复杂度不一致,需要具体问题具体分析。

## 5 仿真实验

为了评估本文提出算法的有效性,本文采用吞吐量和到达率、服务功能实例的 CPU 负载率和带宽利用率作为性能评价指标,并与图匹配策略的特征向量分解算法(Eigendecomposition of adjacency matrices, Eigen)进行比较来体现本文提出的基于加权图的链路映射算法的有效性。

#### 5.1 实验环境

本实验的硬件环境为:处理器 Intel © Xeon® CPU E5-2650 0 @ 2.00 GHz × 32、内存 94.4 GB、图形 llvmpipe (LLVM 6.0, 256 bits)、硬盘 1000GB 的 PC 机。软件环境为:OVS 2.5.5 版本、KVM 2.5.0 版本、一个宿主机系统 Ubuntu 16.04 LTS 64 位、几个虚拟机系统 Ubuntu 16.04 LTS 64 位。虚拟机系统主要用于仿真服务功能节点。

#### 5.2 算法性能对比

在仿真实验中,对于测量指标均以 6 个服务功能作为基准进行测试。对于吞吐量的计算,主要统计流量流经不同服

务实例下的虚拟机的流量总和。服务到达率是计算完成服务链部署的流量数与全部数据流数的比值,服务功能实例的CPU负载率则是计算 VM 虚拟机的 CPU 负载的累计分布函数,带宽利用率也同理。

图 5 是当服务功能链的长度为 6 时,随着流量数的增加 2 种算法的吞吐量的变化曲线图。可以看出,随着流量数的提高,当流量数在 100 条以下时,2 种算法的吞吐量的变化是一致的,之后两者的吞吐量差距在逐渐增大,直到差距为 100 Mbps。这是因为相比 Eigen 算法,本文算法能够将流量更快地加载到服务实例之上,满足更多的服务请求。图 6 同样反映了在流量数增加时服务到达率的变化。可以看到与 Eigen 算法相比,本算法能够实现更高的服务到达率,这主要是由于服务放置能够更快满足流量的部署任务。

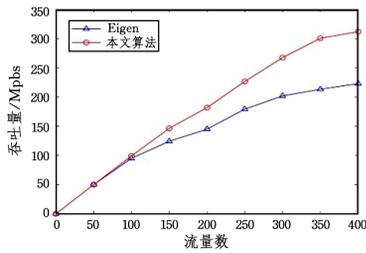


图 5 吞吐量与流量数的关系图

Fig. 5 Graph of throughput versus throughput

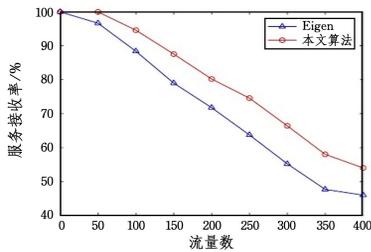


图 6 服务到达率和流量数的关系

Fig. 6 Relationship between service arrival rate and traffic volume

图 7 和图 8 分别反映了可用 CPU 资源的分布情况和链路带宽情况。这里采取 CDF 曲线来表现资源的分布情况。该曲线的左侧反映了网络中的瓶颈问题,曲线中部的曲折情况即斜率反映了资源的分布情况,斜率越大,说明网络资源的分布越均衡。从图 7、图 8 可以看出,本算法中部的曲线斜率与 Eigen 算法的差距较小,本文算法的提高幅度不大,即对 CPU 和链路带宽资源的消耗依然较多,且本文算法的开销情况相较于 Eigen 算法处于曲线的左侧,这意味着其在部署时产生了较多的带宽和缓存开销。之后将对本文提出的算法进行改进来提高 CPU 和链路带宽的利用率。

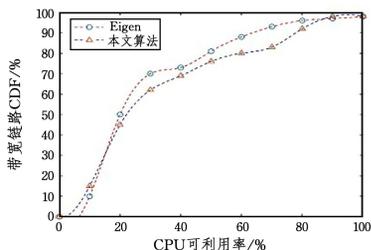


图 7 CPU 可用率

Fig. 7 CPU availability

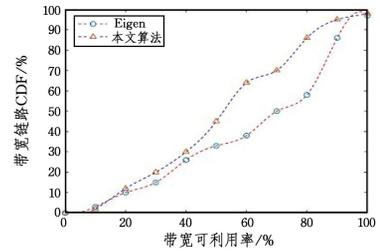


图 8 带宽利用率比较

Fig. 8 Bandwidth availability comparison

**结束语** 本文基于服务链部署研究中的映射问题,提出了一种加权图链路映射算法。通过服务功能和实例拓扑之间的问题抽象,给出了一种数学方法,利用映射矩阵的差值作为评价标准,利用效率矩阵求解初值,最后使用启发式算法进行修正,完成服务功能到实际链路的映射方法,并在仿真环境下进行实验。结果表明,该算法在服务链长度和流量数增加时,链路吞吐量的变化与流量数基本呈正相关,能够较好地满足服务质量的要求。下一步将针对算法存在的局限性进行进一步的优化,对更复杂的网络场景进行研究并在 NFV 仿真环境下进行验证与应用,进一步提高服务功能虚拟链路的映射性能。

## 参考文献

- [1] HUANG M G, WANG T, LIU L, et al. Virtual network function deployment strategy based on software defined network resource optimization [J]. Computer Science, 2020, 47 (S1): 404-408.
- [2] DING W, QI W, WANG J, et al. OpenSCaaS: an open service chain as a service platform toward the integration of SDN and NFV [J]. Network IEEE, 2015, 29(3): 30-35.
- [3] JIANG J C, ZHU K J, ZHANG Y F, et al. Joint Decision Algorithm for DASH Video Transmission Routing and Bitrate Adjustment in Software-Defined Networking [J]. Journal of Chinese Computer Systems, 2017.
- [4] LIU Y C, LUY W, WANG S, et al. An Optimal Deployment Mechanism of Service Function Chain Based on Software Defined Network [J]. Computer application research, 2019, 36 (10): 3089-3093.
- [5] BREMLER-BARR A, HARCHOL Y, HAY D. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions [C] // Conference on AcmSigcomm Conference. ACM, 2016.
- [6] LI X, QIAN C. The virtual network function placement problem [C] // Computer Communications Workshops. IEEE, 2015: 69-70.
- [7] MCGEER R, ANDERSEN D G, SCHWAB S. The Network Testbed Mapping Problem [C] // Testbeds & Research Infrastructures Development of Networks & Communities-international Ict Conference. Springer Berlin Heidelberg, 2010.
- [8] WEN T, YU H, SUN G, et al. Network function consolidation in service function chaining orchestration [C] // IEEE International Conference on Communications. IEEE, 2016.
- [9] LI X, QIAN C. The virtual network function placement problem [C] // Computer Communications Workshops. IEEE, 2015: 69-70.

**结束语** 多协议标签交换 MPLS 是 IP 网络中提升 QoS 性能的首选解决方案。本文通过 MPLS-TE 来实现骨干网络中流量控制和 QoS 架构的结合,创建基于 MPLS-TE 隧道的拓扑结构,为不同的数据流预留资源,以取代传统的路由机制。基于路由协议创建 MPLS-TE 隧道,这样既可提供安全性,也能够提升网络性能。网络测试数据表明,MPLS-TE QoS 算法能够在网络延迟、抖动、丢包率以及吞吐量等方面有良好的性能表现。

## 参 考 文 献

- [1] MENG J, LI W J, YU G R. Data center dynamic resource allocation for maximum utility[J]. *Application Research of Computers*, 2020, 38(6): 1728-1733.
- [2] JIN Y, LIU Y X, WANG X X. SDN-based data center network multi-path traffic scheduling algorithm[J]. *Computer Science*, 2019(6): 90-94.
- [3] GENG H J, WANG W, WANG H, et al. Overview of Traffic Engineering in the Internet Domain[J]. *Journal of Chinese Computers*, 2021, 42(9): 1891-1899.
- [4] GE Y B. Research on Differentiated Services Model Based on MPLS Traffic Engineering[D]. Lanzhou: Lanzhou University, 2019.
- [5] DU Y M, XIAO J H. Priority-based workflow scheduling with multiple QoS constraints in the cloud environment[J]. *Computer Science*, 2019, 46(10): 128-134.
- [6] ALES E M, CROITORU V. Traffic Engineering and QoS in a Proposed MPLS-VPN[J]. 2020 International Symposium on Electronics and Telecommunications (ISETC), 2020, 5(6): 1-4.
- [7] LU G. Design and Implementation of Multiple Services Based on MPLS VPN in Wide Area Network[J]. *China New Telecommunications*, 2019, 21(6): 51-53.
- [8] ZHANG W. Research and Implementation of the Security Scheme of DMVPN and MPLS VPN[D]. Zibo: Shandong University of Technology, 2018.
- [9] AJDOUB M, EL KAMEL A, YOUSSEF H. An Efficient MPLS-Based Approach for QoS Providing in SDN[J]. *Advances in Intelligent Systems and Computing (AISC)*, 2021, 3(5): 497-508.
- [10] QAYYUM A A, ZULFIQAR M, ABRAR M. Quality of Service Performance Analysis of Voice over IP in Converged MPLS Networks[J]. 2020 International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), 2020, 3(12): 1-4.
- [11] YA C P, GAO C, CHEN Y. Research on link separation path algorithm under SDN architecture[J]. *Computer applications and software*, 2018, 35(9): 183-188.
- [12] XIAO B. Optimal deployment strategy of carrier Ethernet MPLS-TP QoS based on PBB[J]. *Communications technology*, 2021, 54(5): 1143-1150.
- [13] LIU L, YU H F. Service chain mapping algorithm of virtual network function based on resource splitting[J]. *Computer application research*, 2016, 33(8): 2440-2445.
- [14] LI D, LAN J L, WANG P, et al. Service function chain deployment method based on optimal weighted graph matching[J]. *Journal of Communication*, 2019, 40(3): 10-18.
- [15] UMEYAMA S. An eigendecomposition approach to weighted graphmatching problems[J]. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1988, 10(5): 695-703.
- [16] ZHAI D, MENG X R, KANG Q Y. Delay and Reliability Optimization Oriented Service Function Chain Deployment Method [J]. *Journal of Electronics and Information*, 2020, 42(10): 2386-2393.
- [17] GHRIBI C, ZEGHLACHE D. A scalable algorithm for the placement of service function chains[J]. *IEEE Transactions on Network and Service Management*, 2016, 13(3): 533-546.
- [18] MALIS A, BRYANT S, HALPERN J, et al. MPLS Transport Encapsulation for the Service Function Chaining (SFC) Network Service Header (NSH) [OL]. <https://www.rfc-editor.org/info/rfc8596>.
- [19] RAO L L, ZHANG D Z. Solving the 3-SAT problem based on genetic and simulated annealing algorithms[J]. *Modern Computer(Professional)*, 2012(10): 14-16, 36.



**JIANG Jian-feng**, born in 1983, master degree, associate professor. His main research interests include network technology, Internet of things, virtualization, cloud computing technology.



**GAO Ming**, born in 1979, associate professor. His main research interests include new network architecture and industrial Internet.



**ZHOU Hui-ying**, born in 1997, postgraduate. Her main research interests include new network architecture and industrial Internet.

(上接第 480 页)