端到端时延上限确定的服务链部署算法

王泽南[1,2]，张娇[1,2]，汪硕[1,2]，黄韬[1,2]，F.Richard Yu[3]
（1. 网络通信与安全紫金山实验室，江苏 南京 211111；2. 北京邮电大学网络与交换技术国家重点实验室，北京 100876；
3. 加拿大卡尔顿大学，渥太华 K1S 5B6）

摘 要：针对现有服务链部署算法无法保证通过服务链的每一个数据包的端到端时延问题，提出端到端时延上限确定的服务链部署算法。首先，基于网络演算得到服务链的端到端时延上限；然后，通过协同优化服务链路由与虚拟网络功能节点资源分配，实现确定的服务链端到端时延上限。实验结果表明，所提算法能在提高服务链接收数量的同时，保证通过服务链的每一个数据包的端到端时延均满足业务需求。

关键词：端到端时延；服务链；网络功能虚拟化；网络演算
中图分类号：TN91
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021189

Service chain deployment algorithms for deterministic end-to-end delay upper bound

WANG Ze’nan[1,2], ZHANG Jiao[1,2], WANG Shuo[1,2] , HUANG Tao[1,2], F.Richard Yu[3]
1. Purple Mountain Laboratories, Nanjing 211111, China
2. State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
3. Carleton University, Ottawa K1S 5B6, Canada

Abstract: To solve the problem that the current service chain deployment algorithms cannot guarantee the delay of each packet passing through the service chain (SC), a SC deployment algorithm for deterministic end-to-end delay upper bound was proposed. First, the end-to-end delay bound of the SC was derived based on network calculus. Then, the deterministic end-to-end delay bound of the SC was achieved by collaboratively optimizing the routing of SC and the resource allocation of the virtual network function nodes in the SC. The experimental results show that the proposed algorithm can effectively improve the volume of accepted SC while guaranteeing that the end-to-end delay of each packet satisfies the delay requirements.

Keywords: end-to-end delay, service chain, network function virtualization, network calculus

1 引言

随着 5G 网络的发展，远程医疗、自动驾驶、AR/VR、工业自动化等业务正在成为可能，并处于稳步推进的过程中。这些新业务对网络端到端时延提出了更严苛的要求。例如，自动驾驶需要网络提供低于 10 ms 的端到端时延，保证汽车能够及时地执行命令[1]。游戏玩家在体验 AR/VR 游戏时，需要网络提供低于 20 ms 的端到端时延，以消除游戏图像和动作不匹配造成的眩晕[2]。因此，提供端到端严格的时延保障对这些业务至关重要。

当今的网络依赖于众多的网络功能设备，文献[3]指出网络中近半的设备均为网络功能设备。随着网络功能虚拟化（NFV, network function virtualization）

收稿日期：2021−06−11；**修回日期**：2021−09−11
基金项目：国家重点研发计划基金资助项目（No.2020YFB1805200）；国家自然科学基金资助项目（No.61872401, No.62132022）；霍英东青年教师基金资助项目（No.171059）
Foundation Items: The National Key Research and Development Program of China (No.2020YFB1805200), The National Natural Science Foundation of China (No.61872401, No.62132022), Fok Ying-Tong Education Foundation (No.171059)

技术[4]的普及，可以预见越来越多的网络功能设备将以虚拟网络功能（VNF, virtual network function）的形式进行部署。软件定义网络（SDN, software define network）[5]作为与 NFV 互补的技术，可以实现流量在 VNF 之间按顺序进行传递，从而实现 VNF 按序串成链来提供网络服务，这也称为服务链。在面对时延敏感业务的服务链请求时，如何保障服务链端到端的时延吸引了学术界和工业界的关注。

现有许多文献[6-9]研究了如何通过优化服务链的部署来为服务链提供端到端的时延保障。但是这些工作都无法为服务链提供严格的时延保障。例如，文献[6-7]在建模的过程中，认为数据包通过 VNF 节点的时延是固定的，这将对服务链的端到端时延建模带来较大的误差。由于 VNF 节点分配的资源大小是可以变化的，而 VNF 节点的数据包处理速率将随着分配资源的大小而改变，从而影响数据包通过 VNF 节点的时延。文献[8-9]通过使用排队论来避免这一问题，然而，基于排队论计算得到的时延为数据包平均时延，不能保证每一个数据包均能在该时延内完成传输。显然，这对一些时延敏感的业务是不可容忍的。

为了解决上述问题，一些工作提出了基于网络演算计算服务链的端到端时延。由于基于网络演算得到的时延为时延上限，因此能保证全部的数据包在该时延内完成传输。例如，文献[10]率先将网络演算应用于计算服务链中单条业务流的端到端时延中。由于不同的服务链会共享网络基础设施，不同的业务流量之间会存在资源竞争。因此，文献[10]中的模型不能扩展到多条服务链的场景。文献[11]考虑了服务链之间的资源竞争并基于随机网络演算推导了服务链的端到端时延，并将推导得到的结果在实验仿真中进行了验证。然而，上述工作仅仅是基于网络演算推导了服务链的端到端时延，并未进一步将网络演算与服务链的部署相结合。

本文将网络演算应用到服务链的部署问题中，旨在保障部署后的服务链具有严格的端到端时延上界。严格的时延保障指经过服务链的每一个数据包的时延都符合业务的需求。这种为业务流量提供严格时延保障的服务链称为确定性服务链（DSC, deterministic service chain）。在 DSC 的部署问题中，DSC 的目标是为接收的服务链请求提供严格的端到端时延保障的同时，提高接收的服务链的数量。首先对 DSC 的部署问题进行建模，由于网络演算在推导服务链时延的过程中引入了非线性约束，因此问题最终被建模为混合整数非线性规划。为了简化问题的求解，DSC 问题被分解为 2 个子问题，分别是服务链路由子问题与 VNF 节点资源分配子问题，通过引入最大允许的 VNF 总时延，实现对 2 个子问题的协同优化。最后，通过实验验证了所提模型和算法的有效性，结果显示所提算法能在提高接收的服务链数量的同时，严格保障接收的每一条服务链的端到端时延。

2　网络演算基本概念

时延是网络性能的一个重要指标，目前存在多种不同的工具和方法对网络端到端性能进行建模和分析。其中最常见的工具是排队理论，它将数据包的到达建模为泊松过程，并计算平均的时延和队列长度。然而，文献[12]指出，网络中的流量具有自相似性和突发性，因此泊松过程不能准确地描述流量的到达特征。此外，排队论得到的时延为平均时延，不能保证每个数据包都能在该时延内完成传输。因此，另一种工具——网络演算应运而生。

在网络演算中，最基本且最关键的概念是到达曲线、服务曲线和服务曲线串联。到达曲线描述流将要发送的最大数据量，服务曲线描述服务节点保证处理的最小数据量。如图 1 所示，假设业务到达流量经过了令牌桶过滤器（TBF, token bucket filter），TBF 的发送速率为 ρ，令牌桶的大小为 σ，则经过 TBF 的流的到达曲线可表示为 $\rho t+\sigma$。业务流量被发送到节点进行处理，假设该节点的服务曲线为 $\gamma(t-\theta)$，该服务曲线表示该节点的处理速率为 γ，数据包在得到处理之前将等待 θ 时间[13]。基于到达曲线和服务曲线，可以得到 2 个重要概念，分别是时延上限和队列长度上限。时延上限的值等于到达曲线与服务曲线的最大水平偏差，队列长度上限的值等于到达曲线和服务曲线的最大垂直偏差。如果用 $\alpha(t)$ 和 $\beta(t)$ 分别表示到达曲线和服务曲线，则时延上限的表达式为式(1)，队列长度上限的表达式为式(2)。在这里，可以得到时延上限的值为 $\sigma/\gamma+\theta$，队列长度的上限为 $\rho\theta+\sigma$。

$$\sup_{t\geqslant 0}\{\inf_{d\geqslant 0}\{d:\alpha(t)\leqslant\beta(t+d)\}\} \tag{1}$$

$$\sup_{t\geqslant 0}\{\alpha(t)-\beta(t)\} \tag{2}$$

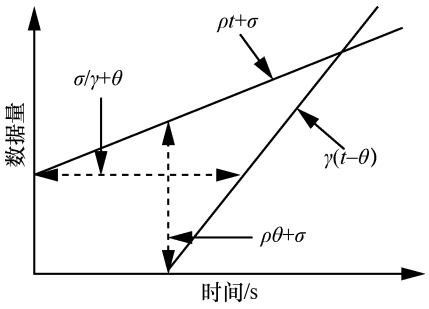


图1　网络演算中的到达曲线和服务曲线

上述为网络演算在单个节点中的运用，然而在网络中，通常存在多个节点串联的情况。此时，端到端的时延上限或队列长度上限不能简单地通过累加单个 VNF 节点上的时延上限或队列长度上限来获得。网络演算理论给出了计算多节点串联情况下的端到端服务曲线的方法。如图 2 所示，2 个独立的节点分别对应 2 条服务曲线，假设这两个节点的服务曲线分别为 $\beta_1(t)=\gamma_1(t-\theta_1)$ 和 $\beta_2(t)=\gamma_2(t-\theta_2)$。然后，串联后的 2 个节点的端到端服务曲线为 $\beta_1(t)\otimes\beta_2(t)$，其中符号 $\otimes$ 表示最小化卷积，其具体表达式如式(3)所示。在图 2 的情况中，2 个节点串联后的端到端服务曲线是 $\gamma_2(t-\theta_1-\theta_2)$。

$$\beta_1(t)\otimes\beta_2(t)=\inf_{0\leqslant s\leqslant t}\{\beta_1(t-s)+\beta_2(s)\} \tag{3}$$

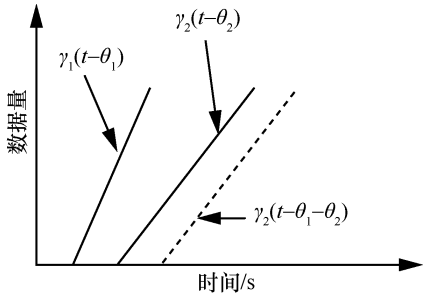


图2　网络演算中节点串联后的服务曲线

网络演算能够计算多个网络节点串联后端到端的服务曲线，而服务链则由多个 VNF 节点串联组成。这意味着，当得到服务链中每个 VNF 的服务曲线后，网络演算能够计算服务链的端到端服务曲线，从而计算数据包经过服务链的时延上限。因此，网络演算在本文问题场景中具有良好的适用性。

3　问题描述

本节给出了 DSC 部署问题的系统模型和数学建模。DSC 的目标是为每一个接收的网络业务请求确定最佳的服务链部署方案，在保证服务链端到端时延严格满足业务需求的同时，使接收的服务链总量最大。

3.1　系统模型

1) 底层网络基础设施

底层网络建模为有向图，其中 $\overline{\mathcal{V}}$ 表示底层物理节点的集合。底层物理节点可能是一个边缘计算节点或者是一个小型的数据中心。$\overline{v}_n$ 表示集合中的第 n 个底层物理节点，其可用资源表示为 $\overline{c}_n$。二进制变量 $\overline{e}_{nn'}$ 表示从 $\overline{v}_n$ 至 $\overline{v}_{n'}$ 之间存在物理链路。底层物理链路 $\overline{e}_{nn'}$ 的可用带宽和传输时延分别表示为 $\overline{b}_{nn'}$ 和 $\overline{t}_{nn'}$。特别地，$\overline{b}_{nn'}=\overline{b}_{n'n}$ 且 $\overline{t}_{nn'}=\overline{t}_{n'n}$ 表示同一底层物理链路的 2 个方向共享相同的带宽和传输时延。不同的底层物理节点支持不同类型的 VNF。Ψ 表示网络中支持的全部 VNF 类型的集合，ψ_h 为集合中第 h 个 VNF 类型。假设一个底层物理节点可以支持一部分 VNF 类型，$P_{hn}=1$ 表示第 h 个 VNF 类型 ψ_h 可以在 $\overline{v}_n$ 节点中被支持。每个底层节点中同一类型的 VNF 将被视为一个整体，将按照整体分配一定数量的资源。

2) 网络业务请求

在本文问题中，假设全部的网络业务请求同时全部到达（模型和算法也支持业务请求逐个到达的情况）。对于每一个网络业务请求，网络运营商需要决定是否对其接收。使用 $\mathcal{S}=\{s_1,s_2,\cdots,s_K\}$ 表示全部网络业务请求的集合，其中 K 为集合 $\mathcal{S}$ 中业务请求的数量。第 k 个网络业务请求包含一条服务链。服务链由一组按序排列的 VNF 节点 F^k 和一组虚拟链路 L^k 组成，为了便于说明，分别用 f_i^k 和 l_j^k 表示业务请求中的第 i 个 VNF 节点和第 j 条虚拟链路。服务链中的每一个 VNF 都有一种对应的类型，$Q_{ih}^k=1$ 表示 f_i^k 的类型为 ψ_h。此外，每一种类型的 VNF 都会被分配一定的资源。业务请求除了包含的服务链，还会指定业务流量的起点和终点，分别表示为 $\overline{v}_{n_{\text{src}}^k}$ 和 $\overline{v}_{n_{\text{dst}}^k}$。业务流量由源点的用户生成，在进入网络之前首先会经过 TBF 进行整形。TBF 的参数包含 ρ^k 和 σ^k，这意味着允许终端用户一次性发送大小为 σ^k 的数据量，但平均的发送速率不能超过 ρ^k。业务流量从入口节点开始，依次按序经过全部的 VNF 节点，最终到达终点节点。所有数据包的端到端时延应满足业务的时延要求，记为 D^k。

3) 服务链部署示例

以一个例子来介绍业务请求中服务链的部署。部署主要包括 VNF 节点的映射、虚拟链路的映射以及 VNF 节点的资源分配。部署示例如图 3 所示。假设业务请求中包含的服务链由 3 个 VNF 组成，源点为底层物理节点 1，终点为底层物理节点 4。为了便于表示服务链的源点和终点，构造了一个伪头部 VNF f_{src}^k 和伪尾部 VNF f_{dst}^k。这 2 个伪 VNF 提前映射到源点和终点。接下来，需要确定每个 VNF 和虚拟链路的映射。在本例中，VNF_1 映射到底层物理节点 2，而 VNF_2 和 VNF_3 都映射到物理节点 3。业务流量通过最短路径依次遍历底层物理节点 1、2、3、4。同时虚拟链路被映射到对应的路径上，特别是，VNF_2 和 VNF_3 之间的虚拟链路并没有映射到物理链路，而是映射到底层物理节点 3 内部的链路。当服务链中的节点和链路的映射确定后，需要为 VNF 所映射的 VNF 实例分配资源，即完成了一条服务链的部署。

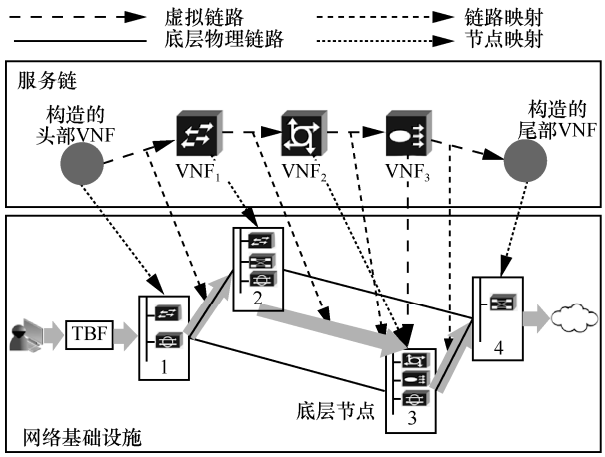


图 3　系统模型中的服务链部署示例

3.2　数学建模

本节将基于上述系统模型，对 DSC 的部署问题进行数学建模。

1) DSC 部署问题的优化目标

DSC 部署问题的优化目标是最大化接收的业务量，且保证服务链的端到端上限时延严格满足业务的需求。接收的业务总量表示为式(4)，其中 W^k 表示是否接收业务请求 s^k。

$$\sum_{k=1}^{K} W^k \rho^k \tag{4}$$

2) 底层物理节点容量限制

在系统模型中，R_{hn} 表示分配给节点 $\overline{v}_n$ 上类型为 ψ_h 的 VNF 的总资源量。式(5)保证了分配给一个底层物理节点上所有类型的 VNF 的资源总量不能超过该底层物理节点的资源容量。

$$\sum_{h=1}^{H} R_{hn} \leqslant \overline{c}_n,\ \forall n \tag{5}$$

3) 底层物理链路容量限制

底层物理链路也有带宽限制。首先，为了表示虚拟链路是如何映射到底层物理链路的，定义变量 $Z_{jnn'}^k$。如果业务请求 s_k 中的第 j 条虚拟链路映射到 $\overline{v}_n$ 至 $\overline{v}_{n'}$ 之间的物理链路，则 $Z_{jnn'}^k = 1$。此外，物理链路 2 个方向的流量将共享带宽。因此，式(6)保证了物理链路上的总流量满足其带宽限制。

$$\sum_{k=1}^{K}\sum_{j=1}^{J^k}\left(Z_{jnn'}^k + Z_{jn'n}^k\right)\sigma^k \leqslant \overline{e}_{nn'}\overline{b}_{nn'},\ \forall n,n' \tag{6}$$

4) VNF 映射限制

为了表示服务链中的 VNF 是如何映射到底层物理节点的，定义变量 Y_{in}^k，$Y_{in}^k = 1$ 表示业务 s_k 中的第 i 个 VNF 被映射到 $\overline{v}_n$ 上。式(7)确保了接收的每一个请求中的每个 VNF 都完成映射且仅被映射一次。

$$\sum_{n=1}^{N} Y_{in}^k = W^k,\ \forall k,i \tag{7}$$

5) VNF 类型限制

在系统模型中，假设每个底层物理节点支持部分的 VNF 类型，例如，高性能防火墙需要 FPGA 硬件加速，则只有配备了 FPGA 的物理节点才能支持该网络功能。式(8)表示一个 VNF 只能被映射到支持该 VNF 类型的底层物理节点。

$$Y_{in}^k \leqslant \sum_{h=1}^{H} Q_{ih}^k P_{hn},\ \forall k,n,\forall i \in [2, I^k - 1] \tag{8}$$

6) 虚拟链路映射限制

每条虚拟链路可以从 2 个不同的方向映射到一条物理链路。此外，每条虚拟链路可以映射到多条物理链路。因此，使用式(9)保证一条虚拟链路不会同时映射到物理链路的 2 个方向上，以避免产生环路。此外，使用式(10)保证每个底层物理节点的输出流量总量与输入流量总量相等，如此，同一个业务请求中的虚拟链路最终可以形成端到端连续的路由路径。

$$Z^k_{jnn'} + Z^k_{jn'n} \leqslant 1,\ \forall k, j, n, n' \tag{9}$$

$$\sum_{n'=1}^{N}\left(Z^k_{jnn'} - Z^k_{jn'n}\right) = \sum_{i=j}\left(Y^k_{in} - Y^k_{i+1,n}\right),\ \forall n, k, j \tag{10}$$

7) 端到端时延限制

最后一个限制条件是端到端时延的限制。D^{k*} 表示部署后的服务链的端到端时延上限。D^{k*} 的确切形式将在第 4 节中详细介绍。式(11)表示服务链的端到端时延应严格满足业务的时延要求。

$$D^{k*} \leqslant D^k, \forall k \tag{11}$$

4 服务链端到端时延上限计算

在问题的数学建模中，服务链部署后的端到端时延暂时用 D^{k*} 表示。本节将应用网络演算推导 D^{k*} 的具体表达式。

根据第 2 节给出的示例，需要计算服务链的端到端时延上限，则首先需要获取业务流量的到达曲线以及服务链的端到端服务曲线。在系统模型中，业务流量在进入网络之前由 TBF 进行整形。因此，业务流量的到达曲线很容易得到。式(12)表示业务 s_k 的到达曲线。

$$\rho^k t + \sigma^k \tag{12}$$

接下来，将推导业务 s_k 的端到端服务曲线。业务流量在网络中会经过 VNF 节点、物理交换机和物理链路，这些网元都有其各自的服务曲线。首先，研究 VNF 节点的服务曲线。本文使用速率−时延服务曲线对 VNF 节点进行建模。由于分配给 VNF 的资源是可变化的，因此 VNF 服务曲线中的速率参数会随之变化。为此，通过实验验证了 VNF 节点的速率与分配的资源之间的关系。实验中开发了 2 种示例性质的 VNF，分别为 VNF-Firewall 和 VNF-NAT。调整分配给 VNF 的 CPU 资源的百分比并测量 VNF 的速率，例如当分配的 CPU 资源为 40% 时，则意味着数据包处理进程消耗了 CPU 40%的时间片，结果如图 4 所示。从图 4 可以看出，VNF 的速率与分配的资源量成正比。因此，分配的资源与速率的关系如式(13)所示，其中 λ_{hn} 是一个常数，可以通过实验获得。最终，$\overline{v}_n$ 上类型为 ψ_h 的 VNF 的总服务曲线如式(14)所示。

$$\frac{R_{hn}}{\lambda_{hn}} = \gamma_{hn} \tag{13}$$

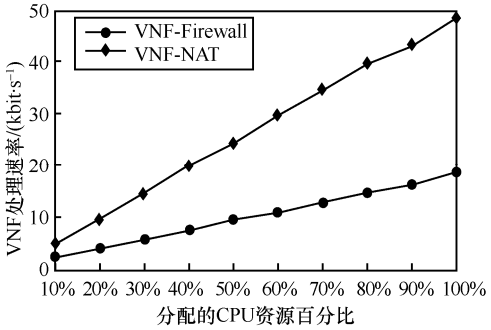


图 4 VNF 处理速率与所分配资源的关系

$$\frac{R_{hn}}{\lambda_{hn}}\left(t - \theta_{hn}\right) \tag{14}$$

式(14)中的服务曲线代表的是整个 VNF 的服务曲线。然而，多个服务链可能共享同一个 VNF 并竞争资源。因此，对于每一条服务链，需要计算其在 VNF 上独立的服务曲线。基于单个 VNF 上独立的服务曲线，计算服务链端到端时延上限的一种简易方法是根据业务的到达曲线和 VNF 上独立的服务曲线计算服务链经过每一个 VNF 的时延上限，然后端到端的时延上限为每一个 VNF 上的时延上限的累加。然而，由于“pay burst once”现象的存在，这种简易的方法会放大端到端的时延上限。因此，正确的方法是在考虑服务链之间竞争的情况下，计算服务链端到端整体的服务曲线，并根据端到端整体的服务曲线和业务流量的到达曲线计算端到端时延上限。

为此，首先需要推导服务链经过单个 VNF 时独立的服务曲线。根据文献[14]可知 s_k 所经过的 VNF 上交叉流量的到达曲线。式(15)表示底层物理节点 $\overline{v}_n$ 上类型为 ψ_h 的 VNF 上的总流量大小。基于式(15)，S_k 中第 i 个 VNF f^k_i 所在的 VNF 实例上的总流量大小可以表示为式(16)。为了简化 T^k_i 的表达式，定义一个变量 χ^k_i，其值如式(17)所示。然后，T^k_i 的值可以简化为式(18)，s_k 所经过的 VNF 上交叉流量的到达曲线可以表示为式(19)。根据文献[14]中第 4.B 节中给出的理论，针对 f^k_i 的独立服务曲线仍然为 rate-latency 的类型，其具体形式如式(20)所示，其中 γ^k_i 和 θ^k_i 的值分别如式(21)和式(22)所示。

$$T_{hn} = \sum_{k=1}^{K}\sum_{i=1}^{I^k} Y^k_{in} Q^k_{ih}\left(\rho^k t + \sigma^k\right),\ \forall n, h \tag{15}$$

$$T_i^k=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^kT_{hn}=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\sum_{k=1}^{K}\sum_{i=1}^{I^k}Y_{in}^kQ_{ih}^k\rho^kt+\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\sum_{k=1}^{K}\sum_{i=1}^{I^k}Y_{in}^kQ_{ih}^k\sigma^k,\ \forall k,i \tag{16}$$

$$\chi_i^k=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\sum_{k=1}^{K}\sum_{i=1}^{I^k}Y_{in}^kQ_{ih}^k,\ \forall k,i \tag{17}$$

$$T_i^k=\chi_i^k\rho^kt+\chi_i^k\sigma^k \tag{18}$$

$$\left(\chi_i^k-1\right)\rho^kt+\left(\chi_i^k-1\right)\sigma^k \tag{19}$$

$$\gamma_i^k\left(t-\theta_i^k\right) \tag{20}$$

$$\gamma_i^k=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\frac{R_{hn}}{\lambda_{hn}}-\left(\chi_i^k-1\right)\rho^k \tag{21}$$

$$\theta_i^k=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\left(\theta_{hn}+\frac{\lambda_{hn}}{R_{hn}}\left(\chi_i^k-1\right)\sigma^k\right) \tag{22}$$

在得到 s_k 中单个 VNF 独立的服务曲线后，可以根据网络演算理论轻松地得到 s_k 端到端的服务曲线。根据网络演算理论[15]，s_k 中多个串联的 VNF 的服务曲线如式(23)所示。符号 ⊗ 表示极小化卷积。s_k 端到端的服务曲线仍然是 rate-latency 类型，如果使用式(24)表示 s_k 的端到端服务曲线，则 γ^k 和 θ^k 的值分别如式(25)和式(26)所示。最后，可以基于网络演算理论得到 s_k 中所有 VNF 节点串联后的端到端时延上限为式(27)。

$$\gamma_1^k\left(t-\theta_1^k\right)\otimes\gamma_2^k\left(t-\theta_2^k\right)\otimes\cdots\otimes\gamma_{I^k}^k\left(t-\theta_{I^k}^k\right) \tag{23}$$

$$\gamma^k\left(t-\theta^k\right) \tag{24}$$

$$\gamma^k=\min\{\gamma_1^k,\gamma_2^k,\cdots,\gamma_{I^k}^k\} \tag{25}$$

$$\theta^k=\sum_{i=1}^{I^k}\theta_i^k \tag{26}$$

$$D_v^{k*}=\frac{\sigma^k}{\gamma^k}+\theta^k \tag{27}$$

由式(25)和式(26)可以观察到，端到端服务曲线的速率参数取决于全部串联的 VNF 节点中的最小速率，而端到端服务曲线中的时延参数是全部串联的 VNF 节点的时延之和。这一观察结果指导本文将更多的资源分配给瓶颈 VNF 节点，以提高串联的 VNF 节点中最小的速率，从而提高端到端服务曲线的速率。此外，这对研究交换机和物理链路的时延也具有一定的指导意义。交换机的服务曲线认为是 rate-latency 类型，可以表示为 $\gamma_s(t-\theta_s)$。由于交换机的处理速率 γ_s 大于全部的 VNF 节点，因此在端到端服务曲线中考虑交换机并不会影响 γ^k 的值。因此，数据包每经过一个交换机只会为数据包增加一个固定的时延值，即 θ_s。服务链路由路径中包含的交换机数量等于包含的物理链路的数量减 1。因此，数据包经过交换机产生的时延的值如式(28)所示。

$$D_s^{k*}=\left(\sum_{j=1}^{J^k}\sum_{n=1}^{N}\sum_{n'=1}^{N}Z_{jnn'}^k-1\right)\theta_s \tag{28}$$

对于物理链路，文献[14]指出其服务曲线是一个脉冲函数，即速率为无限大，时延为一个固定值。这也符合常识的判断，物理链路中的数据包不会产生排队，数据包经过物理链路的时延等于物理链路的传输时延。在系统模型中，使用 $\overline{t}_{nn'}$ 表示 $\overline{v}_n$ 至 $\overline{v}_{nn'}$ 的传播时延。因此，数据包经过物理链路产生的时延的值如式(29)所示。

$$D_l^{k*}=\sum_{j=1}^{J^k}\sum_{n=1}^{N}\sum_{n'=1}^{N}Z_{jnn'}^k\overline{t}_{nn'} \tag{29}$$

业务请求 s_k 部署后的服务链端到端时延 D^{k*} 由 3 部分组成，分别是 VNF 时延 D_v^{k*}、交换机时延 D_s^{k*} 以及链路时延 D_l^{k*}。最终，D^{k*} 的值如式(30)所示。

$$D^{k*}=D_l^{k*}+D_s^{k*}+D_v^{k*} \tag{30}$$

5　服务链部署算法

本节介绍一种名为 JRRA（joint routing and resource allocation）的启发式算法来解决 DSC 的部署问题。JRRA 首先确定最优服务链路由路径，该路径依次包含所有需要的 VNF 节点，然后确定包含的 VNF 节点的资源分配量。服务链路由路径的选择和 VNF 节点的资源分配量将决定服务链的端到端时延。虽然已有许多相关文献[16]研究了不同场景下的服务链优化部署问题，但是尚没有算法能运用于 DSC 问题，同时本问题还存在 3 个挑战。

第一个挑战来自服务链的路由。在不考虑 VNF 节点时延的情况下，需要找到端到端时延最小的路径，同时，该路径需要满足一些额外的要求。第二个

挑战来自 VNF 节点的资源分配。从式(25)、式(26)和式(27)可以看出，服务链中任一 VNF 节点分配的资源量将影响数据包通过服务链的时延。因此，服务链中所有 VNF 节点的资源分配应该协同进行考虑。此外，由于不同的服务链之间会共享 VNF 节点，不同的服务链中共享的 VNF 节点的资源分配也应该协同进行考虑。第三个挑战来自服务链路由与 VNF 节点分配的协同，由于两者都将对服务链的端到端时延产生影响，因此需要将两者协同进行考虑。

JRRA 算法分为两步。第一步，将服务链路由建模成了一个可解的线性规划问题，此外还提出了一种基于多层拓扑的启发式服务链路由方法来解决第一个挑战。第二步，通过推导确定为每个 VNF 节点分配的资源量来解决第二个挑战。通过引入最大允许的 VNF 时延参数将第一步与第二步进行协同来解决第三个挑战。

5.1 算法概述

算法 1 JRRA 算法主流程

1) 将 $\mathcal{S}=\{s_k\}$ 集合中的业务请求按 ρ^k / J^k 的大小从大到小进行排序

2) for $s_k \in \mathcal{S}$ do

3) 为 s_k 构造拓扑 $\mathcal{G}^k$

4) while True do

5) FS-Path $\leftarrow$ FeasibleShortestPath($\mathcal{G}^k, s_k$)

6) if FS-Path 不为空 then

7) $D_{v-\max}^{k*}$ 的值等于 D^k 减去 FS-Path 的时延

8) U-VNFs $\leftarrow$ VNF_Resource($s_k, D_{v-\max}^{k*}$)

9) if U-VNFs 不为空 then

10) 在拓扑 $\mathcal{G}^k$ 中将 U-VNFs 包含的 VNF 删去

11) continue

12) else

13) s_k 被接收了，即 $W^k=1$; Break

14) end if

15) else

16) s_k 被拒绝了，即 $W^k=0$; Break

17) end if

18) end while

19) end for

JRRA 算法的流程如算法 1 所示，首先根据 ρ^k / J^k 的值（每单位服务链长度的吞吐）降序排列所有业务请求，然后按序逐个部署业务请求。对于单个业务请求的部署，将按照图 5 所示的示例进行说明。假设图 5 中的业务请求 s_k 的带宽要求为 5 Mbit/s，业务从 $\bar{v}_{n_{\text{src}}^k}$（节点 A）开始，到 $\bar{v}_{n_{\text{dst}}^k}$（节点 F）结束，所需的 VNF 类型依次为 VNF_1、VNF_2 和 VNF_3。首先去除剩余带宽小于 5 Mbit/s 的物理链路，构造一个拓扑 $\mathcal{G}^k$（算法 1 的第 3 行），即节点 A 和节点 B 之间的物理链路将被移除。拓扑 $\mathcal{G}^k$ 中剩余的每条链路的时延等于其原来的时延加上它所连接的交换机的转发时延的一半，这是为了将节点时延转移至链路时延。接下来的目标是在 $\mathcal{G}^k$ 中为 s_k 找到最短且可行路径（FS-Path, feasible and shortest path）。FS-Path 应满足如下 3 个要求。1) 从 $\bar{v}_{n_{\text{src}}^k}$ 开始，到 $\bar{v}_{n_{\text{dst}}^k}$ 结束。2) 路径上所有物理链路的剩余带宽都能够承载业务。3) 端到端时延（仅包含交换机时延和链路传输时延，不包含 VNF 节点时延）最小。如何找到 FS-Path 的方法将在 5.2 节中介绍，在这里使用一个函数 FeasibleShortestPath( )表示该方法（算法 1 的第 5 行）。如果找不到 FS-Path，s_k 将被拒绝。否则，将继续在 FS-Path 中为其包含的 VNF 节点分配资源。

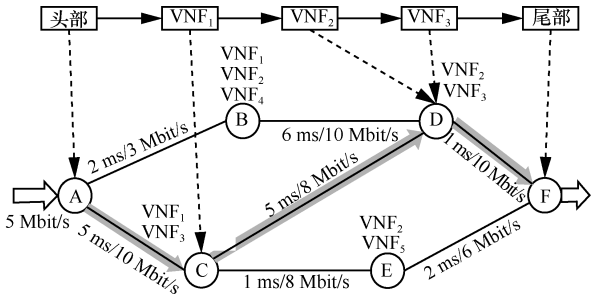


图 5 JRRA 算法部署服务链示例

在 VNF 节点资源分配的过程中，引入一个参数，名为最大允许 VNF 节点时延 $D_{v-\max}^{k*}$。$D_{v-\max}^{k*}$ 的值等于业务的端到端时延需求 D^k 减去 FS-Path 的总时延（算法 1 的第 7 行）。然后，分配给每个 VNF 节点资源可以根据 $D_{v-\max}^{k*}$ 推导得到，具体推导过程在 5.3 节中描述，这里使用函数 VNF_Resource( ) 来替换它（算法 1 的第 8 行）。由于分配给每个 VNF 节点的资源数量在寻找服务链路由的过程中是不可预测的，因此无法确定底层物理节点是否能够为 VNF 节点提供足够的资源。因此，VNF 节点所需的资源可能存在不能被满足的情况。为了处理这种

情况，函数 VNF_Resource( )将返回资源不能被满足的 VNF 节点，这些 VNF 节点包含在 U-VNFs 集合中。U-VNFs 集合中的节点将从拓扑 $\mathcal{G}^k$ 中删除。然后，在更新后的 $\mathcal{G}^k$ 将重新寻找 FS-Path（算法 1 的第 11 行）。如果所有 VNF 节点所需要的资源都可以满足，则业务请求 s_k 将被接收。

5.2　服务链路由与节点映射

本节将介绍算法 1 中函数 FeasibleShortestPath() 的实现。根据 FS-Path 的要求，可以通过求解以下 ILP 模型来找到 FS-Path。

$$\min\ D_s^{k*} + D_l^{k*}$$

$$\text{s.t.}\ \ \text{式(6)}\sim\text{式(10)}$$

$$\text{var.}\ W^k, Y_{in}^k, Z_{jnn'}^k = \{0,1\}, \forall k, i, j, n, n'$$

然而，当网络规模较大时，基于 ILP 的解决方案面临计算时间长的问题。因此，本文也提出了一种高效的启发式算法来确定 FS-Path。经典的 Dijkstra 算法[17]可以得到 2 个节点之间的最小加权路径，即时延最小的路径。然而，Dijkstra 无法找到按序经过所需的 VNF 的最小加权路径。文献[18]中提出的基于多层拓扑的服务链路由方法可以有效地解决这一问题，但该方法不能保证路径包含的物理链路的剩余带宽能够满足业务的需求。虽然剩余带宽小于业务带宽需求的物理链路已经在拓扑 $\mathcal{G}^k$ 中被提前移除，但业务流量可能会多次经过同一条物理链路，从而导致物理链路的剩余带宽出现不能够满足需求的情况。例如，在图 5 中，当业务的路由路径为 A→C(VNF$_1$)→E(VNF$_2$)→C(VNF$_3$)→E→F 时，端到端时延为 10 ms，小于图中标注的路由路径。但是，业务流量在节点 C 和节点 E 之间的物理链路上经过了 3 次，因此对该链路的带宽要求为 15 Mbit/s，然而该链路的剩余带宽仅为 8 Mbit/s。因此，本节提出了一个时延惩罚因子 α（一个大于 1 的常数），并将其应用于文献[18]所提方法中。当文献[18]中的方法找到的路由路径包含剩余带宽不足的物理链路时，将这些物理链路的时延乘以 α。如此，这些物理链路在最小加权路径的寻找过程中将逐渐不被优先选择。

算法 2　JRRA 服务链路由算法

1) 构造一个多层拓扑 $\mathcal{G}_{\text{multi}}^k$
2) while True do
3) S-Path ← Dijkstra($\mathcal{G}_{\text{multi}}^k, \overline{v}_{n_{\text{src}}^k}, \overline{v}_{n_{\text{dst}}^k}$)
4) if S-Path 为空集 then
5) return ∅
6) end if
7) if S-Path 的时延大于 D^k then
8) return ∅
9) end if
10) U-Links ← BW_Checker(S-Path)
11) if U-Links 不为空集 then
12) U-Links 中链路的时延乘以 α
13) continue
14) else
15) FS-Path ← S-Path
16) return FS-Path
17) end if
18) end while

算法 2 给出了 JRRA 算法中完整的服务链路由算法过程。首先，基于拓扑 $\mathcal{G}^k$ 构造一个新的多层拓扑图，如图 6 所示，其中层数等于服务链的长度加 1，每一层的拓扑与 $\mathcal{G}^k$ 保持一致。层之间的连接取决于所需 VNF 类型的位置。例如，节点 B 和节点 C 支持的类型为 VNF$_1$，则节点 B$_1$ 与节点 B$_2$ 相连，节点 C$_1$ 与节点 C$_2$ 相连。这确保了当路由路径从第 1 层到达第 2 层时，必定会经过类型为 VNF$_1$ 的 VNF。类似地，将其他相邻的两层之间进行连接。每层中物理链路的时延与 $\mathcal{G}^k$ 保持一致，连接相邻两层的链路的时延设置为 0。指定节点 A$_1$ 和节点 F$_4$ 分别作为服务链路由的起点和终点，运用 Dijkstra 算法寻找到起点和终点之间的最小加权路径 S-Path（算法 2 中第 3 行），S-Path 将依次经过第一层到达第四层，也就是说，选择的 S-Path 也将依次包含所需的 VNF。然后，将检查得到的 S-Path，以保证其中所包含的物理链路的剩余带宽是足够的（算法 2 中第 10 行）。不合格的物理链路将加入 U-Links 集合，并将 U-Links 集合中的链路的时延乘以 α 后再重新寻找 S-Path（算法 2 中第 10～13 行）。最终，通过检查的 S-Path 即 FS-Path。

5.3　VNF 节点资源分配

本节将介绍算法 1 中的 VNF_Resource( )函数的实现。在推导 VNF 的资源数量之前，已经确定了最大允许的 VNF 节点时延，即 $D_{v-\max}^{k*}$。接下来，将基于 $D_{v-\max}^{k*}$ 的值，以业务请求 s_k 为例，推导出为 s_k 所对应的 FS-Path 包含的 VNF 节点分配的最优资

源量。分配的资源量应满足以下 3 个要求。1) 端到端总的 VNF 节点时延≤ $D_{v-\max}^{k*}$；2) 在 s_k 之前接收部署的业务的端到端总 VNF 节点时延不应增加；3) VNF 节点的处理速率应大于所承载的全部业务的带宽需求。当满足上述 3 个要求时，分配给 VNF 节点的最小资源量即最优资源量。

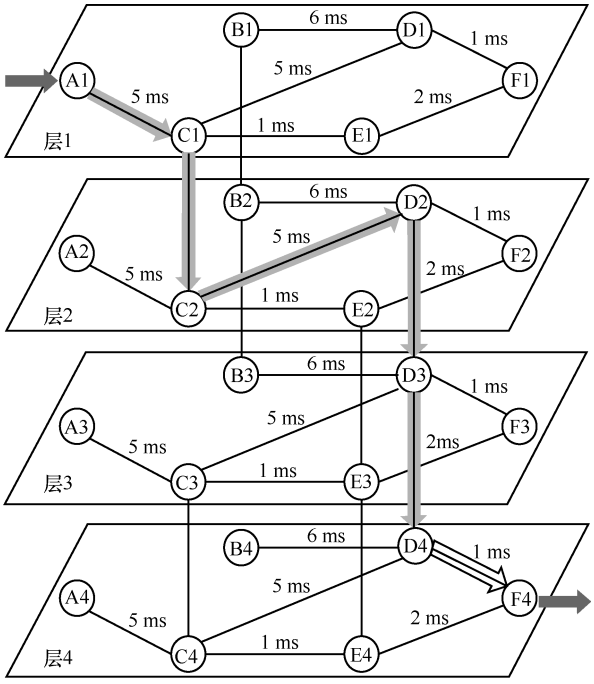


图 6 JRRA 算法中的多层拓扑示例

接下来，以 s_k 为例说明如何推导出最优的 VNF 资源分配方案。为了简化推导过程中使用到的表达式，首先定义了式(31)所示的变量。然后，式(21)和式(22)可以分别简化为式(32)和式(33)。其中 $\overline{R_i^k}$ 表示分配给 s_k 中第 i 个 VNF 部署所在的 VNF 实例的资源量。由于端到端的 VNF 总时延与分配给 VNF 的资源成反比，则当端到端的总 VNF 时延等于 $D_{v-\max}^{k*}$ 时，分配的资源量最小。因此，将 $D_{v-\max}^{k*}$ 和式(33)中 θ_i^k 的值代入式(27)，可以得到式(34)。

$$\overline{R_i^k}=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^kR_{hn},\ \overline{\lambda_i^k}=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\lambda_{hn}\ ,$$

$$\overline{\theta_i^k}=\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^kQ_{ih}^k\theta_i^k,\ \overline{\rho_i^k}=(\chi_i^k-1)\rho^k,$$

$$\overline{\sigma_i^k}=(\chi_i^k-1)\sigma^k \tag{31}$$

$$\gamma_i^k=\frac{\overline{R_i^k}}{\overline{\lambda_i^k}}-\overline{\rho_i^k} \tag{32}$$

$$\theta_i^k=\overline{\theta_i^k}+\frac{\overline{\lambda_i^k}}{\overline{R_i^k}}\times\overline{\sigma_i^k} \tag{33}$$

$$D_{v-\max}^{k*}=\frac{\sigma^k}{\gamma^k}+\sum_{i=1}^{I^k}\left(\overline{\theta_i^k}+\frac{\overline{\lambda_i^k}}{\overline{R_i^k}}\times\overline{\sigma_i^k}\right) \tag{34}$$

由式(25)可知，服务链端到端服务曲线的速率由服务链中处理速率最小的 VNF 决定，处理速率最小的 VNF 将限制端到端服务曲线的速率，从而增加端到端的 VNF 时延。因此，应该提高处理速率最小的 VNF 的处理速率，避免出现瓶颈。另一方面，如果某一 VNF 的处理速率大于服务链端到端服务曲线的速率，这对于提高服务链整体处理速率没有帮助，从而造成资源浪费。综上，服务链中的每一个 VNF 的处理速率都将与服务链端到端的处理速率保持一致，因此，服务链中的每个 VNF 都应该具有相同的速率，据此，可以得到式(35)。将式(35)中的 $\overline{R_1^k}$, $\overline{R_2^k}$, …, $\overline{R_{I^k}^k}$ 的表达式代入式(34)，可以得到 γ^k 的值。然后，再将 γ^k 的值代入式(35)，可以得到 $\overline{R_1^k}$, $\overline{R_2^k}$, …, $\overline{R_{I^k}^k}$ 的值。

$$\gamma^k=\gamma_1^k=\gamma_2^k=\cdots=\gamma_{I^k}^k\Rightarrow\gamma^k=\frac{\overline{R_1^k}}{\overline{\lambda_1^k}}-\overline{\rho_1^k}=$$

$$\frac{\overline{R_2^k}}{\overline{\lambda_2^k}}-\overline{\rho_2^k}=\cdots=\frac{\overline{R_{I^k}^k}}{\overline{\lambda_{I^k}^k}}-\overline{\rho_{I^k}^k}\Rightarrow\overline{R_1^k}=\overline{\lambda_1^k}\left(\gamma^k+\overline{\rho_1^k}\right),$$

$$\overline{R_2^k}=\overline{\lambda_2^k}\left(\gamma^k+\overline{\rho_2^k}\right),\cdots,\overline{R_{I^k}^k}=\overline{\lambda_{I^k}^k}\left(\gamma^k+\overline{\rho_{I^k}^k}\right) \tag{35}$$

然而，上述资源分配方案可能不能满足第二个和第三个要求。以业务 s_k 中第 i 个 VNF 所在的 VNF 实例为例，另一个业务 $s_{k'}$ 中的 VNF 可能在业务 s_k 部署之前就已经存在于该 VNF 实例中。在部署 s_k 时，需要重新确定分配给该 VNF 实例的资源量，应保证不增加 $s_{k'}$ 对应的 VNF 节点的总时延。为此，只需要保证 $\gamma^{k'}$ 的值不降低，$\theta^{k'}$ 的值不增加即可。因此，可以得到式(36)和式(37)。至此，第二个要求已经满足。根据第三个要求，即 VNF 节点的处理速率应大于所承载的全部业务的带宽需求，可以得到式(38)。最终，确定满足 3 个要求，即同时满足式(35)、式(36)、式(37)和式(38)的最小 VNF 节点资源分配量，即最优的 VNF 节点资源分配方案。

$$\gamma^{k'}\leqslant\sum_{n=1}^{N}\sum_{h=1}^{H}Y_{in}^{k'}Q_{ih}^{k'}\frac{R_{hn}}{\lambda_{hn}}-\left(\chi_i^{k'}-1\right)\rho^{k'} \tag{36}$$

$$\theta_i^{k'} \geqslant \sum_{n=1}^{N}\sum_{h=1}^{H} Y_{in}^{k'} Q_{ih}^{k'} \left(\theta_{hn} + \frac{\lambda_{hn}}{R_{hn}} \left(\chi_i^{k'} - 1 \right) \sigma^{k'} \right) \quad (37)$$

$$\frac{R_{hn}}{\lambda_{hn}} \geqslant \sum_{k=1}^{K}\sum_{i=1}^{I^k} Y_{in}^{k} Q_{ih}^{k} \rho^{k}, \ \forall n,h \quad (38)$$

综上，DSC 问题的优化目标是帮助服务提供商最大化接收的业务量，即最大化式(4)的值，同时通过协同优化服务链路由与 VNF 节点资源分配，使通过服务链的数据包的最大时延小于业务的时延需求，即保证每一个数据包均能在指定时延内完成传输。

6　实验分析

本节通过数值模拟来评估所提模型和算法的性能。首先介绍实验设置，然后从不同方面对算法的性能进行评估，并对实验结果进行讨论。

6.1　实验设置

为了获得准确的统计，每个数据点通过平均 10 次独立模拟的结果得到。基于 Python 进行数值模拟，利用 Python 中的 PySCIPOpt 套件[19]求解 ILP 模型。此外，运行数值模拟的计算机配备了主频为 3.40 GHz 的 CPU 以及大小为 16 GB 的内存。实验在 Internet Topology Zoo[20]提供的 2 个真实的网络拓扑中进行，其中网络拓扑 1 由 42 个物理节点和 66 条物理链路组成，而网络拓扑 2 由 13 个物理节点和 15 条物理链路组成。每个物理节点包含的 CPU 核数在[50,100]中随机生成，此外，每个物理节点对 VNF 类型的默认支持率为 70%，假如网络中存在 30 种不同类型的 VNF，则每个物理节点将随机支持其中的 21 种 VNF 类型。每条物理链路包含 2 个参数，包括可用带宽和传播时延。每条物理链路的带宽在[10,100] Gbit/s 中随机生成，传播时延在[1,5] ms 中随机生成。

为了生成业务请求，首先生成一组 VNF 类型，包含 30 种不同的 VNF 类型。对于每一种 VNF 类型，所需 CPU 资源和处理速率之间的关系已经在式(13)中进行了讨论。式(13)中的常数 λ_{hn} 在[0,1]中随机产生，例如，$\lambda_{hn}=0.5$ 表示在物理节点 $\bar{v}_n$ 上，类型为 ψ_h 的 VNF 处理 1 Gbit/s 的流量需要 0.5 个 CPU 核（占用 1 个 CPU 50%的运行周期）。业务请求中的服务链包含的 VNF 的数量从[2,7]中随机选择，每一个 VNF 的类型将从 VNF 类型的集合中随机选择。此外，每个业务请求的入口和出口节点从全部的物理节点中随机选择。每个业务请求的流量在进入网络前都将由 TBF 进行流量整形。每个业务请求中 TBF 的 rate 参数在[100,1 000] Mbit/s 中随机生成，burst 参数在[1,10] Mbit/s 中随机生成。最后，当网络拓扑 1 作为实验网络拓扑时，共随机生成 2 000 个业务请求，而当网络拓扑 2 作为实验网络拓扑时，共随机生成 500 个业务请求。如果算法能为业务请求找到一个可行的解决方案，则该请求将被接收，否则请求被拒绝。

本节对比了以下 2 种算法。1) JRRA-MLT 算法。JRRA-MLT 算法基于多层拓扑图为业务寻找服务链的路由。2) JRRA-ILP 算法。JRRA-ILP 算法与 JRRA-MLT 算法的不同之处在于 JRRA-ILP 通过 5.2 节中的 ILP 模型来寻找服务链的路由。JRRA-MLT 和 JRRA-ILP 算法均通过 5.3 节中的方法确定 VNF 节点的资源分配量。

6.2　实验结果

DSC 部署问题的目标是帮助服务提供商最大化接收的业务量，即式(4)所示的值。因此，在实验中，使用接收的业务量作为算法的性能评估指标，并测试了一些因素对算法性能的影响。

图 7 展示了采用不同的实验网络拓扑的情况下不同算法接收的业务量大小。从图 7 中可以看出，JRRA-ILP 算法的性能表现优于 JRRA-MLT 算法。这是因为基于 ILP 确定的服务链路由路径具有更小的端到端时延，当链路时延和交换机时延越小时，则对应的最大允许的 VNF 节点时延则变得更大，从而减少 VNF 节点对资源的使用。由于 VNF 节点资源使用的降低，在网络中 VNF 节点资源总量固定的情况下，所能服务的业务总量得到增加。此外，可以观察到，在 2 种网络拓扑中，JRRA-MLT 算法的性能仅比 JRRA-ILP 算法降低了 10%左右，这证明了所设计的基于多层网络拓扑的服务链路由算法能有效地寻找到可行且端到端时延相对较小的服务链路由路径。考虑到基于多层网络拓扑的服务链路由算法计算复杂度更低，该算法适用于大规模网络拓扑，并能取得与最优结果近似的性能表现。

1) 服务链长度的影响

首先评估了服务链长度对算法性能的影响。本文测试了 4 种不同范围的服务链长度，分别是(1,2]、[3,4]、[5,6]和[7,8)。评估结果展示在图 8 和图 9 中。可以观察到，在 2 种不同的实验网络拓扑中，随着服务链长度的增加，接收的业务总量均减小，造成

这一现象的原因有以下 2 个。首先，服务链长度的增加意味着服务链中包含的 VNF 数量的增加，则接收相同大小的服务链将消耗更多的 VNF 节点资源，因此，在物理节点资源总量一定的情况下，随着服务链长度的增加，接收的业务总量下降。第二个原因在于，随着服务链长度的增加，意味着服务链路由长度的增加，即服务链所经过的物理链路数量增加，这导致服务链端到端时延组成中的交换机时延和链路时延的占比提高，从而导致最大允许的 VNF 节点时延降低。因此，服务链需要占用更多的 VNF 节点资源来降低 VNF 节点的时延。在这双重原因叠加下，接收的业务总量迅速下降。

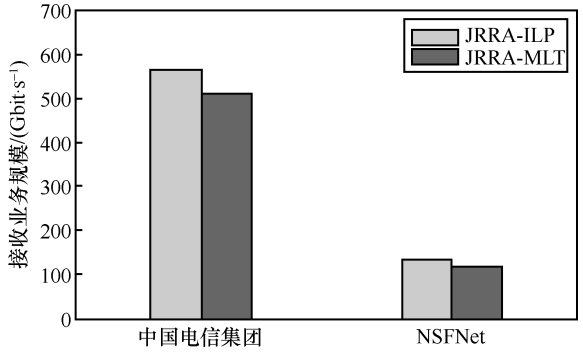


图 7 采用不同的实验网络拓扑的情况下不同算法接收的业务量大小

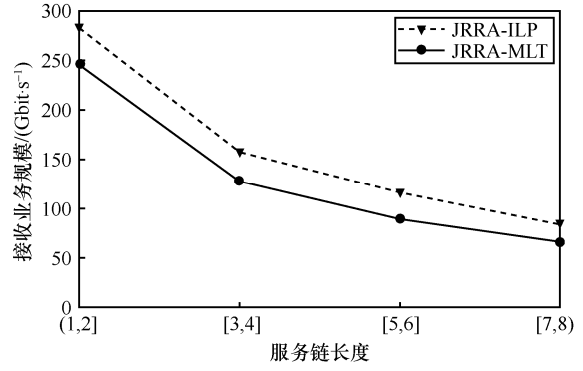


图 8 网络拓扑 1 中服务链长度的影响

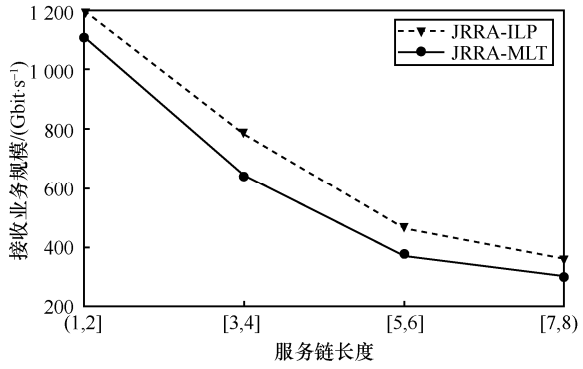


图 9 网络拓扑 2 中服务链长度的影响

2) 物理节点对 VNF 类型的支持率的影响

假设每个底层物理节点只能支持部分的 VNF 类型，将物理节点上支持的 VNF 类型数量占网络中 VNF 类型的总数量定义为物理节点对 VNF 类型的支持率，默认情况下每个物理节点的 VNF 类型支持率为 70%。这里评估物理节点对 VNF 类型的支持率对业务接收总量的影响。从图 10 和图 11 中可以看出，2 种算法接收的业务总量均随着物理节点对 VNF 类型的支持率的上升而增加。这可以通过多层网络拓扑进行解释，当物理节点对 VNF 类型的支持率上升时，同一种类型的 VNF 将在更多的物理节点上得到支持，这意味在服务链路由的过程中，多层网络拓扑的层与层之间的连接链路数将增加。如此，在源点和终点之间存在更多的可行路径可供选择，则找到端到端时延更小的服务链路由路径的概率也越大。当服务链路由路径的端到端时延越小时，最大允许的 VNF 节点时延将增加，服务链对 VNF 节点资源的占用将降低，从而更多的业务请求可以被接收。

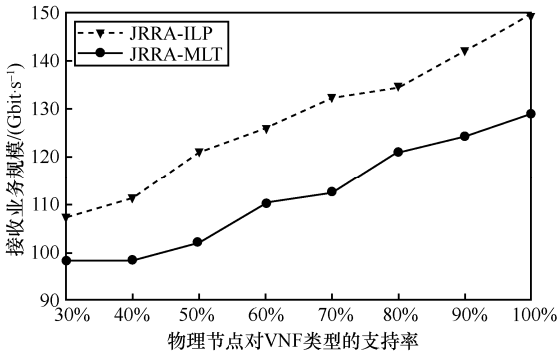


图 10 网络拓扑 1 中物理节点对 VNF 类型支持率的影响

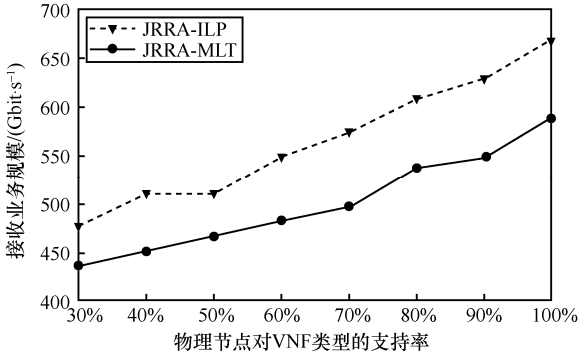


图 11 网络拓扑 2 中物理节点对 VNF 类型支持率的影响

3) 业务时延要求的影响

接下来，进一步研究业务时延要求的影响。默认情况下，业务的时延要求在[20,100] ms 中随机生

成。在产生业务请求的过程中将时延的要求固定，探究设置不同业务时延需求时接收的业务总量的变化。图 12 和图 13 显示，随着业务时延要求的上升，接收的业务总量也会增加。这一点很容易解释，根据算法 1 第 7 行，当业务的时延要求较大时，在服务链路由路径不变的情况下，最大允许 VNF 节点时延变大。在上文也多次讨论过了最大允许 VNF 节点时延变大将增大接收的业务总量。此外，观察到在网络拓扑 1 中，当业务时延要求设置为 20 ms 时，接收的业务总量显著下降。这是因为网络拓扑 1 的规模较大，当业务时延要求设置为 20 ms 时，部分业务因找不到可行的服务链路径而被直接拒绝（算法 2 中第 7～8 行）。由于网络拓扑 2 的规模较小，这种情况并未发生。

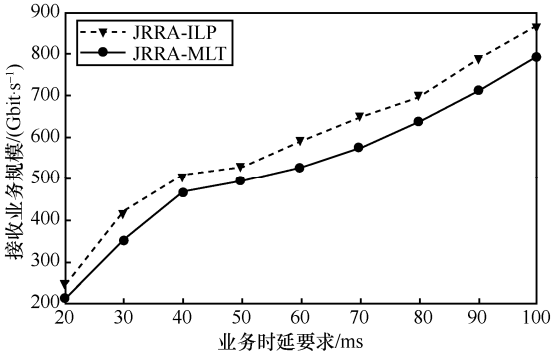


图 12　网络拓扑 1 中业务时延要求的影响

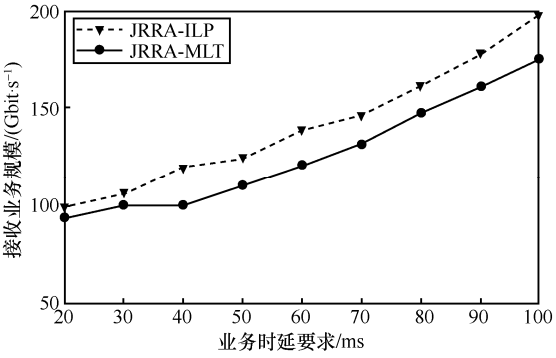


图 13　网络拓扑 2 中业务时延要求的影响

4) 业务在线到达场景下算法性能

最后，评估业务在线到达场景下算法的性能表现。由于不能提前得知全部业务请求的信息，因此无法对业务请求进行排序。每到达一个业务请求，运行算法对业务进行部署，若无法找到满足业务时延的部署方案，则业务请求被拒绝。图 14 和图 15 展示了随着累计到达的业务请求数量的增加，接收的业务总量的变化。可以看到，在前期业务请求到达时，JRRA-ILP 和 JRRA-MLT 算法均能为业务找到可行的部署方案而接收业务请求，从而 2 种算法的性能表现一致。随着接收的业务数量逐渐增加，由于 JRRA-MLT 算法比 JRRA-ILP 算法在接收同一个业务请求时将消耗更多的 VNF 节点资源，从而导致 JRRA-MLT 算法更早地耗尽了 VNF 节点资源。例如，在网络拓扑 1 中，当第 600 个左右的业务请求到达时，JRRA-MLT 的业务接收率开始下降，接收的业务总量开始落后于 JRRA-ILP。

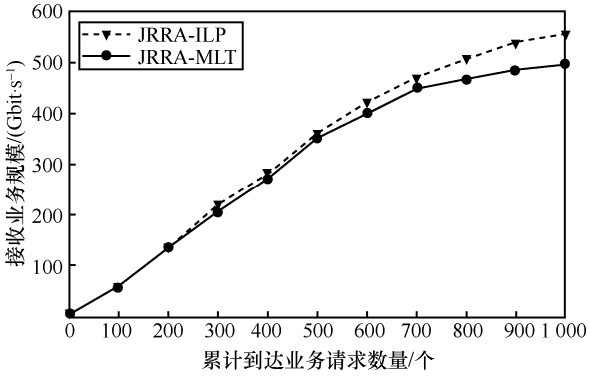


图 14　网络拓扑 1 中业务在线到达场景下算法性能表现

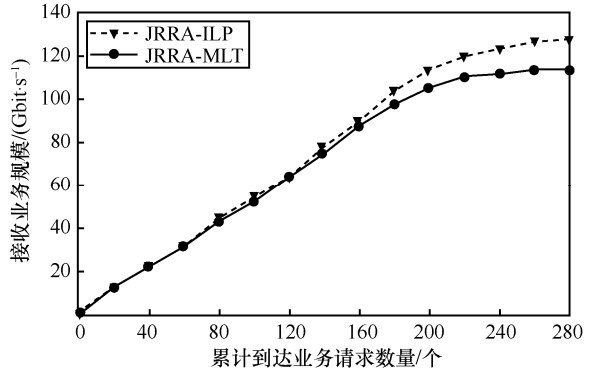


图 15　网络拓扑 2 中业务在线到达场景下算法性能表现

7　结束语

远程医疗、自动驾驶、工业自动化等业务对网络端到端时延的要求更加严格。本文研究了 DSC 的部署问题，旨在为网络业务中的服务链提供端到端时延严格保障的同时，最大化接收的业务总量。本文将网络演算运用于服务链的端到端时延计算中，实现为业务提供严格的端到端时延保障；将 DSC 部署问题建模为混合整数非线性规划问题，并将该问题拆解为服务链路由子问题和 VNF 节点资源分配子问题；将服务链路由子问题建模为可解的线性规划问题，同时也提出了一种高效的启发式

法。此外，理论推导了 VNF 节点的最佳资源分配值。最后，通过实验验证了所提算法的性能，结果显示所提算法在性能表现上与最优解接近。

本文研究成果在未来有望运用于部署了虚拟网络功能的 5G 核心网中，为远程医疗等业务提供确定性时延保障。不过在实际生产环境中设计解决 DSC 问题时，需要进一步考虑 VNF 实例的不同类型的服务曲线以及不同的部署模型，这也是 DSC 问题未来的研究方向。

参考文献：

[1] SAMII S, ZINNER H. Level 5 by layer 2: time-sensitive networking for autonomous vehicles[J]. IEEE Communications Standards Magazine, 2018, 2(2): 62-68.

[2] HUAWEI. Immersive VR and AR experiences with mobile broadband[EB]. 2021.

[3] SHERRY J, RATNASAMY S. A survey of enterprise middlebox deployments[R]. 2012.

[4] MIJUMBI R, SERRAT J, GORRICHO J L, et al. Network function virtualization: state-of-the-art and research challenges[J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 236-262.

[5] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[6] BARI F, CHOWDHURY S R, AHMED R, et al. Orchestrating virtualized network functions[J]. IEEE Transactions on Network and Service Management, 2016, 13(4): 725-739.

[7] QU L, ASSI C, SHABAN K, et al. A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks[J]. IEEE Transactions on Network and Service Management, 2017, 14(3): 554-568.

[8] GOUAREB R, FRIDERIKOS V, AGHVAMI A H. Virtual network functions routing and placement for edge cloud latency minimization[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2346-2357.

[9] BHAMARE D, SAMAKA M, ERBAD A, et al. Optimal virtual network function placement in multi-cloud service function chaining architecture[J]. Computer Communications, 2017, 102: 1-16.

[10] DUAN Q. Modeling and performance analysis for service function chaining in the SDN/NFV architecture[C]//Proceedings of 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). Piscataway: IEEE Press, 2018: 476-481.

[11] MIAO W, MIN G Y, WU Y L, et al. Stochastic performance analysis of network function virtualization in future Internet[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 613-626.

[12] PAXSON V, FLOYD S. Wide area traffic: the failure of Poisson modeling[J]. IEEE/ACM Transactions on Networking, 1995, 3(3): 226-244.

[13] LOCKWOOD J W, MONGA M. Implementing ultra low latency data center services with programmable logic[C]//Proceedings of 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects. Piscataway: IEEE Press, 2015: 68-77.

[14] FIDLER M. Survey of deterministic and stochastic service curve models in the network calculus[J]. IEEE Communications Surveys & Tutorials, 2010, 12(1): 59-86.

[15] LE BOUDEC J-Y, THIRAN P. Network calculus: a theory of deterministic queuing systems for the Internet[EB]. 2004.

[16] INTEL. Data plane development kit (DPDK)[EB]. 2021.

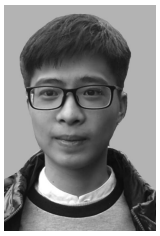
[17] JOHNSON D B. A note on Dijkstra's shortest path algorithm[J]. Journal of the ACM, 1973, 20(3): 385-388.

[18] DWARAKI A, WOLF T. Adaptive service-chain routing for virtual network functions in software-defined networks[C]//Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization. New York: ACM Press, 2016: 32-37.

[19] MILTENBERGER M. An interface from Python to the SCIP Optimization Suite[EB]. 2021.

[20] KNIGHT S, NGUYEN H X, FALKNER N, et al. The Internet topology zoo[J]. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765-1775.

[作者简介]



王泽南（1994− ），男，浙江湖州人，网络通信与安全紫金山实验室在站博士后，主要研究方向为网络功能虚拟化、网络智能等。

张娇（1986− ），女，河北保定人，博士，北京邮电大学副教授，主要研究方向为数据中心网络、网络功能虚拟化、软件定义网络、未来网络体系架构等。

汪硕（1991− ），男，河南灵宝人，博士，北京邮电大学讲师，主要研究方向为数据中心网络、软件定义网络、网络流量调度等。

黄韬（1980− ），男，重庆人，博士，北京邮电大学教授，主要研究方向为网络体系架构、软件定义网络、云网融合等。

F.Richard Yu（1974− ），男，加拿大卡尔顿大学教授、加拿大工程院院士，主要研究方向为互联网自主智能、自动驾驶、网络空间安全等。