计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(5):178−186 [doi: 10.15888/j.cnki.csa.009516]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于资源感知的多域服务功能链编排成本优化①

徐九韵[1], 脱颖超[1], 赵耀鹏[1], 李世宝[2]

[1](中国石油大学 (华东) 计算机科学与技术学院, 青岛 266580)
[2](中国石油大学 (华东) 海洋与空间信息学院, 青岛 266580)
通信作者: 脱颖超, E-mail: z21070245@s.upc.edu.cn

摘　要: 网络功能虚拟化技术的兴起使得实例化为服务功能链 (SFC) 的网络服务能够共享基底网络, 缓解了传统网络体系结构僵化的问题. 然而, 网络中大量服务请求给多域 SFC 编排带来了新的挑战. 首先由于域内网络资源信息及内部策略的保密性, 使得多域 SFC 的编排更为复杂. 其次多域 SFC 编排要确定最佳候选编排域集, 先前的研究较少考虑域间负载的均衡性, 对服务接受率造成了消极影响. 此外跨网络域编排服务请求对服务的成本和响应时间提出了更严格的要求. 为解决上述挑战, 在本文中, 我们首先针对多域网络隐私性需求, 提出了域级图的构造方法; 然后基于域间负载均衡提出了域权重的计算方法进行 SFC 编排域的选择; 最后, 针对多域网络成本和响应时间需求, 提出编排算法. 实验结果表明, 提出的算法有效地权衡了平均服务成本和接受率, 并且在服务平均响应时间方面也得到了优化.

关键词: 网络功能虚拟化; 多域网络; 服务功能链编排; 资源感知编排; 成本优化

引用格式: 徐九韵,脱颖超,赵耀鹏,李世宝.基于资源感知的多域服务功能链编排成本优化.计算机系统应用,2024,33(5):178–186. http://www.c-s-a.org.cn/1003-3254/9516.html

Resource-aware Cost Optimization for Multi-domain Service Function Chain Orchestration

XU Jiu-Yun[1], TUO Ying-Chao[1], ZHAO Yao-Peng[1], LI Shi-Bao[2]

[1](College of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China)
[2](College of Oceanography and Space Informatics, China University of Petroleum, Qingdao 266580, China)

Abstract: The emergence of network function virtualization (NFV) technology enables network services instantiated as service function chains (SFCs) to share the underlying network, alleviating the rigidity of traditional network architectures. However, the large number of service requests in the network brings new challenges to multi-domain SFC orchestration. For one thing, the privacy of the intra-domain resource information and internal policies of the network makes multi-domain SFC orchestration more complicated. For another, multi-domain SFC orchestration requires the determination of the optimal set of candidate orchestration domains. Nevertheless, previous studies rarely considered the inter-domain load balance, which negatively affected the service acceptance rate. In addition, the orchestration of service requests across network domains places more stringent requirements on the cost and response time of the service. To address the above challenges, this study proposes a construction method for domain-level graphs to meet the privacy requirement of multi-domain networks. Then, a calculation method for domain weight based on the inter-domain load balance is proposed to select SFC orchestration domains. Finally, the study proposes an orchestration algorithm considering the cost and responses time requirements of multi-domain networks. The experimental results show that the proposed algorithm effectively trades off the average service cost and the acceptance rate and also optimizes the average

① 基金项目: 国家自然科学基金面上项目 (61972417)
收稿时间: 2023-12-13; 修改时间: 2024-01-10; 采用时间: 2024-01-18; csa 在线出版时间: 2024-04-07
CNKI 网络首发时间: 2024-04-10

service response time.

Key words: network function virtualization; multi-domain network; service function chain (SFC) orchestration; resource-aware orchestration; cost optimization

随着物联网设备数量的爆炸式增加, 到达网络的服务请求正变得密集和多样化, 传统的网络设备为硬件专用, 其功能由硬件和软件紧密耦合实现, 难以灵活适应网络请求. NFV 是一种很有应用前景的技术, 为物联网中 5G 设备实现超低时延、高带宽等不同的性能需求奠定了基础[1], 其通过将传统网络中专用的硬件中间件 (例如防火墙、入侵检测系统、网络地址转换器) 转变为通用设备上基于软件的虚拟网络功能 (virtual network function, VNF) 实例, 使底层异构的网络资源成为服务共享的公共基础设施, 提升了服务的灵活性和弹性, 从而可以满足用户指数级增长的服务需求, 同时能够控制资本支出和运营成本.

到达网络的用户服务请求由 NFV 实例化为服务功能链 (service function chain, SFC), 它由一组有序虚拟网络功能构成, 如图 1 中的 SFC 示例所示. 利用 NFV 技术, SFC 可以在底层网络之间进行服务和资源编排, 以实现更灵活和高效的资源分配[2]. SFC 编排过程是指服务提供商在 NFV 网络中按序对组成 SFC 的 VNF 进行资源分配和优化, 进行流量引导和控制, 在满足资源约束的同时优化不同网络服务的编排需求, 实现端到端的服务的过程[3]. 先前的研究对单域 SFC 的编排做出了贡献[4−7], 推动了 NFV 学术研究和工业应用的发展, 然而这些工作大多没有考虑多域场景 (即 SFC 需要部署在不同的域上), 随着边缘计算和物联网的兴起, 服务提供商正在将服务从集中式云转移到边缘网络, 而由于边缘网络缺乏资源、严格的延迟或服务质量需求, 单个边缘网络难以覆盖各类服务功能, 因此多个基础设施提供商 (infrastructure provider, InP) 会达成合作协议, 组成多个 InP 管理的多域网络, 实现资源共享, 从而提升整体服务收益.

图 1 展示了多域 SFC 编排场景, 物联网中如虚拟现实、视频聊天等网络请求被实例化为 SFC 通过接入网络发送到多个 InP 管理的边缘网络在不同 InP 管理的域中进行编排, 其流量依次通过被部署在 3 个域的 VNF, 从起点 (*src*) 到达终点 (*dst*) 实现端到端的服务, 如果需要进一步分析则发送到核心网络进行处理.

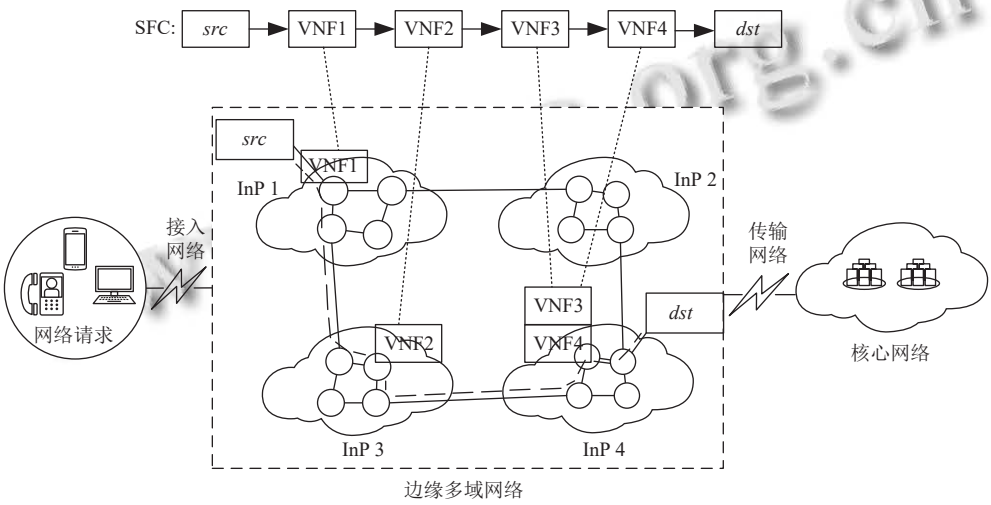


图 1 多域 SFC 编排场景

现有的多域 SFC 编排算法分为集中式和分布式[8]. Guo 等人[9]提出基于强化学习的编排算法优化编排成本. Sarrigiannis 等人[10]建立了基于本地域和外部域的架构, 并提出了一种基于跳数的启发式算法来降低 SFC 编排的成本. Zhang 等人[11]利用 Dijkstra 算法提出启发式服务编排算法, 对服务成本进行了优化. 然而, 与单域 SFC 编排不同, 由于多域网络安全和业务竞争问题, InP 不愿意共享其拓扑和内部策略信息, 因此这

些方法不适合信息公开有限的场景. 为了维护 InP 的隔离性和隐私性, Sun 等人[12]基于边界网关协议[13]对多域网络进行了抽象, 然后对时延和负载均衡进行了优化. Joshi 等人[14]提出 pSMART 算法通过匹配域中的 VNF 类型来确定候选路径, 然后根据单个域中部署的 VNF 类型的历史成功率来选择编排域. Toumi 等人[15]提出了一种集中式跨域 SFC 编排框架, 设置多域编排器和单域编排器, 从应用层面实现了多域 SFC 编排过程. 上述研究中, 没有有效的解决域间负载均衡的问题, 使得网络域的服务接受率表现欠佳. Liu 等人[8]和 Dalgkitsis 等人[16]提出了分布式编排方法, 但未给出域内的编排算法. Chen 等人[17]提出了一种分布式框架, 其中入口编排器确定最短成本路径, 然后沿路径的候选域的编排器确定 SFC 编排方案. 上述分布式方法避免了信息泄露问题, 但其划分方案需要域间频繁通信才能达成共识, 增加了通信成本和响应延迟.

综合上述研究, 多域 SFC 的编排面临以下问题, 首先各网络域不愿意披露域内敏感信息来进行 SFC 编排决策, 导致多域 SFC 编排时缺少关键信息, 使编排方案效果不佳; 其次先前的研究在进行 SFC 编排时只考虑了服务的性能, 忽略了域间负载的平衡, 使得各个域负载不均导致服务接受率的降低. 本文针对上述挑战, 首先, 考虑到多域网络由不同的 InP 管理, 各个 InP 出于安全、业务竞争等原因, 因此不愿意公开域内的网络拓扑和详细资源信息, 本文基于物理网络设置域级图来隐藏敏感信息, 进一步协调多域资源以实现经济有效且资源高效的网络服务. 其次, 针对域间负载平衡问题, 设计了域负载权重的计算方法, 该计算方法弥补先前研究中使用整个域资源计算权重导致 VNF 部署失败的缺点, 将该权重的计算细化到域内节点的资源负载, 提高服务接受率. 此外, 物联网中如视频通话等服务, 对服务提供商而言, 针对只有降低成本和提高网络服务性能才能带来有利可图的商业模式[18], 需要设计算法降低编排成本及服务响应时间.

1 系统模型及问题定义

1.1 物理网络建模

物理网络被建模为无向图$G=(N,L)$, 其中 N 和 L 分别是物理服务器节点和节点之间的物理链路的集合. 在多域场景中, 网络由域间链路连接的 M 个管理域组成. 第 i 个域可以表示为$G_i=(N_i,L_i)$. L_{inter}表示域间物理链路的集合. 物理节点$n_i \in N_i$的总计算资源容量和剩余资源分别记为n_i^{cpu}和n_i^{resd}. 此外, 用ϕ_{n_i}来表示n_i的邻居节点, $l_i^{n,m} \in L_i\ (n \in N_i, m \in \phi_{n_i})$表示节点 n 和 m 之间的连接链路. 同一域内节点 n 和 m 之间的链路带宽容量和通信延迟分别表示为$b^{l_i^{n,m}}$和$d^{l_i^{n,m}}$, 同时域间链路的带宽和时延分别表示为$b^{l_{\text{inter}}^{n,m}}$和$d^{l_{\text{inter}}^{n,m}}$,其中$l_{\text{inter}}^{n,m} \in L_{\text{inter}}$.

1.2 SFC 建模

SFC 被建模为有向图, 每个 SFC 请求$r \in R$表示为$r=(N^r,L^r,\eta^r,src,dst,dly^r)$, 其中$N^r$和$E^r$分别表示 SFC 请求的虚拟功能节点和虚拟链路集合. 对于请求节点$n_j^r \in N^r(1 \leqslant j \leqslant |N^r|)$, $dem^{n_j^r}$表示节点n_j^r的计算资源需求. 然后, $l_k^r \in L^r\ (1 \leqslant k \leqslant |N^r|-1)$是连接两个相邻虚拟节点$n_k^r$和$n_{k+1}^r$的虚拟链路. η^r为 SFC 请求的带宽需求, src和dst为每个 SFC 请求流量的入口和出口, dly^r为完成请求的最大容忍延迟.

1.3 域级图建模

为保证域内资源信息的隐私性, 将图 2(a) 的物理拓扑进行抽象, 构建了域级图如图 2(b), 保留了域间链路及其端点. 域级图被建模为$G_D=(W,I)$, 其中 W 为每个域的权重参数集合, I 是域间链路资源信息集合, 该集合包含了域间链路时延、带宽容量和单价. 域 i 的权重$W_i \in W$为域中节点的平均可用资源, 旨在加强域内可用资源丰富且域内负载节点较少的域的服务部署, 实现域间负载均衡, 并提高请求接受率. W_i由式 (1) 表示.

$$W_i=\frac{\dfrac{1}{|N_i|}\cdot\sum\limits_{n_i\in N_i}\dfrac{n_i^{\text{resd}}}{n_i^{\text{cpu}}}}{\sum\limits_{i=1}^{M}\left(\dfrac{1}{|N_i|}\cdot\sum\limits_{n_i\in N_i}\dfrac{n_i^{\text{resd}}}{n_i^{\text{cpu}}}\right)} \tag{1}$$

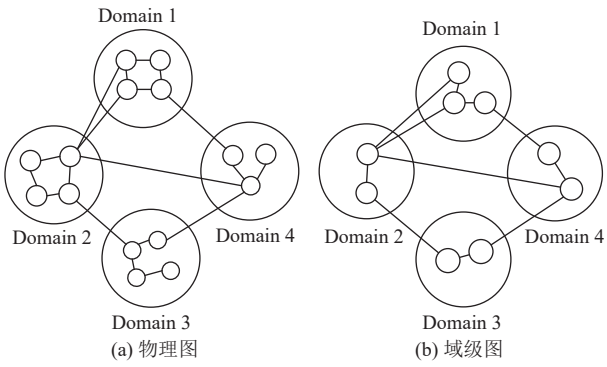


图 2 域级图构建

1.4 编排成本优化

多域 SFC 编排需要跨域进行服务部署对成本和响

应时间提出了更严格的要求, 因此本文提出资源感知的编排算法旨在实现编排成本最小化. 针对每个 InP 基于商业因素并没有公开域内资源信息的现实情况, 因此编排成本分为域间成本C_{inter}和域内成本C_{intra}两部分, 目标函数如下:

$$\min C = \min(C_{\text{inter}} + C_{\text{intra}}) \tag{2}$$

● 域间成本最小化: 将域级图与 SFC 的 *src* 和 *dst* 结合起来, 最小化域间链路成本, 从而确定多域编排的候选路径. 域间链路成本的目标函数C_{inter}如下:

$$C_{\text{inter}} = \sum_{e_k^r \in E^r} \sum_{e_{\text{inter}}^{n,m} \in E_{\text{inter}}} \beta_{\text{inter}}^{n,m} \cdot \eta^r \cdot y^{l_k^r, l_{\text{inter}}^{n,m}} \tag{3}$$

其中, $\beta_{\text{inter}}^{n,m}$为域间链路带宽单价, $y^{l_k^r, l_{\text{inter}}^{n,m}}$为决策变量, 该值为 1 时表示虚拟链路$l_k^r$被部署在物理链路$l_{\text{inter}}^{n,m}$上, 否则为未部署.

● 域内成本最小化: 单位时间 SFC 的域内编排成本由节点单位计算成本C_{comp}和单位带宽成本C_{band}组成, 域内编排成本C_{intra}可表示为:

$$\begin{aligned} C_{\text{intra}} &= C_{\text{comp}} + C_{\text{band}} \\ &= \sum_{n_i \in N_i} \sum_{n_j^r \in N_r} \alpha_d \cdot dem_{n_j^r} \cdot x^{n_j^r, n_i} + \sum_{n \in N_i, m \in \phi_n} \sum_{e_k^r \in E_r} \beta_d \cdot \eta^r \cdot y^{l_k^r, l_i^{n,m}} \end{aligned} \tag{4}$$

其中, α_d和β_d为域 d 计算资源和带宽的单价, $x^{n_j^r, n_i}$和$y^{l_k^r, l_i^{n,m}}$为决策变量, 当其值为 1 时, 表示虚拟节点、链路成功部署在物理网络上, 否则部署失败.

上述目标函数必须遵循以下约束.

式 (5) 约束请求中每个 VNF 只能被分配一次:

$$\sum_{i=1}^{M} \sum_{n_i \in N_i} x^{n_j^r, n_i} \leqslant 1, \ \forall r \in R, \forall n_j^r \in N^r \tag{5}$$

式 (6) 为物理节点计算资源容量限制.

$$\sum_{n_j^r \in N^r} x^{n_j^r, n_i} \cdot dem^{n_j^r} \leqslant n_i^{\text{resd}}, \ \forall i \in [1, M], \forall n_i \in N_i \tag{6}$$

式 (7) 和式 (8) 分别为域内和域间物理带链路带宽限制.

$$\sum_{k=1}^{|N^r|-1} y^{l_k^r, l_i^{n,m}} \cdot \eta^r \leqslant b^{l_i^{n,m}}, \ \forall i \in [1, M], \forall l_i^{m,n} \in L_i \tag{7}$$

$$\sum_{k=1}^{|N^r|-1} y^{l_k^r, l_{\text{inter}}^{n,m}} \cdot \eta^r \leqslant b^{l_{\text{inter}}^{n,m}}, \ \forall l_{\text{inter}}^{m,n} \in L_{\text{inter}} \tag{8}$$

式 (9) 引入延迟约束, 式 (10) 表示 SFC 请求的端到端延迟 T 必须满足最大容忍延迟的约束.

$$\begin{aligned} T &= T_{\text{proc}} + T_{\text{prop}} + T_{\text{inter}} \\ &= \sum_{n_i \in N_i} \sum_{n_j^r \in N_r} x^{n_j^r, n_i} \cdot t^{n_j^r} + \sum_{l_i^{n,m} \in L_i} \sum_{l_k^r \in L_r} y^{l_k^r, l_i^{n,m}} \cdot d^{l_i^{n,m}} \\ &\quad + \sum_{l_{\text{inter}}^{n,m} \in L_{\text{inter}}} \sum_{l_k^r \in L_r} y^{l_k^r, l_{\text{inter}}^{n,m}} \cdot d^{l_{\text{inter}}^{n,m}} \end{aligned} \tag{9}$$

其中, $t^{n_j^r}$为 VNF 的处理时延.

$$T \leqslant dly^r \tag{10}$$

2 多域服务功能链编排算法

2.1 集中式多域 SFC 编排算法实现

多域 SFC 编排方法首先根据优化目标确定一组最优候选域, 然后在每个域中进行 SFC 编排, 最后将 SFC 部署到物理网络上. 现有网络中的服务对成本、接受率及响应时间提出了更高的需求. 多域 SFC 编排被认定为 NP 难问题[17], 因此, 本文提出一种集中式多域 SFC 编排方法 (MDSCO), 以在可行的时间内获得接近最优的编排, 主要包括 3 个部分: 候选域的获取、SFC 分块以及域内成本优化编排. 首先在多域编排器处确定候选域, 然后基于候选域的权重进行 SFC 分块, 分块获得的子 SFC 被多域编排器发送至各候选域内在域编排器管理下进行编排, 各域编排结束后, 将域内确定的编排方案返回多域编排器, 若失败则返回编排失败结果.

MDSCO 的算法流程如图 3 所示, SFC 请求到达多域编排器后, 各网络的 InP 基于商业因素不公开域内详细资源信息, 因此本文提出了域级图的构建方法隐藏 InP 的网络拓扑和资源信息; 然后算法将域级图和 SFC 请求的入口和出口节点相结合, 使用最小化域间成本 k 最短路算法[19]确定候选编排路径集合 P, 其中候选路径经过的域集为候选编排域集 D; 然后算法基于式 (1) 确定的各域权重进行 SFC 分块; SFC 的分块方案由多域编排器发送至各域的域内编排器, 然后各候选域分别对分配到的子 SFC 进行编排, 如果某个候选域的子 SFC 编排失败, 算法将尝试下一条成本最低的候选路径; 反之, 所有候选域都编排成功则向多域编排器返回编排成功的方案.

算法 1. 多域 SFC 编排算法 (MDSCO)

输入: SFC 请求集合 R, 物理网络 G.

输出: SFC 在多域物理网络的编排方案.

1. 将物理网络抽象为域级图;
2. 将 SFC 的 *src* 和 *dst* 与域级图结合确定 SFC 编排的候选路径集合 *P*, 并根据式 (3) 计算 *P* 的通信成本对候选路径排序;
3. **while** $P^g \in P \neq \varnothing$ **do**
4. **if** P^g 中链路的带宽需求大于物理带宽容量, 则重新选择下一条候选路径进行编排;
5. **end if**
6. 根据候选路径 P^g 经过的路径确定候选域集合 *D*;
7. **for** $d \in D$ **do**
8. 调用算法 2 对子 SFC 进行域内部署;
9. **if** 算法 2 部署失败, 则选择下个候选域部署;
10. **end if**
11. **end for**
12. **if** 请求 *r* 部署成功并满足时延等约束
13. **return** 多域 SFC 编排方案;
14. **end if**
15. **end while**
16. **return** 请求 *r* 编排失败.

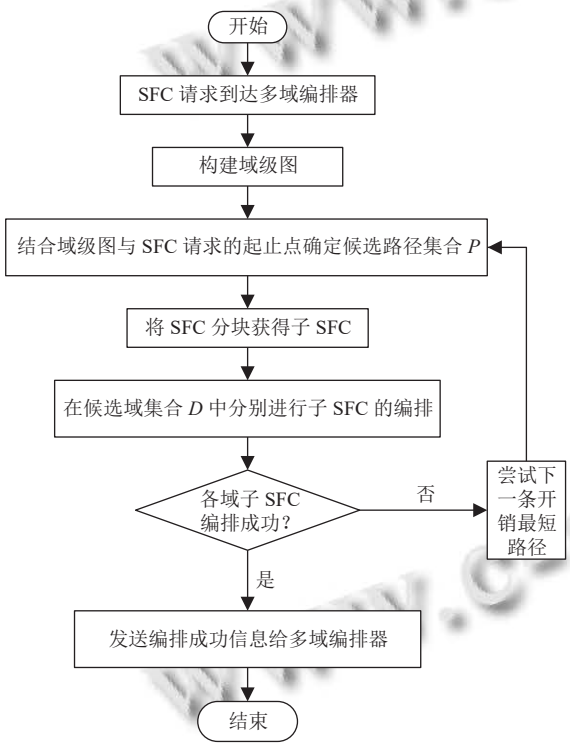


图 3 MDSCO 算法流程图

多域 SFC 算法的伪代码见算法 1, 算法包含的 3 部分的具体实现如下.

(1) 候选域的获取: 为了降低多域复杂网络拓扑的编排成本和计算复杂度, 构建候选路径集来过滤管理域从而获取编排候选域, 候选路径的构造方法如下.

将 SFC 流的入口和出口与算法 1 第 1 行抽象获得的域级图相结合, 基于域级图公开的单位域间链路传输成本构建候选路径集 *P*. 第 *g* 条最小候选路径 $P^g \in P$ 的单位域间传输成本计算如下:

$$C^{P^g} = \sum\nolimits_{p \in P^g} \beta^p_{\text{inter}} \tag{11}$$

其中, *p* 为物理网络中任意一条域间链路, 将最短路径 P^g 经过的域视为候选域集合 *D*.

(2) SFC 分块: 为充分利用各域资源, 避免因各域资源利用不均导致网络拥塞, 违反服务的最大时延容忍约束造成编排失败. 本算法考虑基于各域可用资源的权重进行 SFC 分块, 实现域间负载均衡, 提高域内子 SFC 编排的成功率, 从而提高服务接受率, 算法使用域级图中披露的权重集合 *W* 对 SFC 进行分块, 第 *i* 个域中子 SFC 的长度如式 (12).

$$len_i = \left\lfloor \frac{W_i}{\sum_{i=1}^{D} W_i} \cdot len \right\rfloor \tag{12}$$

$$len_{\max} = len_{\max} + \left(len - \sum_{i=1}^{D} len_i \right) \tag{13}$$

其中, *len* 为 SFC 的长度, $len_{\max}$ 为权重最大的域中的 SFC 的长度.

(3) 域内子 SFC 编排: 通过候选域获取、SFC 分块操作多域编排器将 SFC 划分为多个子 SFC, 并将其发送给各域的域内编排器, 各域内的子 SFC 基于成本最小化的编排的伪代码见算法 2. 首先基于算法 1 选择的路径 P^g 确定每个子 SFC 流量的入口和出口, 然后根据式 (4) 域内成本最小化函数使用 *k* 最短路算法确定域内候选编排路径集合 P^d; 算法 2 的第 3–9 行首先根据候选路径的带宽容量排除不满足带宽容量的候选路径, 然后进行 VNF 的部署, 淘汰不满足计算资源需求的物理节点, 选择候选路径上的下一个物理节点; 最后将部署结果返回到算法 1 的多域编排器. 各域编排器将各子 SFC 的编排结果发送给多域编排器进行确认, 确定最终多域 SFC 的编排方案.

算法 2. 域内子 SFC 编排

输入: 子 SFC 请求 r_d, 候选路径 P^g.
输出: r_d 在域 *d* 上的编排方案.

1. 结合 P^g 和域级图信息集合确定域 *d* 的 *src* 和 *dst*;
2. 使用式 (4) 求 *src* 和 *dst* 之间成本最小的路径集合 P^d;
3. **while** $p \in P^d \neq \varnothing$ **do**
4. **if** *p* 中链路的带宽需求大于物理带宽容量, 选择 P^d 中下一条路径;

5. **end if**
6. **for** $n_j^r \in r_d$ **do**
7. **if** 物理节点n_i计算资源不能满足n_j^r的需求, 则进入 p 的下一个物理节点;
8. **end if**
9. **end for**
10. **end while**
11. **if** r_d部署成功 **return** r_d部署方案;
12. **return** r_d部署失败.

2.2 MDSCO 重新分块算法实现

在域内编排过程中, 由于多域网络未披露域内详细资源信息, 初始分块方案中子 SFC 可能无法全部编排成功, 为提升算法服务接受率, 本文在 MDSCO 的基础上提出了 SFC 的重新分块算法 MDSCO-RP. 该算法的思想是当某域内子 SFC k 条候选路径均编排失败后, 算法不是从整个 SFC 的起点重新分块, 而是保存 SFC 编排成功的最长长度 (即部署成功的 VNF 个数最多) 的候选路径, 并将该候选路径返回, 然后从下一个候选域开始, 对剩余 SFC 基于剩余候选域的权重重新分块, 调用算法 2 对重新分块后的子 SFC 进行编排.

2.3 时间复杂度分析

MDSCO 和 MDSCO-RP 时间复杂度主要分为两部分: 域间候选域的获取时的时间复杂度$O(k|D|(|L_{\text{inter}}|+|D|\log|D|))$ 与算法 2 在域内编排时的时间复杂度$O(k\sum_{i=1}^{|D|}(k|N_i|(|L_i|+|N_i|\log|N_i|)))$之和, 其中 k 为 k 最短路候选路径的值, $|D|$和$|L_{\text{inter}}|$为候选域的数量和域间链路的数量, $|N_i|$和$|L_i|$分别为域 i 中服务节点和链路的数量.

3 实验分析

3.1 实验设置

在本节中, 设置了实验来验证本文提出的算法性能, 仿真实验采用 Python 语言实现, 实验在 CPU Intel(R) Core(TM) i5-7300H 的计算机进行. 网络拓扑使用来自 Internet Topology Zoo[20]中的 4 个拓扑, 其节点和链路设置如表 1 所示.

(1) 网络拓扑设置: 实验网络设置 4 个域, 分别使用表 1 中的 4 种拓扑, 任意两个域都至少有一条路径连接, 为保证多域信息的隐私性, 域间链路的信息只包含相邻域的链路信息, 根据先前研究[8,12]设置模拟实验参数, 资源和成本的指标均为虚拟单位. 域中的节点的计算资源和链路的带宽容量分别服从均匀分布 400–600 个单位和 800–1 200 个单位, 域内节点的单位计算资源价格和带宽价格均服从 U(0.1, 0.8). 对于域间链路, 带宽容量服从均匀分布 2 500–4 000 个单位, 其单位带宽价格服从均匀分布 U(1, 4). 此外, 域内和域间链路时延分别服从均匀分布 U(1, 6) 和 U(10, 20).

表 1 实验网络拓扑设置

类别	BtAsiaPac	Canerie	Xspedius	Geant2012
节点数	20	32	34	40
链路数	31	41	49	61

(2) SFC 请求设置: SFC 由 5–10 个 VNF 组成, 其 *src* 和 *dst* 均匀分布在物理网络中. SFC 中 VNF 的计算资源和带宽需求分别服从均匀分布 U(5, 10) 和 U(10, 20), VNF 处理延迟$t^{n_j^r}$和服务最大容忍时延分别服从均匀分布 U(0, 2) 和 U(10, 100). SFC 请求的到达服从平均值为 5 的泊松分布, SFC 的生命周期服从平均单位为 1 200 的指数分布.

(3) 算法对比: 比较了多域 SFC 编排的 3 种算法以及本文提出的重新分块算法, 实验设置环境均与本文一致.

1) DFSC[17]. 该算法为分布式编排算法, 利用 SFC 源与目的地之间的路径确定候选域, 并从资源单价最低的域开始编排, 最小化目标成本.

2) Heuristic[11]. 该算法为集中启发式算法, 其未对多域信息进行隐私性操作, 首先基于 Dijkstra 算法最小化 SFC 流的源节点和目的节点的带宽成本, 然后减少参与编排的候选域, 将 VNF 放置在资源利用率少的服务器节点, 提升资源利用率.

3) WGT_LB[12]. 该算法为集中式算法, 根据域可用资源进行 SFC 分区, 然后使用贪心算法在域内进行最小化成本编排.

4) MDSCO-RP. 本文第 2.2 节提出了 MDSCO 的重新分块算法, 该算法为提升 SFC 编排成功率, 对 MDSCO 进行改进.

(4) 指标对比: 算法对比以下指标, 其中实验结果为同一环境设置下重复 10 次实验的平均数据.

1) 接受率 AR, 指成功部署编排解决方案的 SFC 数量与到达的 SFC 总数的比率, 定义如下:

$$AR = \frac{sum(successSFC)}{sum(arrivedSFC)} \tag{14}$$

其中, $sum(successSFC)$和$sum(arrivedSFC)$分别为成功

编排和到达网络的 SFC 的数量.

2) 平均服务成本C^{avg}, 指根据编排方案成功部署 SFC 的平均成本, 定义如下:

$$C^{\text{avg}}=\frac{\sum_{r\in successSFC} C^r}{sum(successSFC)} \tag{15}$$

其中, C^r为使用式 (2) 计算的第 r 个编排成功的请求的服务成本.

3) 平均响应时间$T^{\text{avg}}_{\text{response}}$, 指到达 SFC 的决策时间之和与到达 SFC 总数的比值, 定义如下:

$$T^{\text{avg}}_{\text{response}}=\frac{\sum_{r\in ArrivedSFC} T^r_{\text{end}}-T^r_{\text{start}}}{sum(arrivedSFC)} \tag{16}$$

其中, T^r_{end}和T^r_{start}分别为第 r 个请求的开始和结束时间.

3.2 实验结果与分析

(1) 不同 SFC 数量下编排算法对比

图 4 展示了 200–2 000 个 SFC 分别输入网络后各算法的表现, 从图 4(a) 可以看出随着网络中请求数量的增加, 各算法的接受率呈现下降趋势, 其中 MDSCO 和 MDSCO-RP 的接受率分别比 WGT_LB 约低 4% 和 2%, 但如图 4(b) 所示, 当输入 2000 个 SFC 时, WGT_LB 的平均服务成本比 MDSCO 和 MDSCO-RP 分别高约 13%, 这是由于 WGT_LB 使用贪心算法对网络进行遍历实现部署, 因此服务接受率高于 MDSCO 和 MDSCO-RP, 但贪心算法为提升服务接受率造成的路径折返使其成本优化性能削弱, 而 MDSCO 和 MDSCO-RP 权衡了平均服务成本和接受率, 在接受率较高的前提下, 降低了编排成本. 由于 DFSC 和 Heuristic 优先考虑编排成本最小化, 均未考虑域间负载平衡, 其中 DFSC 优先在资源单价较低的域进行编排, 随着请求数量的增加造成该域的资源不足; Heuristic 减少参与编排的域, 算法考虑了节点的资源利用率阈值但未考虑节点间链路带宽容量, 同样会导致域内资源不足, DFSC 和 Heuristic 多域资源利用率较低, 其接受率下降趋势最明显. 此外, Heuristic 的服务接受率高于 DFSC, 这是由于 DFSC 总是从单价最低的域开始编排, 其多域资源利用率更差. 图 4(b) 中 DFSC 的平均服务成本随着请求数量的增多逐渐高于 MDSCO, 后比 MDSCO 低, 这是由于 DFSC 在请求数量较少时, 其成本优化性能较好, 但随着请求的增加 SFC 部署到资源单价更高的域中, 其成本优化性能下降; 当 SFC 的数量高于 1 000 时, 接受率的降低使其服务成本低于 MDSCO. 同样的, Heuristic 的平均服务成本随着请求数量的增多逐渐高于 MDSCO, 后比 MDSCO 低, 这是由于基于 Dijkstra 的带宽成本最小化随着域内资源的消耗, 候选路径的成本优化性能下降, 当 SFC 的数量高于 1 600 时, 接受率的降低使其服务成本低于 MDSCO. 从图 4(a) 和图 4(b) 中显然可以得出, MDSCO-RP 有效提升了服务接受率, 其平均服务成本也逐渐优于 MDSCO, 这是由于 MDSCO-RP 在 SFC 编排失败后, 不需要从 SFC 的起点重新进行编排, 降低了编排成本.

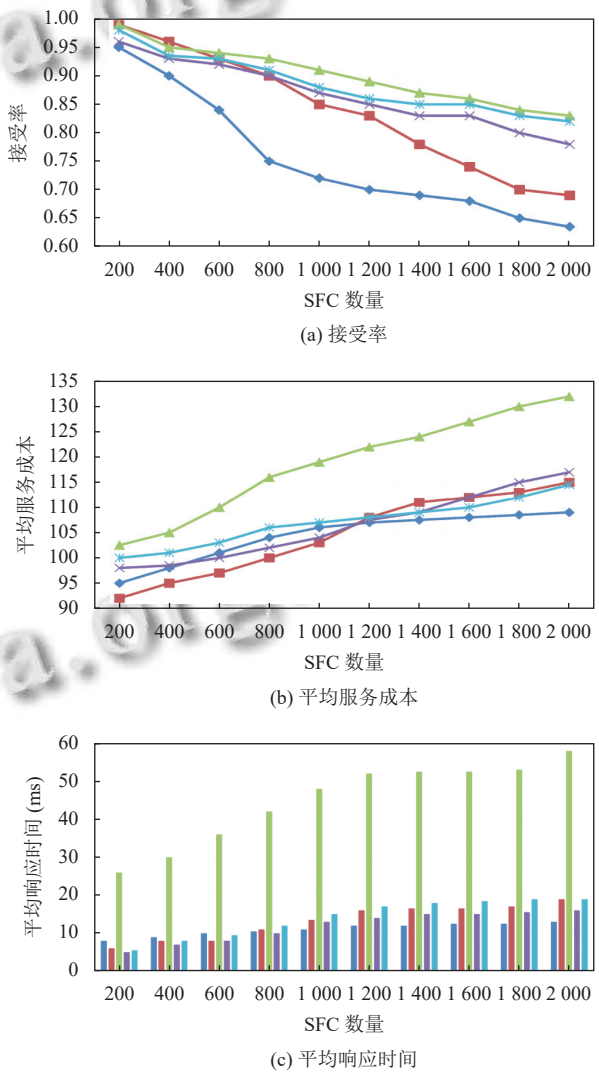


图 4 不同 SFC 数量下编排算法对比

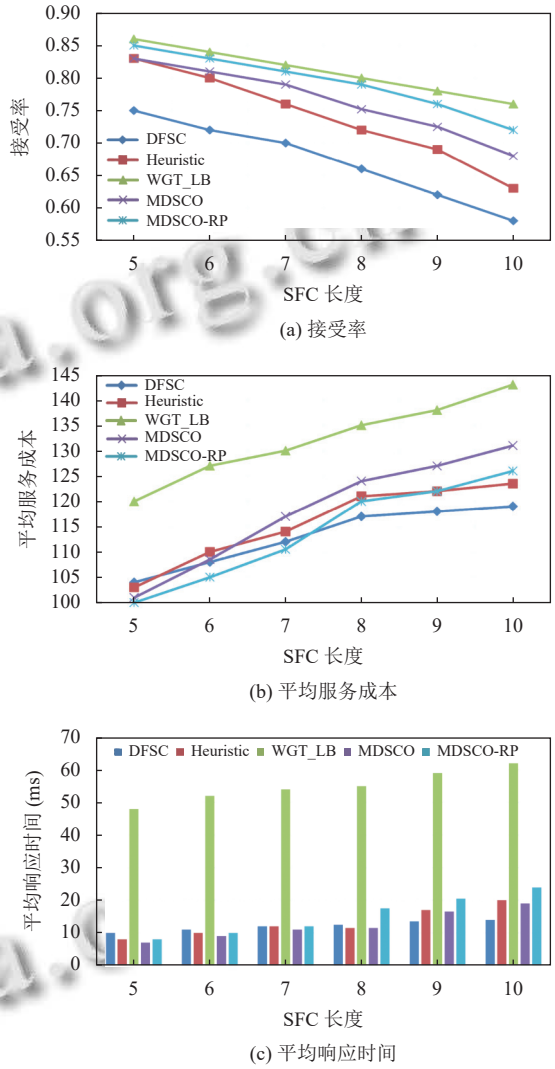
如图 4(c) 随着 SFC 数量的增加, 3 种算法平均响应时间呈上升趋势, WGT_LB 的平均响应时间比 MDSCO 和 DFSC 高约 40 ms, 这是同样是贪心算法较

k 最短路算法需要遍历更多的节点, 导致其响应时间大大增加; DFSC 的接受率逐渐低于 MDSCO, 表明其服务拒绝率较高, 其平均响应时间逐渐低于 MDSCO. Heuristic 的平均响应时间与 MDSCO 相近. MDSCO-RP 的平均响应时间略高于 MDSCO, 这是由于 MDSCO 编排失败直接拒绝, 而 MDSCO-RP 进行了重新分块.

(2) 不同 SFC 长度下编排算法对比

图 5 展示了将 2 000 个 SFC 输入网络, 随着 SFC 长度的增加, 服务接受率呈下降趋势, 其中如图 5(a) 所示 MDSCO 和 MDSCO-RP 的接受率分别比 WGT_LB 低约 5% 和 3%, DFSC 接受率远低于 MDSCO 和 WGT_LB, 同前文分析一致, DFSC 忽略域间负载平衡使其接受率表现较差, WGT_LB 通过遍历网络节点使其接受率高于 MDSCO. 随着 SFC 长度的增加, Heuristic 的接受率与 MDSCO 的差距逐渐增大, 同前文分析一致, Heuristic 将 SFC 编排到尽可能少的域中, 该算法只考虑了节点的阈值, 使得域内链路带宽容量不足, 导致以此域为流量起点或终点的 SFC 编排失败. SFC 长度下平均服务成本如图 5(b) 所示, WGT_LB 遍历网络节点进行 SFC 部署时, 虽然接受率表现较好, 但如前文所分析的使用贪心算法会导致路径折返, 提升了服务成本, 因此 WGT_LB 的平均服务成本比 MDSCO 高约 10%; DFSC 和 Heuristic 的平均服务成本逐渐高于 MDSCO, 后逐渐低于 MDSCO, 这是由于随着 SFC 长度的增加, 网络负载逐渐增大, 而 DFSC 和 Heuristic 域间负载失衡更为明显, 导致其接受率大大降低, 服务成本逐渐低于 MDSCO. 基于前文阐述 MDSCO-RP 在 SFC 编排失败后, 不需要从 SFC 的起点重新进行编排, 降低了编排成本, 因此 MDSCO-RP 在 2 000 个 SFC 输入网络 (即资源短缺) 的平均服务成本优于 MDSCO.

图 5(c) 展示平均响应时间随 SFC 长度的增加呈上升趋势, 其中 WGT_LB 的响应时间比 MDSCO 和 DFSC 高约 45 ms, 原因为当请求数量较大时, 贪心算法比 k 最短路时间复杂度更高; 随着 SFC 长度的增长, DFSC 接受率远低于 MDSCO, 因此其响应时间逐渐低于 MDSCO. 与前文一致, Heuristic 的响应时间与 MDSCO 相近. MDSCO-RP 的响应时间略高于 MDSCO, 这是由于随着域内资源的减少 MDSCO 的 编排失败率高于 MDSCO-RP. 因此我们提出的算法 MDSCO 相较于 DFSC 和 Heuristic 能够权衡服务成本和接受率, 在保证服务接受率性能可接受的前提下, 优化了服务成本, 重新分块算法 MDSCO-RP 的服务成本和服务接受率均得到了提升. 此外 MDSCO 和 MDSCO-RP 的平均响应时间随着网络负载的增大性能比较稳定.



(a) 接受率

(b) 平均服务成本

(c) 平均响应时间

图 5 不同 SFC 长度下编排算法对比

4 结论与展望

本文提出了基于资源感知的多域 SFC 编排成本优化算法 MDSCO 和 MDSCO-RP, 对域级图进行建模保护域内资源信息的机密性; 结合域级图披露的网络资源负载设计了一种域权重的计算方法实现域间负载的平衡, 提升了服务接受率; 并基于目标函数设计算法实现了服务成本和响应时间的优化. 接下来的研究工作是当一个或多个 VNF 发生故障或失效时, 考虑 SFC 的可用性和运营成本等因素设计 SFC 编排算法.

参考文献

1 Kaur K, Mangat V, Kumar K. A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture. Computer Science Review, 2020, 38: 100298. [doi: 10.1016/j.cosrev.2020.100298]

2 Yu H, Taleb T, Zhang JW. Deterministic latency/jitter-aware service function chaining over beyond 5G edge fabric. IEEE Transactions on Network and Service Management, 2022, 19(3): 2148–2162. [doi: 10.1109/TNSM.2022.3151431]

3 Wang L, Dolati M, Ghaderi M. CHANGE: Delay-aware service function chain orchestration at the edge. Proceedings of the 5th IEEE International Conference on Fog and Edge Computing (ICFEC). Melbourne: IEEE, 2021. 19–28.

4 贾雨宁, 魏翼飞, 周军华. 基于 SDN 与 NFV 的服务功能链编排算法. 北京邮电大学学报, 2022, 45(2): 85–90. [doi: 10.13190/j.jbupt.2021-218]

5 Zu JC, Hu GY, Yan JJ, *et al*. A community detection based approach for service function chain online placement in data center network. Computer Communications, 2021, 169: 168–178. [doi: 10.1016/j.comcom.2021.01.014]

6 Liu YC, Lu H, Li X, *et al*. Dynamic service function chain orchestration for NFV/MEC-enabled IoT networks: A deep reinforcement learning approach. IEEE Internet of Things Journal, 2021, 8(9): 7450–7465. [doi: 10.1109/JIOT.2020.3038793]

7 张岳, 张俊楠, 吴晓春, 等. 基于改进灰狼优化算法的服务功能链映射算法. 电信科学, 2022, 38(11): 57–72. [doi: 10.11959/j.issn.1000-0801.2022275]

8 Liu Y, Zhang HQ, Chang DX, *et al*. GDM: A general distributed method for cross-domain service function chain embedding. IEEE Transactions on Network and Service Management, 2020, 17(3): 1446–1459. [doi: 10.1109/TNSM.2020.2993364]

9 Guo SY, Qi YY, Jin Y, *et al*. Endogenous trusted DRL-based service function chain orchestration for IoT. IEEE Transactions on Computers, 2022, 71(2): 397–406. [doi: 10.1109/TC.2021.3051681]

10 Sarrigiannis I, Antonopoulos A, Ramantas K, *et al*. Cost-aware placement and enhanced lifecycle management of service function chains in a multidomain 5G architecture. IEEE Transactions on Network and Service Management, 2022, 19(4): 5006–5020. [doi: 10.1109/TNSM.2022.3187314]

11 Zhang CC, Wang XW, Dong AW, *et al*. Dynamic network service deployment across multiple SDN domains. Transactions on Emerging Telecommunications Technologies, 2020, 31(2): e3709. [doi: 10.1002/ett.3709]

12 Sun G, Li YY, Liao D, *et al*. Service function chain orchestration across multiple domains: A full mesh aggregation approach. IEEE Transactions on Network and Service Management, 2018, 15(3): 1175–1191. [doi: 10.1109/TNSM.2018.2861717]

13 Abujoda A, Papadimitriou P. DistNSE: Distributed network service embedding across multiple providers. Proceedings of the 8th International Conference on Communication Systems and Networks (COMSNETS). Bangalore: IEEE, 2016. 1–8.

14 Joshi KD, Kataoka K. pSMART: A lightweight, privacy-aware service function chain orchestration in multi-domain NFV/SDN. Computer Networks, 2020, 178: 107295. [doi: 10.1016/j.comnet.2020.107295]

15 Toumi N, Bernier O, Meddour DE, *et al*. On cross-domain service function chain orchestration: An architectural framework. Computer Networks, 2021, 187: 107806. [doi: 10.1016/j.comnet.2021.107806]

16 Dalgkitsis A, Garrido LA, Rezazadeh F, *et al*. SCHE2MA: Scalable, energy-aware, multidomain orchestration for beyond-5G URLLC services. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(7): 7653–7663. [doi: 10.1109/TITS.2022.3202312]

17 Chen C, Nagel L, Cui L, *et al*. Distributed federated service chaining: A scalable and cost-aware approach for multi-domain networks. Computer Networks, 2022, 212: 109044. [doi: 10.1016/j.comnet.2022.109044]

18 邱航, 汤红波, 游伟. 基于深度 Q 网络的在线服务功能链部署方法. 电子与信息学报, 2021, 43(11): 3122–3130. [doi: 10.11999/JEIT201009]

19 Yen JY. Finding the *K* shortest loopless paths in a network. Management Science, 1971, 17(11): 712–716. [doi: 10.1287/mnsc.17.11.712]

20 Knight S, Nguyen HX, Falkner N, *et al*. The Internet topology zoo. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765–1775. [doi: 10.1109/JSAC.2011.111002]

(校对责编: 张重毅)