支持多模态网络的可扩展异构服务功能链并行编排部署系统

陈浩[1,2,3]，杨芫[1,2]，徐明伟[1,2,3,4]，裴丹[1,2]，尤艺霖[2,3,4]

（1. 清华大学计算机科学与技术系，北京 100084；2. 北京信息科学与技术国家研究中心，北京 100084；
3. 中关村实验室，北京 100083；4. 清华大学网络科学与网络空间研究院，北京 100084）

摘 要：针对如何在万台服务器规模的多模态数据中心网络内，实现秒级的服务功能链扩容编排部署问题，提出一种可扩展的异构服务功能链并行编排部署（SHOD）系统。SHOD 系统对数据中心进行分区，并采用资源使用量最小化的并行编排器设计，从而将服务功能链扩容编排时间降低至秒级。SHOD 系统采用满足单一权职原则的中介者设计模式，实现了对异构设备的可扩展的编排部署和服务功能链建链。实验结果表明，SHOD 系统的扩容编排时间下降至 Daisy 的 $\frac{1}{12}$，同时仅引入 1.5 倍的 CPU 占用率开销。

关键词：多模态网络；网络功能虚拟化；服务功能链；可扩展性；编排部署系统

中图分类号：TN393

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022181

Parallel orchestration and deployment system for scalable heterogeneous service function chain supporting polymorphic network

CHEN Hao[1,2,3], YANG Yuan[1,2], XU Mingwei[1,2,3,4], PEI Dan[1,2], YOU Yilin[2,3,4]

1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
2. Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China
3. Zhongguancun Laboratory, Beijing 100083, China
4. Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China

Abstract: Aiming at the problem of second-level service function chain orchestration and deployment in the polymorphic data center with the scale of 10 000 servers, a scalable heterogeneous SFC parallel orchestration and deployment (SHOD) system was proposed. The scale-out orchestration time of service chain was reduced to seconds by partitioning the data center and adopting a parallel orchestrator design that minimizes resource usage. Using a mediator design model that satisfies the single-responsibility principle, the heterogeneous equipment scalable orchestration and deployment were realized as well as service function chain building chain. Experimental results show that in a data center with 10 000 servers, the scale-out orchestration time of SHOD system is reduced to $\frac{1}{12}$ of the existing system-Daisy, while only 1.5 times CPU utilization overhead is introduced.

Keywords: polymorphic network, NFV, service function chain, scalability, orchestrator and deployment system

收稿日期：2022−01−10；**修回日期**：2022−03−28

通信作者：杨芫，yangyuan_thu@mail.tsinghua.edu.cn

基金项目：国家重点研发计划基金资助项目（No.2019YFB1802504）；国家自然科学基金资助项目（No.61872209, No.61832013）

Foundation Items: The National Key Research and Development Program of China (No.2019YFB1802504), The National Natural Science Foundation of China (No.61872209, No.61832013)

0 引言

随着网络功能虚拟化（NFV, network function virtualization）技术和虚拟网络功能即服务（VNFaaS , virtual network function as a service）的兴起[1-2]，越来越多的公有云平台，如 Amazon Web Services、Google Cloud、Azure、AliCloud 等开始提供服务功能链（SFC, service function chain）服务。SFC 通常由多个虚拟网络功能（VNF, virtual network function）串联组成，如防火墙、负载均衡器等。多个串联的网络功能共同为租户的网络提供安全和性能优化等服务。过去，租户通常采购硬件网络功能在其本地网络部署 SFC。近年来，租户为了节省资本支出（CAPEX, capital expenditure）和运维成本（OPEX, operating expense），开始选择将 SFC 外包给公有云。在外包的过程中，租户向公有云提交 SFC 请求，而公有云的 SFC 编排部署系统负责在数据中心内编排、部署这些 SFC 请求。编排（又称资源编排）是指在众多功能基础设施中选择一部分设施来承载租户提出的 SFC 请求。具体而言，编排 SFC 的过程包括：1) 计算 SFC 中每个 VNF 应被部署于哪一台功能基础设施上（如服务器）；2) 计算 SFC 中相邻的 2 个 VNF 之间的网络转发路径。部署 SFC 的过程包括：1)在相应的功能基础设施上启动 VNF 进程实例；2)下发流表实现 SFC 建链[3]。

随着租户对个性化需求的提升和多元化终端的发展，多模态网络[4]这一概念被提出。多模态网络在数据层依托多种异构的全维可定义功能基础平台，借助控制层支持多模态寻址路由功能的控制器和服务层的“感知、决策、执行”一体的管理系统，满足租户专业化的需求。在多模态网络中，数据中心作为重要的场景，其中的 SFC 编排部署系统需要能够结合上层租户的需求和下层网络的服务能力来管理网络资源并承载业务。具体来说，多模态数据中心网络中的 SFC 编排部署系统面临以下两方面的问题。

首先，多模态网络的服务层需要结合上层租户的个性化需求实现 SFC 编排部署。近几年，租户逐渐要求 SFC 具有弹性扩容能力以应对流量的动态变化，公有云需要对租户的 SFC 进行扩容编排部署，从而保证租户的业务不受影响[5-6]。随着租户对服务质量需求的提升，在未来多模态网络中提供秒级的扩容编排部署变得十分重要。随着数据中心网络规模的增加，一个中型数据中心已经具备上万台服务器和上千台交换机，低性能的 SFC 编排部署系统设计很容易导致扩容编排部署时间过长，降低租户的服务体验。

其次，多模态数据中心网络存在异构的功能基础设施，如商用 X86 平台和基于 Tofino 芯片的 P4 交换机[7]等，并且未来可能会加入新型功能基础设施。多种异构功能基础设施加剧了数据中心的管理复杂程度，对 SFC 编排部署系统的软件设计提出了很大的挑战。

综上所述，支持多模态的数据中心网络中的 SFC 编排部署系统需要同时满足以下需求。

1) 管理万台服务器规模的数据中心，实现秒级的 SFC 弹性扩容编排部署。

2) 支持在异构功能基础设施上开展 SFC 的统一编排和部署，其中异构设施至少包括基于 X86 平台的服务器和基于可编程芯片的交换机。

如何在未来多模态的大规模数据中心网络中满足以上需求是 SFC 编排部署系统需要面对的重要问题。

学术界已经提出了一些 SFC 编排部署系统，如 Stratos[8]、NFV-RT[9]、Apple[10]、Hyper[11]、MicroNF[12]。Stratos、Apple 和 NFV-RT 的扩容编排时间复杂度较高且不支持异构 SFC 的编排部署。Hyper 和 MicroNF 能够对异构设备进行编排部署，但是它们不具备弹性扩容编排部署能力。Daisy[13]是为千台服务器规模的数据中心设计的 SFC 编排部署系统，然而它的可扩展性不足以胜任万台服务器规模的数据中心。Daisy 采用的编排算法的计算时间与拓扑规模近似且成正比，因此当拓扑规模较大时，Daisy 的编排时间会超过 1 min。例如，在 1 536 台服务器构成的 Leaf-Spine 数据中心网络中，Daisy 扩容 5 Tbit/s 的 SFC 所需的编排时间达到了 247 s。综上所述，以上编排部署系统均无法满足在万台服务器的数据中心网络内实现秒级扩容编排部署这一个性化需求。

在多模态网络中，异构 SFC 编排部署系统面临的最大挑战是高可扩展性。具体而言，本文的目标是在万台服务器规模的数据中心中实现秒级的扩容编排部署。为应对这一挑战，可以利用数据中心网络拓扑结构的特点对其进行分区，将每次编排决策的区域限制在一个分区内便可减少编排时间。进一步地，多个分区的编排任务可以通过增加编排算

法实例的方式实现并行的扩容编排，进一步减少 SFC 的扩容编排时间。

本文设计并实现了支持多模态网络环境的可扩展异构服务功能链并行编排部署（SHOD, scalable heterogeneous SFC parallel orchestration and deployment）系统。SHOD 系统将数据中心划分为多个不相交的分区，避免分区不连续或多个分区重叠导致的 SFC 编排失败，从而保证 SFC 编排部署的正确性。不同于已有的 SFC 编排部署系统采用的单一编排器设计方案，SHOD 系统采用分区并行编排的思路，通过负载均衡的方法将不同的 SFC 扩容请求分配到不同的编排器实例，保证耗时最长的一个编排器实例的编排时间在秒级。客观上，增加编排器数量必然会减少扩容编排时间，然而过多的编排器实例会占用过多的计算存储资源。为了最优化编排器实例数量，SHOD 系统构建了期望编排时间模型，根据该模型计算出满足期望编排时间的最少编排器数量。此外，SHOD 系统还采用了中介者设计模式来提供对异构 SFC 的支持，实现了对基于 X86 服务器、P4 可编程设备甚至新型多模态功能基础设施的网络功能的统一编排部署与建链。

本文的主要贡献如下。

1) 不同于已有 SFC 编排部署系统的单编排器设计，本文提出了基于并行编排器设计的可扩展 SFC 编排部署系统，可以支持万台服务器规模的数据中心的秒级扩容编排。

2) 不同于已有的异构 SFC 编排部署系统，SHOD 系统采用了符合单一权职原则的中介者设计模式，使新型异构功能基础设施及其控制器可以灵活加入系统，而不需要修改编排器。

3) 实现了 SHOD 原型系统，支持基于 Fastclick 的容器化 VNF[14]和基于 P4[15]可编程交换机的网络功能，实验结果显示，SHOD 系统可以实现秒级异构 SFC 编排部署。

1　相关工作

近年来，学术界出现了很多关于 SFC 编排部署系统的研究工作[8-13]。Stratos[8]是一个支持 SFC 水平扩展的 SFC 编排部署系统，其系统结构可以分为转发器控制器和资源管理器两部分。转发控制器负责向交换机下发流表并建立新的 SFC。资源管理器包括 SFC 资源瓶颈检测器和水平扩展模块。当检测器检测到服务器资源不足时，Stratos 会启动新的 VNF 实例；当检测到网络带宽不足时，Stratos 会将 SFC 迁移到网络不拥塞的区域。Stratos 的平均编排速度大约为每秒 67 条 SFC，不足以满足大规模数据中心内秒级的编排需求。

NFV-RT[9]是为数据中心网络设计的 SFC 编排部署系统，其系统结构包括控制器和资源管理器两部分。NFV-RT 对所有的 SFC 请求进行初始编排，编排过程分成 3 个阶段。首先，计算租户的 SFC 实例数量以满足租户的带宽请求；然后，将租户的所有 SFC 分配到数据中心的一个分发点（PoD, point of delivery）中；最后，确定每个网络功能实例部署的服务器。NFV-RT 不支持对可编程设备的网络功能进行扩容编排，同时其单编排器的设计导致 NFV-RT 缺乏在大规模拓扑下实现秒级扩容编排部署的能力。

Apple[10]是为软件定义网络（SDN, software defined network）设计的 SFC 编排部署系统，由资源编排器和 SDN 控制器组成。SDN 控制器包含动态处理器和优化引擎。动态处理器负责检测虚拟机的资源使用量是否超过阈值来判断是否需要进行 SFC 扩容。优化引擎通过运行求解器计算最优的编排部署方案，然而其编排时间复杂度是非多项式的，无法满足数据中心内的快速弹性扩容编排需求。

Hyper[11]是一个异构 SFC 编排部署系统，由编排器、中介者、硬件功能管理器、软件功能管理器和转发策略实施器组成。Hyper 同样采用了中介者来实现对不同的异构设备的管理，但是其中介者同时负责网络功能的编排部署和拓扑收集，不符合软件工程中的单一权职原则，造成系统过于僵化。同时，Hyper 还将 SFC 建链的功能划分到编排器中，将网络功能编排的功能划分到中介者中，这种设计导致 Hyper 对基于求解器的编排算法无法提供很好的支持，降低了 Hyper 的应用范围。同时，Hyper 的单编排器设计无法支持大规模数据中心的秒级弹性扩容编排。

MicroNF[12]是一种异构 SFC 建链系统。它通过微服务架构来承载不同的分组处理模块，通过消除 SFC 中的重复模块并采用可编程网卡降低了 SFC 的时延；同时提出了策略解析器屏蔽底层设备的异构特性，减轻编排器的压力。然而，MicroNF 并未对大规模的数据中心编排进行设计，导致其缺乏扩展性。

Daisy[13]是一个高可扩展的异构 SFC 编排部署

系统，支持对 SFC 进行水平扩容和流量工程。Daisy 提出了 3 个算法实现 SFC 的编排，分别是随机分配算法、NetPack 算法和 VNFSolver 算法。其中，NetPack 算法可以在较短的时间内实现 SFC 编排，然而在大规模的数据中心拓扑下，其编排时间依旧会超过 1 min，不足以满足未来多模态网络数据中心的秒级弹性扩容编排需求。

2 系统设计

2.1 总体结构设计

SHOD 系统是一个可以应用于多模态数据中心网络中的高可扩展异构 SFC 编排部署系统。SHOD 系统的总体设计思想主要有以下两点。1) 利用中介者设计模式实现服务层模块对异构控制器组的统一调用，并且按照单一权职原则对中介者进行设计，以提升系统各模块的内聚性。2) 利用分区并行编排系统的设计，实现对大规模数据中心的秒级扩容编排，以提高 SHOD 的可扩展性。

SHOD 总体结构如图 1 所示。从整体上看，SHOD 依照多模态智慧网络的技术体系框架[4]而采用分层设计，包括服务层模块和控制器组两部分。服务层模块主要负责响应租户的 SFC 请求，并针对数据中心的流量变化做出扩容调整。控制器组负责根据服务层模块的编排结果，将 SFC 部署到相应的数据中心异构功能基础设施上。SHOD 系统结构的创新点主要体现在服务层模块的独特设计，具体如下。

1) 不同于已有的 SFC 编排部署系统，SHOD 系统采用了多编排器的设计，通过编排请求预处理器对编排请求的负载均衡功能实现了并行化编排，大大提高了扩容编排速度。

2) 相对于僵化的传统 SFC 编排部署系统，SHOD 系统创新性地采用单一权职原则的中介者设计模式，由此设计出了统一服务功能链部署模块。该模块定义了和控制器交互的统一应用程序接口（API, application programming interface），屏蔽了异构控制器的具体实现，使 SHOD 系统更容易添加未来新型的多模态网络基础设施控制器，大大增加了系统的灵活性。

2.2 服务层模块

SHOD 系统的服务层模块包括控制面板、编排请求预处理器、编排管理数据库、编排器、统一服务功能链部署模块、测量模块和扩容模块。

控制面板是 SHOD 系统的 Web 前端部分。租户可以登录控制面板页面提交 SFC 编排部署请求。租户的 SFC 请求内容包括 SFC 中的网络功能种类、网络功能之间的连接顺序，以及 SFC 的带宽需求。随后，租户的 SFC 编排部署请求会被发送到编排请求预处理器。

编排请求预处理器负责对所有租户的 SFC 扩容编排部署请求进行预处理。这一过程包括给每个租户的请求分配一个通用唯一识别码（UUID, universally unique identifier），将租户的请求注册到编排管理数据库。为满足未来多模态网络中的秒级扩容编排需求，编排请求预处理器还需要将所有租户的请求负载均衡到各个编排器实例中，同时保证任意一个编排器的编排时间为秒级。

编排器负责对租户的 SFC 扩容请求进行编排。对于租户请求的 SFC，编排器将其视为逻辑 SFC；然后，编排器根据租户的逻辑 SFC 带宽需求计算对应的 SFC 实例数量，以及网络功能实例和异构功能基础设施节点之间的映射关系；最后，编排器计算

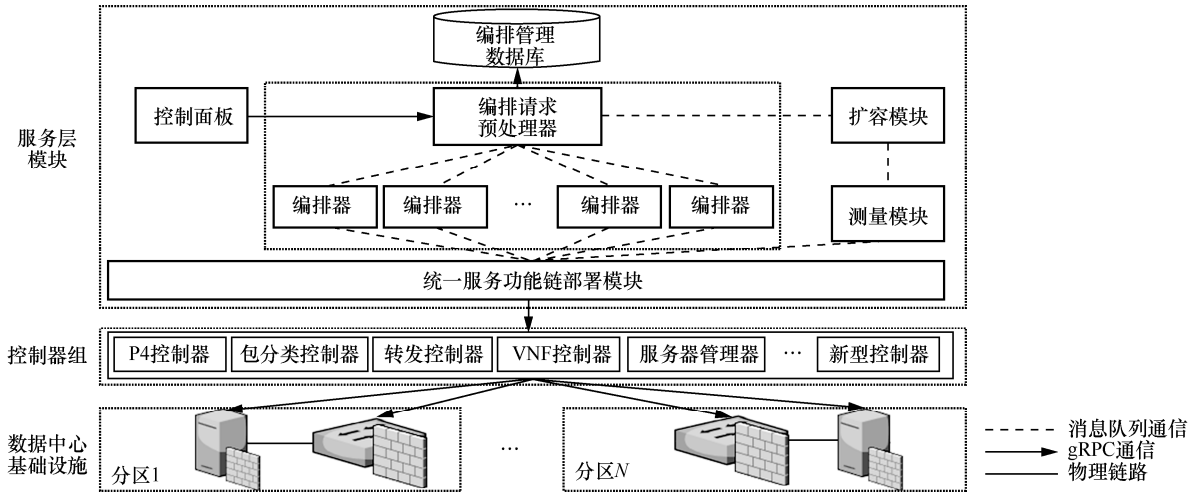


图 1 SHOD 总体结构

网络功能实例之间的网络转发路径。为了提升编排系统的可扩展性，SHOD 系统采用了多编排器实例的设计。在完成对租户 SFC 的扩容编排之后，编排器会将编排结果填入扩容部署请求命令，并发送给统一服务功能链部署模块。

统一服务功能链部署模块是连接服务层模块与控制器组的重要枢纽，专门为异构功能基础设施环境设计。由于不同种类功能基础设施的部署接口不同，需要为每种功能基础设施实现一个控制器，由此产生的多种异构设备控制器给编排部署带来了软件系统灵活性方面的挑战。考虑到未来多模态数据中心网络会不断地加入新型设备和更多第三方厂商的网络功能，SFC 编排部署系统必须能够灵活添加新型控制器。为了满足这一需求，SHOD 系统采用中介者模式设计了统一服务功能链部署模块，它连通了 SHOD 系统的服务层模块和控制器组。通过规定统一的 SFC 部署接口，SHOD 系统实现了对底层设备和控制器的封装，避免了为特定设备开发特定服务层模块而带来的系统僵化问题。

具体地，统一服务功能链部署模块负责对来自编排器的 SFC 扩容部署请求进行复制和分发，即为每一个控制器复制一个 SFC 扩容部署请求命令，并将该命令发送到对应的控制器上，包括 P4 部署命令、包分类部署命令、转发路径部署命令、VNF 部署命令和服务器部署命令。SHOD 系统的统一服务功能链部署模块是符合单一权职原则的，它仅负责请求命令的复制和分发。为实现对异构设备控制器的抽象隔离，统一服务功能链部署模块对控制器提供了统一的 API，具体如下。

1) ADD_SFC(SFC)，添加逻辑 SFC。

2) ADD_SFCI(SFC_instance)，添加 SFC 实例。

3) DEL_SFCI(SFC_instance)，删除 SFC 实例。

4) DEL_SFC(SFC)，删除逻辑 SFC。

5) GET_SERVER_SET()，获取服务器信息。

6) GET_TOPOLOGY()，获取数据中心拓扑。

7) GET_SFCI_STATE(SFC_instance)，获取 SFC 实例的流量和状态信息。

随着新型设备的出现，SHOD 系统不需要对统一编排部署模块进行修改。根据软件工程的依赖倒置原则，仅需要开发符合统一服务功能链部署模块 API 的设备控制器，即可实现对新型设备的控制，满足了未来多模态数据中心网络中灵活添加新型功能基础设施的需求。

测量模块负责收集数据中心的所有信息，包括异构功能基础设施（如服务器）的状态信息、交换机组成的拓扑信息、每条 SFC 实例的状态信息和流量信息。不同设备的信息收集功能由各个设备的控制器封装并通过统一的接口提供。

扩容模块主要根据测量模块测量到的 SFC 流量信息，依据预先指定好的规则决策是否对租户的 SFC 进行扩容。SHOD 采用对网络功能实例进行性能建模的方法得到单一 SFC 实例可以处理的最大流量。当扩容模块检测到某一条 SFC 实例的输入流量超过了其最大可承载流量时，扩容模块会生成添加 SFC 实例的请求并发送给编排请求预处理器。

2.3　并行编排设计

并行 SFC 编排是 SHOD 系统为满足未来多模态数据中心网络的秒级编排需求而提出的核心设计。并行 SFC 编排的基本思想是通过将数据中心进行分区来实现分区间的并行编排。然而，数据中心拓扑并不能被随意划分，原因如下。1) 错误的划分方式可能导致分区的不连通，进而导致分区内编排失败；2) 如果划分的任意 2 个分区之间有重叠，则会导致并行编排出错，例如，2 条 SFC 实例被 2 个编排器实例编排到同一台服务器上，但是该服务器的资源不足以承载这 2 条 SFC 实例。

为此，SHOD 系统采用了数据中心不相交分区的方法。不相交分区方法的主要流程为：1)以 POD 为最小粒度进行分区；2) 合并相邻的 POD 组成更大的分区，直到满足最大可接受的拓扑规模。

相比于已有的 SFC 编排部署系统，SHOD 系统的并行编排计算速度更快，可以在万台服务器规模的数据中心将期望扩容编排时间降低至秒级。然而多个编排器实例会占用更多的 CPU 和内存资源。过多的额外 CPU 和内存消耗会影响 SFC 编排部署系统中其他模块的性能。为减少并行编排占用的资源，需确定满足期望编排时间的最小编排器数量。

为此，SHOD 系统采用了基于数据驱动的方法建立期望编排时间模型来确定最小的编排器数量。建模的总流程为：首先，分析编排算法，并给出编排算法的期望计算时间模型；然后，通过构造各种 SFC 扩容编排部署请求，并记录编排算法的计算时间，来实现对编排算法的期望计算时间的拟合。本文提出的基于数据驱动的期望编排时间模型建立方法适用于任意类型的编排算法，对于如何提升编排算法本身的计算速度已经超出本文工作的研究范围。

本文以 Daisy 使用的算法 NetPack[13]为例阐述最小编排器数量的确定方法，该方法需要分析编排算法来建立期望编排时间模型。NetPack 算法可以分为两层循环。

第一层循环根据所有的服务器构建服务器集合进行编排。在第一层循环内，针对每一个网络功能，NetPack 会无放回地从服务器集合中随机选择一台服务器。设 N_{se} 为数据中心服务器的数量，其中有 $N_{se}^{'}$ 台服务器的资源不足以承载租户请求的 SFC，C 为 SFC 的长度。每当 NetPack 算法随机选择的服务器是 $N_{se}^{'}$ 台资源不足的服务器之一时，NetPack 会继续重新随机选择，直到选择到资源充足的服务器。根据这一过程，可以得出 NetPack 算法随机选择服务器的期望选择次数 E_{cnt} 为

$$E_{cnt}=\frac{\sum_{i=1}^{N_{se}^{'}} i\binom{N_{se}^{'}-i+1}{N_{se}-i}}{\binom{N_{se}^{'}}{N_{se}}} \tag{1}$$

其中，i 表示第 i 次选择到资源充足的服务器。NetPack 算法在 $i=1$ 时立即选择到资源充足的服务器的概率为

$$P(i=1)=\frac{\binom{N_{se}^{'}}{N_{se}-1}}{\binom{N_{se}^{'}}{N_{se}}}=\frac{N_{se}-N_{se}^{'}}{N_{se}} \tag{2}$$

则式(1)的上界为

$$E_{cnt}\leqslant P(i=1)+(1-P(i=1))\times(N_{se}^{'}+1)=N_{se}^{'}(1-P(i=1))+1 \tag{3}$$

由于数据中心的服务器利用率普遍较低，资源不足的服务器数量 $N_{se}^{'}$ 通常远小于总服务器数量 N_{se}，即 $N_{se}^{'}\ll N_{se}$，此时 $P(i=1)\approx 1$，则 $E_{cnt}\approx 1$。

第二层循环中，NetPack 算法调用深度优先搜索（SFC, depth first search）实现无向图中的 SFC 实例路径计算。在满足 SFC 请求带宽 BW 远小于数据中心可用带宽 BW_{ava} 的条件下，即 $BW\ll BW_{ava}$ 时，NetPack 算法的第二层循环仅需要进行一次复杂度为 $O(N_{sw})$ 的 DFS，其中 N_{sw} 为数据中心交换机的数量。

综上所述，在满足条件 $BW\ll BW_{ava}$ 和 $N_{se}^{'}\ll N_{se}$ 的情况下，NetPack 计算一条 SFC 实例的期望计算时间复杂度为 $O(N_{sw}C)$。该期望计算时间复杂度和 Daisy[13]的实验结果相符合：Daisy 的实验结果显示，NetPack 算法的计算时间与拓扑规模 N_{sw} 近似成正比。基于此计算时间复杂度，本文方法建立的扩容编排时间模型 $E(t)$ 为

$$E(t)=k_cMN_{sw}C+T_o \tag{4}$$

其中，k_c 是拟合系数；M 是需要扩容的 SFC 实例数量；T_o 是编排系统的固有时间，即租户的扩容编排请求进入编排系统到扩容部署请求命令被发送到统一服务功能链部署模块之间的时间差。

为了验证式(4)能否准确描述 SHOD 系统的编排时间，本文通过比较模型计算出的期望编排时间与实际的编排时间平均值之间的均方误差（MSE, mean square error）来验证其准确性。本文分别在 5 个不同规模的 Fat-Tree 拓扑[16]下采集了 NetPack 算法的编排时间数据，其中 POD 的数量 K 分别为 20、24、28、32、36。在每个拓扑下，本文分别测量了扩容 100 条、300 条、500 条、700 条、900 条 SFC 实例的编排时间数据，其中扩容 100 条、300 条、700 条、900 条的编排时间作为训练集，扩容 500 条的编排时间作为测试集。具体地，模型拟合系数、固有时间及均方误差如表 1 所示。

表 1 模型拟合系数、固有时间及均方误差

K/个	k_c	T_o/s	MSE/s
20	7.4×10^{-6}	1.85	2.0×10^{-2}
24	8.0×10^{-6}	1.85	6.7×10^{-4}
28	1.3×10^{-5}	1.85	1.3×10^{-2}
32	1.8×10^{-5}	1.85	7.2×10^{-1}
36	2.6×10^{-5}	1.85	1.3

从表 1 可以看出，本文所提模型的最大 MSE 仅为 1.3 s。$K=36$ 的 Fat-Tree 拓扑下的模型预测时间与真实编排时间的比较如图 2 所示。

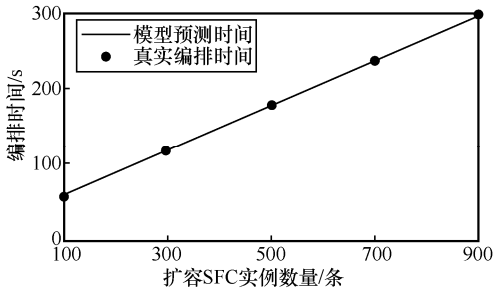


图 2 模型预测时间与真实编排时间的比较

从图 2 中可以看出，本文模型预测时间和真实编排时间是吻合的，基于式(4)的模型可以较好地描述系统期望编排时间。

基于本文模型可以推导出编排器实例数量的计算式。假设期望在 T s 内编排 M 条 SFC 实例进行扩容，同时要保证每个编排器处理的 SFC 实例数量相同。设编排器数量为 n，则每个编排器的编排区域中的设备数量大约是总设备数量的 $\frac{1}{n}$，处理的 SFC 实例数量为总 SFC 实例数量的 $\frac{1}{n}$，则可以得到编排器实例数量应满足

$$n \geqslant \sqrt{\frac{k_c M N_{sw} C}{T - T_o}} \tag{5}$$

取满足式(5)的最小整数作为编排器的数量，在得到最小的编排器数量后，编排器预处理器会将 M 条 SFC 实例分配到 n 个编排器实例中进行并行编排。

2.4　控制器组

为了实现对异构设备的统一部署，SHOD 系统为每一种设备都提供了对应的控制器。SHOD 系统按照基础设施的种类将控制器划分为 5 种，分别是服务器控制器、包分类控制器、P4 控制器、VNF 控制器和转发控制器。服务器控制器负责收集每台服务器的状态信息和资源信息，并把全部信息发送到测量模块。包分类控制器用于数据中心网关的流量分类，负责在网关下发规则，并将不同的 SFC 流量进行区分。P4 控制器和 VNF 控制器是实现 SFC 部署的重要组件。编排器输出的 SFC 实例编排结果中，有一部分 SFC 实例被编排到了 P4 交换机上，这些网络功能由 P4 控制器负责部署。P4 控制器会在相应的 P4 交换机上下发网络功能对应的规则，如访问控制列表（ACL, access control list）。另一部分 SFC 实例被编排到基于 X86 的商用服务器上，这些虚拟网络功能由 VNF 控制器部署。

网络功能的部署时间不仅与功能基础设施的种类有关，也和网络功能的数量有关。通常，在一台基础设施上部署的网络功能数量越多，需要的部署时间就越长。基于这一观察可知，仅仅通过优化部署模块无法保证秒级的 SFC 部署，还需要避免编排器在同一台基础设施上编排过多的网络功能实例。为了实现秒级的异构 SFC 部署，针对任意一种功能基础设施，需要得到其部署网络功能的数量和部署时间之间的对应关系。SHOD 系统通过实际测量的方法得到每台功能基础设施的部署网络功能数量和相应的部署时间。根据这一对应关系，可以根据期望部署时间得到每台功能基础设施的最大可部署网络功能数。通过将可部署网络功能数量的限制条件添加于 SHOD 系统的编排算法中，可以保证在期望时间内完成 SFC 的部署。以 NetPack 算法为例，在第一层循环算法判断一台服务器资源是否充足。SHOD 系统在此处加入一个判断语句来判断一台服务器的最大可部署网络功能数量。如果已经超过了该数量限制，则这一层循环不选择该服务器。本文在实验部分展示了在测试床中测量的异构基础设施网络功能部署时间与网络功能数量之间的对应关系。

在完成对网络功能的部署之后，还需要转发控制器实现 SFC 的建链。目前，已经有很多种 SFC 建链方法。考虑到易部署性，SHOD 系统采用了网络服务头（NSH, network service header）[17]实现跨设备的建链，该方法支持多达 2^{24} 条 SFC，可以满足大规模数据中心的基本需求。转发控制器将 NSH 转发规则下发到服务器所在的软件交换机上实现 NSH 建链。

3　系统原型实现

本文采用 Python 语言实现了 SHOD 原型系统。原型系统基于 RabbitMQ 消息队列作为服务层模块以及控制器之间的通信机制。采用 MySQL 作为编排管理数据库。在控制层中，P4 控制器采用 gRPC 实现了对 P4 的远程控制，转发控制器同样使用 gRPC 实现了对基于 BESS 的软件交换机的控制，包分类控制器采用 gRPC 对网关进行管理，VNF 控制器则采用 Docker API 实现在远程服务器上启动虚拟网络功能容器。

本文实现了基于 BESS 的软件交换机用于 SFC 建链，还采用 FastClick 实现了多种 VNF，包括无状态防火墙、负载均衡器、Monitor 等。此外，还实现了基于 Tofino S9180-32X P4 交换机的无状态防火墙用于部署 ACL。

4　系统性能评估

4.1　实验设置

本文主要通过真实实验来验证 SHOD 系统的可扩展性，并通过小规模测试床评估了 SHOD 系统的扩容部署时间。真实实验中，SHOD 系统被部署于一台 Intel Silver 4210R 服务器上，编排管理数据库载入了真实数据中心拓扑，即常见的 Fat-Tree[16] 拓扑。对于拥有 K 个 POD 的 Fat-Tree 拓扑，每个 POD 由一层 $\frac{K}{2}$ 个机架交换机（ToR, top of rack）和

$\frac{K}{2}$ 个聚合交换机组成完全二分图。此外，还有 $\frac{K^2}{4}$ 个核心交换机，每个核心交换机分别与每个 POD 中的一台聚合交换机相连。为评估不同拓扑规模下 SHOD 系统的弹性扩容编排时间，本文分别在 20 个、24 个、28 个、32 个和 36 个 POD 的 Fat-Tree 拓扑下进行了实验。其中，$K=36$ 的 Fat-Tree 拓扑包含 1 620 台交换机和 11 664 台服务器。机架交换机和聚合交换机之间的链路带宽设置为 10 Gbit/s，聚合交换机和核心交换机之间的链路带宽设置为 10 Gbit/s，每台核心交换机都可作为数据中心的网关，它们负责与主干网的连接。租户的流量由数据中心网关进入数据中心网络。租户的 SFC 请求带宽大小设置为服从 10～100 Mbit/s 分布的随机变量。本文随机生成了 100 条 SFC，并按照已有的研究[18-19]将生成的 SFC 长度范围限制在 2～7，SFC 中的网络功能主要由 ACL 防火墙、NAT、Monitor、WAN 优化器、IDS、VPN 和负载均衡器组成。为了测量 SHOD 的可扩展性，本文测量了 SHOD 对所有 SFC 扩容 2～10 倍所需的扩容编排时间，并设定扩容编排期望时间 T 为 25 s。本文通过 SHOD 系统的消息接口向编排请求预处理器提交所有的 SFC 扩容编排请求，并在统一服务功能链部署模块处记录所有请求的编排时间。本文的对比方案包括已有的编排系统 Daisy 和运行 NetSolver-ILP[20]算法的编排系统。

为进一步考察 SHOD 系统的扩容部署时间，本文搭建了小规模测试床。该测试床由一台商用交换机、一台 Tofino S9180-32X P4 交换机、一台 Intel E5-2603v4 服务器和一台 Intel Silver 4210R 服务器组成。其中，商用交换机用于连接 P4 交换机和服务器的控制平面，P4 交换机用于部署 ACL 网络功能，第一台服务器用于部署虚拟网络功能，第二台服务器用于运行 SHOD 系统。

除非在实验中有特殊说明，否则默认的 SFC 长度为 7，初始 SFC 实例数量为 100，默认需要扩容 900 条 SFC 实例，拓扑为 $K=36$ 的 Fat-Tree。实验一共进行 5 次，取平均值和标准差作为最终结果。

4.2 异构 SFC 扩容部署时间

在测试床中，本文测量了异构 SFC 的扩容部署时间。部署的 SFC 包括一个基于 P4 类型的无状态防火墙以及一个基于 VNF 类型的无状态防火墙，扩容 SFC 实例数量分别为 1～5 个。根据阿里云提供的防火墙实例的规则数量范围，本文设置每个防火墙实例的 ACL 规则为 100 条。图 3 展示了在不同扩容 SFC 实例数量下，基于 P4 类型和基于 VNF 类型的无状态防火墙的扩容部署时间。从图 3 可以看出，即使在扩容 SFC 实例数量为 5 时，SHOD 的 SFC 扩容部署依然可以在秒级时间内完成。通过测量，本文发现基于 P4 的无状态防火墙的扩容部署时间远远小于基于 X86 的扩容部署时间。P4 类型的网络功能扩容部署时间不超过 1 s，而 VNF 类型的网络功能扩容部署时间比较长，这是因为本文采用 DPDK 实现的 VNF 类型的无状态防火墙需要较长的大页内存初始化过程。

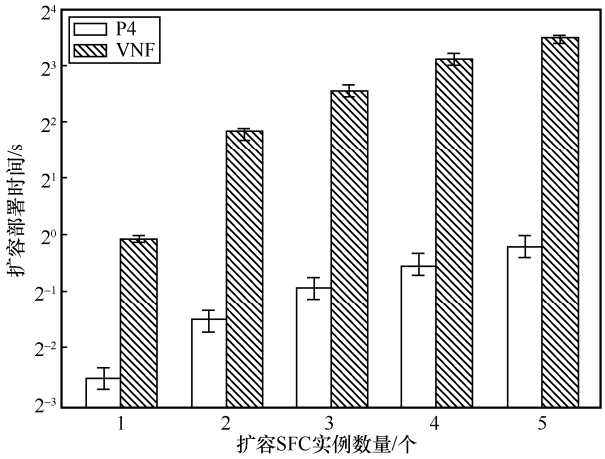


图 3 扩容部署时间与扩容 SFC 实例数量的关系

除此以外，随着扩容 SFC 实例数量的增加，扩容部署时间也随之增加。具体地，当扩容 SFC 实例数量为 1 时，VNF 的扩容部署时间仅需 1 s，而当扩容 SFC 实例数量大于或等于 2 时，每条 SFC 实例的平均部署时间大约为 2.3 s。造成这一现象的原因是本文所实现的 VNF 控制器原型采用了 docker API，而 docker API 在单台服务器上是串行调用的。当仅需部署一条 SFC 实例时，docker API 可以立即返回部署成功的指令。而当部署第二条及之后的 SFC 实例时，docker API 需要等待前一个 SFC 实例部署完成，然后才能进行下一个 SFC 实例的部署。

在得到不同功能基础设施的部署时间与 SFC 实例数量之间的关系后，可以通过限制每台设备的网络功能实例可编排数量来保证秒级部署时间。例如，当期望部署时间限制在 11 s 内时，测试床平台中的 P4 交换机的网络功能实例部署数量最多为 50 个，而服务器的网络功能实例部署数量最多为 5 个。通过限制编排器在每台设备的最大网络功能编排数量，SHOD 可以保证在任意一台设备上部署 SFC 的时间为秒级。

4.3　扩容编排时间

在测试床实验中，SHOD 系统和 Daisy 系统的编排时间相差不大，均约为 2 s。这是因为测试床的规模较小，无法体现出 SHOD 系统的可扩展性。为此，本文在大规模拓扑下进一步比较编排时间。

在测量大规模拓扑下的扩容编排时间之前，本文首先通过模型确定最小资源使用量的编排器数量。为确定最小的编排器数量，SHOD 系统针对每一个拓扑都建立了一个期望计算时间模型，具体拟合系数如表 1 所示。SHOD 系统在不同 POD 数量 K 下的最小编排器数量如表 2 所示。

表 2　　最小编排器数量

K/个	n/个
20	2
24	2
28	2
32	3
36	4

在得到最小化资源使用量的编排器数量之后，本文测量了在不同拓扑规模下，SFC 编排系统 Daisy 和 SHOD 的 SFC 扩容编排时间，同时本文还比较了基于 Gurobi 求解器的编排算法 NetSolver-ILP[20]，实验结果如图 4 所示。从图 4 可以看出，采用 NetSolver-ILP 算法的编排时间是最长的，在最大的拓扑下其编排时间可达 8 h，远超期望编排时间限制。NetSolver-ILP 的编排时间主要由求解器模型的构建时间和模型的求解时间两部分组成。由于在大规模拓扑下，最优化模型的限制条件和决策变量数非常大，导致求解器模型的构建时间就已经超出了秒级，而模型的求解时间则比求解器模型的构建时间更多。Daisy 的 SFC 扩容编排时间会随着拓扑规模的增长而急剧增长，在 $K = 36$ 的 Fat-Tree 拓扑下，Daisy 的扩容编排时间达到了 300 s，远超秒级扩容编排的要求。与之相对应，SHOD 的扩容编排时间并不会随着拓扑规模的增长而增加。在所有拓扑下，SHOD 的扩容编排时间均小于 25 s。具体地，按照拓扑从小到大分别是 16 s、17 s、23 s、21 s 和 24 s。SHOD 能够将编排时间降低至秒级的原因在于其采用了并行编排的设计，随着拓扑规模的增加，SHOD 会相应地增加编排器实例的数量来降低扩容编排时间。

本文还考察了不同扩容 SFC 实例数量下的扩容编排时间，如图 5 所示。基于求解器的编排算法 NetSolver-ILP 的编排时间依旧远超分钟级。随着扩容 SFC 实例数量的增加，Daisy 的扩容编排时间也会线性增加。这是因为 Daisy 采用的 NetPack 编排算法的期望编排时间与需要扩容的 SFC 实例数量呈正比例关系。与 Daisy 不同，SHOD 的扩容编排时间可以保持在期望编排时间以内，这是因为 SHOD 可以通过增加编排器实例的方法降低编排时间，相比于单编排器设计的 Daisy，SHOD 在大规模数据中心网络中的可扩展性更好。

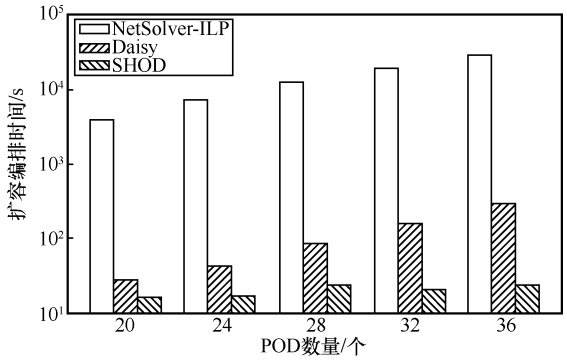


图 4　不同拓扑规模下的扩容编排时间

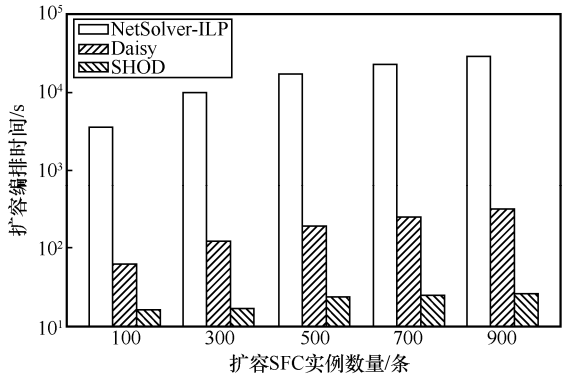


图 5　不同扩容 SFC 实例数量下的扩容编排时间

4.4　系统 CPU、内存资源占用开销

在小规模测试床实验中，SHOD 系统和 Daisy 系统的 CPU 占用率和内存占用率基本相同，大约为 5%和 90 MB。

在大规模拓扑下，为了将 SFC 扩容编排时间降低至秒级，SHOD 采用了多编排器设计。多个编排器会占用额外的计算存储资源。本文评估了在不同拓扑规模下，SHOD 的 CPU 占用率和内存利用量。

平均 CPU 占用率如图 6 所示。从图 6 可以看出，随着拓扑规模的增加，NetSolver-ILP 的 CPU 占用率基本保持在 125%～150%，这是由于 NetSolver-ILP 使用的 Gurobi 求解器会使用到多个核心并发求解优化模型，因此其 CPU 占用率会超过 100%。Daisy 的

CPU 占用率会随着拓扑规模的增加而逐渐接近100%，这是因为 Daisy 仅使用一个编排器运行其NetPack 算法，因而其 CPU 占用率不会超过 100%。SHOD 的 CPU 占用率虽然会随着拓扑规模的增加而缓慢增加，但是从整体上看 SHOD 的 CPU 占用率与其他对比方案都在同一数量级内，例如 $K=36$ 的 Fat-Tree 拓扑下，SHOD 的平均 CPU 占用率相比于 Daisy 仅提高了 1.5 倍，相比于 NetSolver-ILP 仅提高了 70%。这是因为 SHOD 采用了数据驱动的建模方式计算出了资源使用量最小化的编排器数量，使 SHOD 的 CPU 占用率和其他对比方案大致相同。

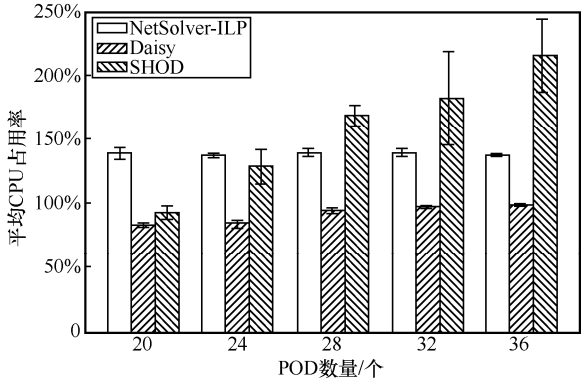


图 6 平均 CPU 占用率

平均内存资源使用量如图 7 所示。其中，在最小的拓扑下，NetSolver-ILP 使用了 14.1 GB 内存。在最大的拓扑下，其内存资源使用量最高可达到 64.5 GB。NetSolver-ILP 使用大量内存的原因在于其求解器建模的优化模型有大量的决策变量和限制条件。相比之下，采用 NetPack 算法的 Daisy 和 SHOD 的平均内存资源使用量最大仅约为 860 MB，远小于 NetSolver-ILP。而 SHOD 的内存资源使用量相比于 Daisy 仅仅提高了 3 倍，这对于当今的大内存商用服务器来说是完全可以接受的。

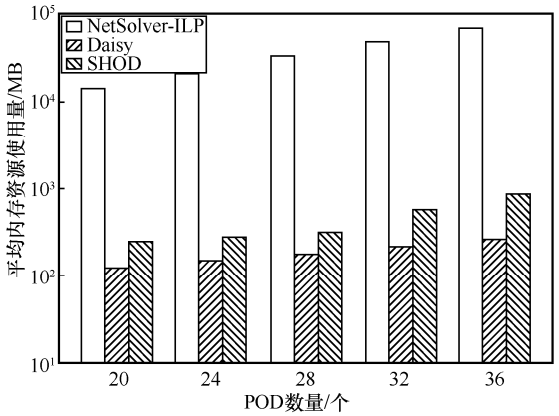


图 7 平均内存资源使用量

5 结束语

本文为未来多模态网络中的大规模复杂公有云提出了一种可扩展的弹性异构服务功能链编排部署系统。通过并行的编排器设计，实现了对大规模网络的弹性扩容编排。通过符合单一权职原则的中介者设计模式，实现了对异构网络设备的统一编排部署和建链。实验结果表明，SHOD 系统可以在秒级完成对万台服务器规模的数据中心的服务功能链扩容编排部署。

参考文献：

[1] SHERRY J, HASAN S, SCOTT C, et al. Making middleboxes someone else's problem[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13-24.

[2] LAN C, SHERRY J, POPA R A, et al. Embark: securely outsourcing middleboxes to the cloud[C]//Proceedings of 13th Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2016: 255-273.

[3] ETSI N. Network functions virtualization (NFV); management and orchestration: ETSI GS NFV-MAN 001 V1. 1.1[S]. 2014.

[4] 胡宇翔, 伊鹏, 孙鹏浩, 等. 全维可定义的多模态智慧网络体系研究[J]. 通信学报, 2019, 40(8): 1-12.
HU Y X, YI P, SUN P H, et al. Research on the full-dimensional defined polymorphic smart network[J]. Journal on Communications, 2019, 40(8): 1-12.

[5] KABLAN M, ALSUDAIS A, KELLER E, et al. Stateless network functions: breaking the tight coupling of state and processing[C]//Proceedings of 14th Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2017: 97-112.

[6] WOO S, SHERRY J, HAN S, et al. Elastic scaling of stateful network functions[C]//Proceedings of 15th Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2018: 299-312.

[7] GAO J Q, ZHAI E N, LIU H H, et al. Lyra: a cross-platform language and compiler for data plane programming on heterogeneous ASICs[C]//Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM Press, 2020: 435-450.

[8] GEMBER A, KRISHNAMURTHY A, JOHN S S, et al. Stratos: a network-aware orchestration layer for virtual middleboxes in clouds[J]. arXiv Preprint, arXiv:1305.0209, 2013.

[9] LI Y, XUAN P L T, LOO B T. Network functions virtualization with soft real-time guarantees[C]//Proceedings of The 35th Annual IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2016: 1-9.

[10] LI X, QIAN C. An NFV orchestration framework for interference-free policy enforcement[C]//Proceedings of IEEE 36th International Conference on Distributed Computing Systems. Piscataway: IEEE Press, 2016: 649-658.

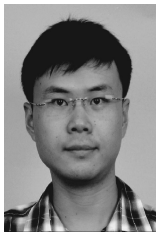
[11] SUN C, BI J, ZHENG Z L, et al. HYPER: a hybrid high-performance framework for network function virtualization[J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2490-2500.

[12] 孙晨, 毕军, 郑智隆, 等. MicroNF: 基于微服务的异构网络功能虚

拟化框架[J]. 通信学报, 2019, 40(8): 54-59.
SUN C, BI J, ZHENG Z L, et al. MicroNF: a microservice-based hybrid framework for NFV[J]. Journal on Communications, 2019, 40(8): 54-59.
[13] KODIROV N, BAYLESS S, RUFFY F, et al. VNF chain allocation and management at data center scale[C]//Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems. New York: ACM Press, 2018: 125-140.
[14] BARBETTE T, SOLDANI C, MATHY L. Fast userspace packet processing[C]//Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communications Systems. Piscataway: IEEE Press, 2015: 5-16.
[15] BOSSHART P, DALY D, GIBB G, et al. P4[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.
[16] AL-FARES M, LOUKISSAS A, VAHDAT A. A scalable, commodity data center network architecture[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63-74.
[17] QUINN P, ELZUR U, PIGNATARO C. Network service header (NSH)[R]. RFC Editor, 2018.
[18] UTTARO J, STIEMERLING M, NAPPER J, et al. Service function chaining use cases in mobile networks[R]. IETF, 2015.
[19] KUMAR S, TUFAIL M, MAJEE S, et al. Service function chaining use cases in data centers[R]. IETF SFC WG, 2015.
[20] BAYLESS S, KODIROV N, IQBAL S M, et al. Scalable constraint-based virtual data center allocation[J]. Artificial Intelligence, 2020, 278: 103196.

[作者简介]

陈浩（1995− ），男，北京人，清华大学博士生，主要研究方向为网络功能虚拟化、高可用服务功能链。



杨芫（1984− ），男，四川乐至人，博士，清华大学助理研究员，主要研究方向为互联网体系结构、下一代互联网、路由与传输协议。

徐明伟（1971− ），男，辽宁朝阳人，博士，清华大学教授，主要研究方向为网络体系结构、下一代互联网、网络空间安全。

裴丹（1973− ），男，河北唐山人，博士，清华大学副教授，主要研究方向为基于机器学习的智能运维。

尤艺霖（1999− ），男，山东淄博人，清华大学硕士生，主要研究方向为可编程网络交换机、高性能网络功能。